ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και υλοποίηση συστήματος συλλογής, επεξεργασίας και απεικόνισης περιβαλλοντικών δεδομένων από δίκτυο ασύρματων αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αικατερίνη Ε. Πλευράκη

Επιβλέπων : Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και υλοποίηση συστήματος συλλογής, επεξεργασίας και απεικόνισης περιβαλλοντικών δεδομένων από δίκτυο ασύρματων αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αικατερίνη Ε. Πλευράκη

Επιβλέπων : Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Σεπτεμβρίου 2011

.....	.....	.....
Φίλιππος Κωνσταντίνου	Αθανάσιος Παναγόπουλος	Ιωάννης Κανελλόπουλος
Καθηγητής Ε.Μ.Π.	Λέκτωρας Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2011

.....

Αικατερίνη Ε. Πλευράκη

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Τα Ασύρματα Δίκτυα Αισθητήρων(Wireless Sensor Networks-WSN) αποτελούνται από χωρικά κατανεμημένους αυτόνομους αισθητήρες με καθήκον την παρακολούθηση διάφορων συνθηκών. Σήμερα, τα WSN χρησιμοποιούνται ευρέως σε πολλές βιομηχανικές και καταναλωτικές εφαρμογές. Ειδικά στην επιστήμη υπολογιστών και στις τηλεπικοινωνίες, αποτελούν ένα ενεργό τομέα έρευνας με μεγάλη δραστηριότητα. Σκοπός αυτής της εργασίας είναι η μελέτη της ασύρματης μετάδοσης περιβαλλοντικών δεδομένων από ένα τέτοιο δίκτυο προς ένα κέντρο συλλογής, η περαιτέρω επεξεργασία, αποθήκευση και τελικά απεικόνιση τους σε μορφή φιλική προς το χρήστη και ευνοϊκή για την εξαγωγή συμπερασμάτων. Τα μεγέθη τα οποία μετρούνται είναι η θερμοκρασία, η υγρασία, η ολική ηλιακή ακτινοβολία, η ενεργή φωτοσυνθετικά ακτινοβολία και η εσωτερική τάση και θερμοκρασία της μονάδας-αισθητήρα.

Αρχικά πραγματοποιείται μια σύντομη μελέτη πάνω στον κόμβο-αισθητήρα και τα συστατικά του μέρη, η οποία στη συνέχεια επεκτείνεται στο δίκτυο αισθητήρων. Μέσα σε αυτό το πλαίσιο, έπειτα από σύντομη αναφορά των εφαρμογών αυτών στη σύγχρονη εποχή, γίνεται μια ανάλυση της επικοινωνίας και της κατανάλωσης ενέργειας σε τέτοια δίκτυα, όπως και των τεχνικών προκλήσεων και απαιτήσεων τους.

Εν συνεχεία, γίνεται παρουσίαση της πλατφόρμας Tmote Sky της εταιρείας Moteiv, η οποία θα αποτελέσει τη μονάδα που θα χρησιμοποιηθεί για τη δημιουργία του ασύρματου δικτύου της εργασίας. Αυτό περιλαμβάνει την περιγραφή των συστατικών και χαρακτηριστικών, αλλά και τον τρόπο μετατροπής των τιμών που λαμβάνονται σε μορφή σύμφωνη με το διεθνές σύστημα μονάδων SI.

Για τη διαχείριση των πλατφορμών χρησιμοποιήθηκε το λειτουργικό TinyOS που χρησιμοποιεί τη γλώσσα προγραμματισμού NesC, η οποία βασίζεται πάνω στη C. Ιδιαίτερη βαρύτητα δίνεται στο πρωτόκολλο Tymo, δεδομένου ότι θα χρησιμοποιηθεί για την επικοινωνία πολλαπλών κόμβων στο δίκτυο.

Στην εφαρμογή που υλοποιήθηκε διακρίνονται τρία επίπεδα: Μονάδες Tmote Sky, Βάση Δεδομένων και Server. Στο πρώτο επίπεδο αναπτύχθηκε μια εφαρμογή σε nesC για την λήψη μετρήσεων από τις μονάδες του δικτύου και προώθηση τους στη μονάδα-βάση. Στο δεύτερο επίπεδο δημιουργήθηκε μια βάση δεδομένων MySQL σε ένα server για την αποθήκευση των δεδομένων και μια εφαρμογή Java για τη σύνδεση του αισθητήρα-βάση με αυτήν. Στο τελευταίο επίπεδο παρουσιάζονται γραφικά οι μετρήσεις που έχουν γίνει, μαζί με το κινούμενο μέσο όρο, σε κατάλληλη ιστοσελίδα, με τη χρήση PHP, JavaScript, Google Charts Api και Google Maps Api.

Η εργασία κλείνει με παρατηρήσεις επί των αποτελεσμάτων και προτάσεις για περαιτέρω ανάπτυξη της εφαρμογής.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ασύρματα δίκτυα αισθητήρων, Tmote Sky, TinyOS, NesC, δειγματοληψία, Δρομολόγηση Πολλαπλών Κόμβων, φίλτρο κινούμενου μέσου όρου, Java, Βάση Δεδομένων, MySQL, PHP, Google Charts, Google maps Api, JavaScript, Joomla

ABSTRACT

A Wireless Sensor Network (WSN) consists of spatially distributed autonomous sensors to monitor various conditions. Nowadays, such networks are used in many industrial and consumer applications. In computer science and telecommunications, in particular, wireless sensor networks are a highly active research area. The scope of this Diploma Thesis is to develop an application in order to acquire environmental data which are wirelessly transmitted through a WSN and then process, store and eventually display them in a user-friendly way. During this application, temperature, humidity, total solar radiation (TSR), photosynthetically active radiation (PAR) and internal voltage and temperature of the sensor-node are measured.

First of all, the components of a sensor-node are presented. The next logical step is analyzing WSN, starting by WSN applications and then going on to describe communication, energy consumption, technical challenges and requirements.

Subsequently, the components and features of the platform Moteiv Tmote Sky are provided. This is the kind of node that will be used to form the WSN needed. Furthermore, it is shown how to convert prices received in a form consistent with the international system of units SI.

Node management is achieved by using the TinyOS operating system and the programming language NesC, which is based on C. An essential part of this thesis is also the Tymo Protocol, which is used for multi-hop routing.

The developed application consists of three distinct levels: Sensor-node Tmote Sky, Database and Server. The first level is a nesC application that samples the sensors of the WSN and then passes the data through the network to the base station. The second level is composed of a MySQL database located on a server and a Java application that obtains the values needed from the base-station and connects to the database to store them. In the final level, PHP is used to retrieve data along with moving average values from the database, in order to be represented graphically in a site, using JavaScript, the Google Charts Api and the Google Maps Api.

In conclusion, some final observations about the results and suggestions for further development of the application are noted.

KEYWORDS: Wireless Sensor Networks, Tmote Sky, TinyOS, NesC, sampling, Multi-Hop Routing, moving average filter Java, Database, MySQL, PHP, Google Charts, Google maps Api, JavaScript, Joomla

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Φίλιππο Κωνσταντίνου για την ανάθεση της συγκεκριμένης διπλωματικής εργασίας και της ευκαιρίας που μου παρείχε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Επίσης θα ήθελα να εκφράσω τις ευχαριστίες μου στον υποψήφιο Διδάκτορα κ. Δημήτρη Παπανικολάου για την αμέριστη βοήθεια που μου παρείχε κατά τη διάρκεια της εκπόνηση της διπλωματικής εργασίας και το λέκτωρα κ. Αθανάσιο Παναγόπουλο που συνέβαλε στην ολοκλήρωση της. Τέλος πρέπει να ευχαριστήσω την οικογένεια μου και τους φίλους μου για ηθική και ουσιαστική υποστήριξη και συμπαράσταση σε όλη τη διάρκεια της φοίτησης μου και ιδιαίτερα το συνάδελφο Βασίλη Σοφρά για τη βοήθεια και τις ορθές παρατηρήσεις του στο παρακάτω κείμενο.

Αικατερίνη Πλευράκη

ΠΕΡΙΕΧΟΜΕΝΑ

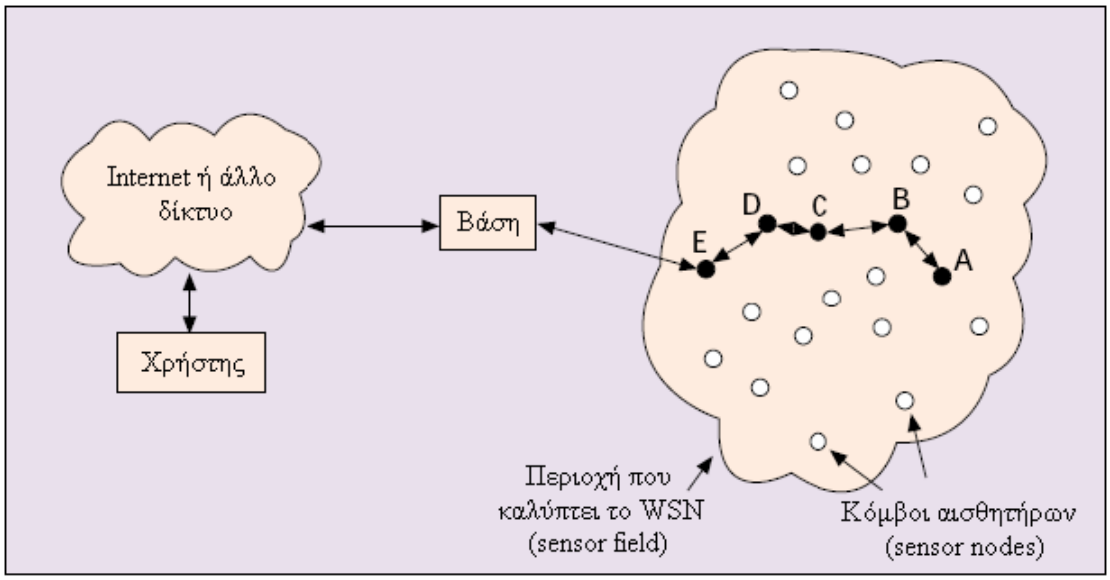
ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. Εισαγωγή

1.1 Εισαγωγή

Η ραγδαία εξέλιξη των ασύρματων τεχνολογιών και η σημαντική αύξηση των ασυρμάτων υπηρεσιών έχουν κάνει τις ασύρματες επικοινωνίες ένα ευρέως διαδεδομένο μέσο για μεταφορά πληροφορίας σε πολλούς τομείς. Επιπρόσθετα, η σύγχρονη τάση για ολοένα μεγαλύτερη ολοκλήρωση και μείωση του μεγέθους έχει δώσει την δυνατότητα για ανάπτυξη πολλών συσκευών χαμηλού κόστους και ισχύος που υλοποιούν ένα πλήθος εφαρμογών. Πιο συγκεκριμένα, η ραγδαία ανάπτυξη των μικροηλεκτρομηχανικών συστημάτων αλλά και της χρήσης των ραδιοσυχνοτήτων (radio frequency - RF) έχει επιτρέψει την ανάπτυξη χαμηλής ισχύος, κατάλληλων για δημιουργία δικτύων και χαμηλού κόστους μικροαισθητήρων (microsensors). Τέτοιοι κόμβοι αισθητήρων είναι ικανοί να συλλέγουν ποικιλία δεδομένων που αφορούν το περιβάλλον γύρω τους, όπως θερμοκρασία, πίεση, κίνηση αντικειμένου κ.α., να τα επεξεργάζονται και ταυτόχρονα να επικοινωνούν με τους γειτονικούς τους κόμβους. Ένα Wireless Sensor Network (εφεξής WSN) αποτελείται από ένα ή περισσότερους σταθμούς βάσης (sink ή base station) και από μερικές δεκάδες ή χιλιάδες κόμβους-αισθητήρες (sensor nodes), που συνδέονται μεταξύ τους με ένα ασύρματο μέσο και αναπτύσσονται σε ένα χώρο είτε μέσα στο φαινόμενο είτε πολύ κοντά σε αυτό. (Εικόνα 1.1). Κάθε κόμβος αισθητήρων συλλέγει δεδομένα τα οποία στη συνέχεια δρομολογούνται μέσω μιας multi-hop επικοινωνίας στη βάση, η οποία επικοινωνεί με τον τελικό χρήστη μέσω του Internet ή άλλου δικτύου.



Εικόνα 1.1 - Ένα ασύρματο δίκτυο αισθητήρων

Μέσα στο πλαίσιο των Ασύρματων Δικτύων Αισθητήρων - WSNs, υπάρχουν πολλές πιθανές δυνατότητες όπου ένα Ασύρματο Δίκτυο Αισθητήρων μπορεί να παραταχθεί για να υποστηρίξει μεγάλο αριθμό εφαρμογών. Η θέση των κόμβων-αισθητήρων δεν χρειάζεται να σχεδιαστεί ή να

προκαθοριστεί. Αυτό επιτρέπει την τυχαία εναπόθεση των κόμβων σε περιοχές μη προσβάσιμες από τον άνθρωπο ή σε επιχειρήσεις αντιμετώπισης καταστροφών. Από την άλλη πλευρά, αυτό σημαίνει ότι τα πρωτόκολλα δικτύου και οι αλγόριθμοι των κόμβων πρέπει να έχουν την ικανότητα να οργανώνονται από μόνα τους. Ένα άλλο χαρακτηριστικό γνώρισμα των ασύρματων δικτύων αισθητήρων είναι η συνεργασία και επομένως η συνεχής επικοινωνία που λαμβάνει χώρα κυρίως μεταξύ των γειτονικών κόμβων. Κάθε κόμβος ενσωματώνει και έναν επεξεργαστή που του δίνει την δυνατότητα, αντί να στείλει κατευθείαν τα δεδομένα σε έναν καθορισμένο κόμβο που έχει αναλάβει την επεξεργασία-μίξη τους, να χρησιμοποιεί ο ίδιος πρώτα τον επεξεργαστή του για την εκτέλεση καθορισμένων απλών υπολογισμών και στη συνέχεια να αποστέλλει μόνο τα απαραίτητα και μερικώς επεξεργασμένα δεδομένα.

Η χρήση συστημάτων ενδιάμεσου λογισμικού (middleware systems) σε παραδοσιακά υπολογιστικά περιβάλλοντα διευκολύνει τη δουλειά του τομέα ανάπτυξης εφαρμογών αφαιρώντας το φορτίο που αφορά στην πολυπλοκότητα των κατανεμημένων συστημάτων. Σύμφωνα με αυτή την προσέγγιση τα WSNs μπορούν να χρησιμοποιούν ένα στρώμα ενδιάμεσου λογισμικού για να υλοποιήσουν συγκεκριμένες λειτουργίες όπως διαχείριση του κόμβου-αισθητήρα, επιλογή κατάλληλου πρωτοκόλλου δικτύου και τη βασικότερη εφαρμογή που αφορά στην παράδοση των δεδομένων που ανιχνεύονται από τον αισθητήρα[1].

Τα ασύρματα δίκτυα αισθητήρων αποτελούν έναν τομέα ο οποίος αναμένεται να βρει σημαντική απήχηση στο άμεσο μέλλον.

1.2 Σκοπός της διπλωματικής

Σκοπός της παρούσας εργασίας είναι η λήψη δεδομένων από το περιβάλλον και το εσωτερικό αισθητήρων που αποτελούν ένα μεγαλύτερο δίκτυο με σκοπό την αποθήκευση, επεξεργασία και απεικόνιση τους. Αναλυτικότερα, κάθε node - μέλος του WSN παίρνει μετρήσεις:

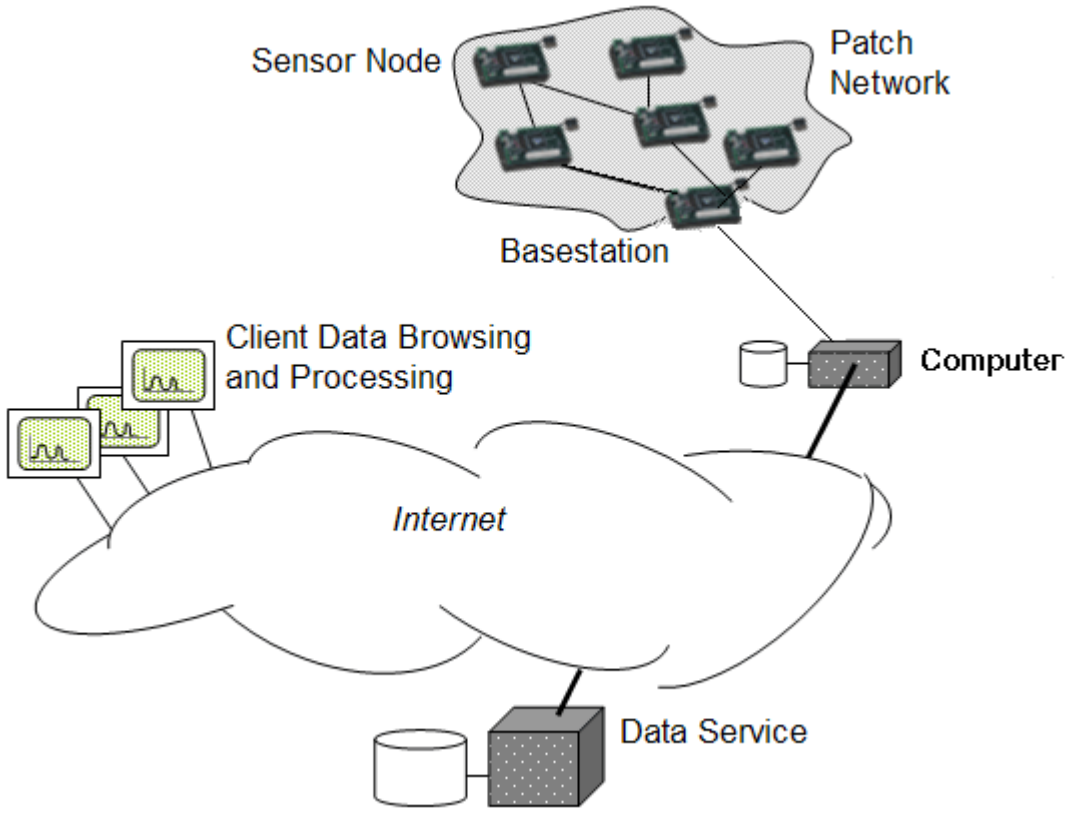
- Θερμοκρασίας
- Υγρασίας
- Ολικής ηλιακής ακτινοβολίας
- Ενεργής φωτοσυνθετικά ακτινοβολίας
- Εσωτερικής θερμοκρασίας
- Εσωτερικής τάσης

Στη συνέχεια, ο στόχος είναι η αποστολή των μετρήσεων στον σταθμό-βάση. Συνεπώς γίνεται ενασχόληση με τη δρομολόγηση πολλαπλών κόμβων (Multi-Hop Routing) στο δίκτυο αισθητήρων. Αρχικά γίνεται απεικόνιση του μη επεξεργασμένων πακέτου δεδομένων που φτάνει στο σταθμό-βάση στην οθόνη του υπολογιστή. Το επόμενο βήμα είναι η μελέτη της οργάνωσης του και τελικά ο διαχωρισμός ωφέλιμου φορτίου (data payload), η μετατροπή των τιμών των μετρήσεων σε φυσικές μονάδες, η απεικόνισή τους στην κονσόλα του λειτουργικού και η εξαγωγή τους σε μια βάση δεδομένων σε κατάλληλο server. Το τελευταίο βήμα της

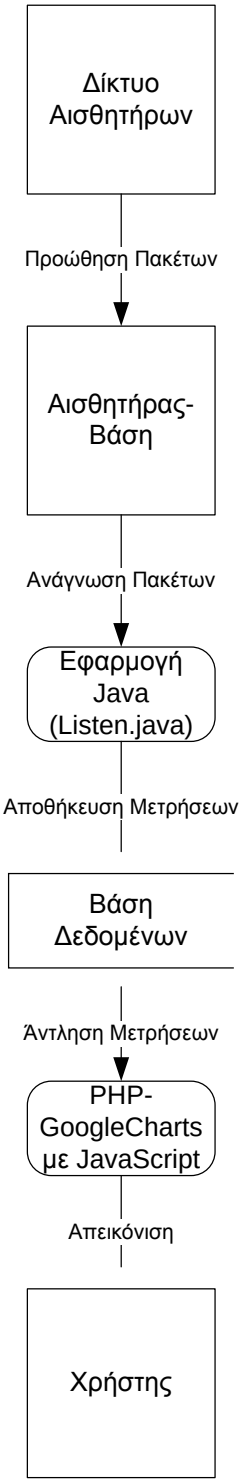
διπλωματικής είναι η γραφική αναπαράσταση της πληροφορίας για πληρέστερη κατανόηση από το χρήστη. Αυτό επιτυγχάνεται με την εξαγωγή των μετρήσεων από τη βάση δεδομένων με κατάλληλη εφαρμογή και την απεικόνιση τους σε διαγράμματα στην web page της εφαρμογής (http://dpapanikolaou.gr/plevraki/) η οποία ανανεώνεται διαρκώς ανά ένα λεπτό έτσι ώστε να απεικονίζει πάντα τα πιο πρόσφατα δεδομένα.

1.3 Περιγραφή συστήματος

Το WSN του συστήματος αποτελείται από μονάδες-αισθητήρες Tmote Sky της εταιρείας Moteiv,οι οποίοι σχηματίζουν ένα δίκτυο το οποίο προωθεί τις μετρήσεις του σε μία μονάδα που βρίσκεται συνδεδεμένη σε υπολογιστή μέσω USB . Ο προγραμματισμός τους πραγματοποιήθηκε μέσω του λειτουργικού συστήματος TinyOS 2.1 και για την υλοποίηση της multi-hop λειτουργίας χρησιμοποιήθηκε το πρωτόκολλο Tymo. Οι μετρήσεις που φτάνουν στο σταθμό-βάση επεξεργάζονται και καταχωρούνται στη βάση δεδομένων μέσω μιας Java εφαρμογής και η βάση έχει δημιουργηθεί με MySQL. Επίσης, η εξαγωγή δεδομένων από τη βάση που βρίσκεται στο server υλοποιείται με PHP, ενώ η απεικόνιση τους γίνεται με Google charts μέσω JavaScript. Στις εικόνες 1.1 και 1.2 φαίνονται η αρχιτεκτονική του συστήματος και το διάγραμμα ροής αντίστοιχα.



Εικόνα 1.1 Αρχιτεκτονική συστήματος



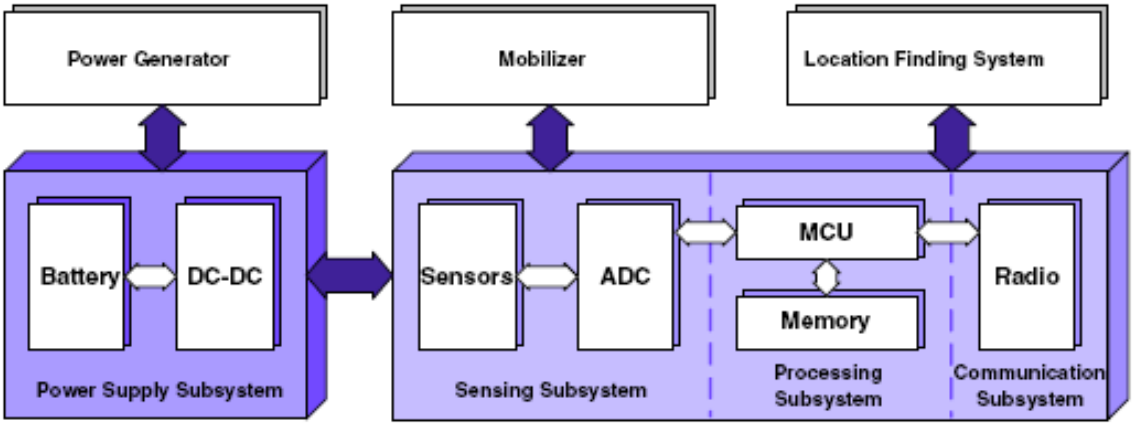
Εικόνα 1.2 Διάγραμμα ροής συστήματος

2. ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ (WIRELESS SENSOR NETWORKS)

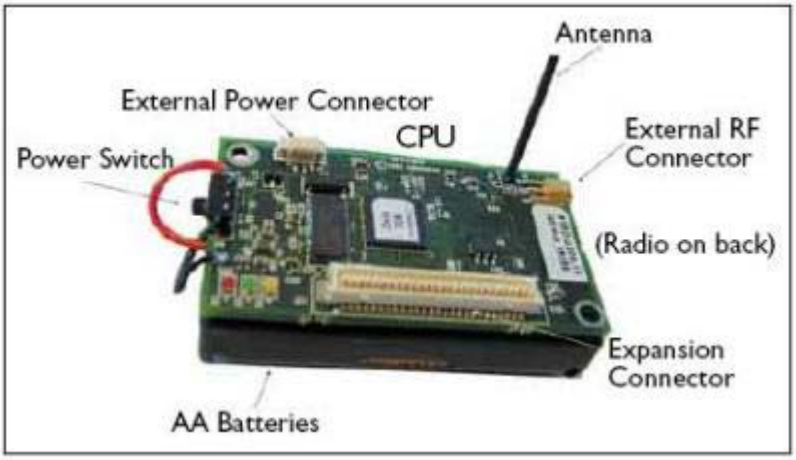
2.1 Κόμβος Αισθητήρας

2.1.1 Συστατικά κόμβου αισθητήρα

Ένας κόμβος αισθητήρα αποτελείται από τέσσερα βασικά συστατικά, το *υποσύστημα αισθητήρων* (sensing subsystem), το *υποσύστημα επεξεργασίας των δεδομένων* (processing subsystem), το *υποσύστημα της ασύρματης επικοινωνίας* (communication subsystem) και τέλος το *υποσύστημα παροχής ισχύος* (power supply subsystem).[2] Στην εικόνα 2.1 μπορούμε να δούμε τα πιο πάνω βασικά υποσυστήματα που συνθέτουν το κόμβο-αισθητήρα και περιγράφονται στη συνέχεια.



Εικόνα 2.1 Αρχιτεκτονική ενός ασύρματου κόμβου αισθητήρα.



Εικόνα 2.2 Κόμβος αισθητήρας

Το *υποσύστημα αισθητήρων* αποτελεί το μέσο με το οποίο ο κόμβος αντιλαμβάνεται το φυσικό κόσμο και σκοπός του είναι η μετατροπή ενός φυσικού ή χημικού μεγέθους σε ηλεκτρικό σήμα, το οποίο συνήθως πρόκειται για μια αναλογική τάση. Αποτελείται, κατά κανόνα, από τρία μέρη: τον ηλεκτρονικό αισθητήρα, τα απαραίτητα ηλεκτρονικά για την προετοιμασία του σήματος και την τελική του μετατροπή σε τάση αν αυτό απαιτείται (Signal-Conditioning), και τέλος τον A/D μετατροπέα (Analog to Digital Converter) για την ψηφιακή μετατροπή του αναλογικού σήματος τάσης. Πολλές φορές, τα τρία αυτά μέρη, είναι δυσδιάκριτα, αφού μπορεί να ενσωματώνονται στο πακέτο του αισθητήρα ή/και του επεξεργαστή, αν πρόκειται για μικροελεγκτή.

Το *υποσύστημα της επεξεργασίας των δεδομένων* αποτελεί την καρδιά του συστήματος και ο ρόλος του είναι να ελέγχει και να εκτελεί τις περισσότερες λειτουργίες του ασύρματου κόμβου. Μερικές από αυτές είναι ο έλεγχος και η δειγματοληψία των αισθητήρων, η επεξεργασία και προσωρινή αποθήκευση των δεδομένων και η λήψη αποφάσεων, η εκτέλεση των τηλεπικοινωνιακών πρωτοκόλλων και των αλγορίθμων δρομολόγησης και εξοικονόμησης ενέργειας. Η υπολογιστική απόδοση του κόμβου και οι ενεργειακές του απαιτήσεις καθορίζονται σε μεγάλο βαθμό από την επεξεργαστική μονάδα, η οποία και αποτελεί τον δεύτερο σε σειρά καταναλωτή ενέργειας του κόμβου, μετά το υποσύστημα ασύρματης επικοινωνίας. Πολλοί επεξεργαστές, μικροελεγκτές, DSPs ακόμα και FPGA είναι διαθέσιμοι στην αγορά και προσφέρονται για χρήση στους κόμβους-αισθητήρες, αλλά συνήθως χρησιμοποιούνται οι μικροελεγκτές. Στους σύγχρονους μικροελεγκτές ενσωματώνονται μνήμες τύπου flash και RAM, A/D μετατροπείς και ψηφιακά I/O σε ένα ολοκληρωμένο κύκλωμα χαμηλού κόστους. Η επιλογή του επεξεργαστή στηρίζεται σε παράγοντες όπως η κατανάλωση ενέργειας, οι απαιτήσεις σε τάση λειτουργίας, το κόστος, η υποστήριξη περιφερειακών, ο χρόνος αφύπνισης και η ταχύτητα του.

Το *υποσύστημα της επικοινωνίας* είναι το πλέον σημαντικό κομμάτι ενός κόμβου μιας και αποτελεί τον μεγαλύτερο καταναλωτή ενέργειας, επηρεάζοντας έτσι την απόδοση του κόμβου αλλά και τη συνολική απόδοση του δικτύου. Θέματα που απασχολούν την έρευνα στον τομέα του υποσυστήματος μετάδοσης αφορούν την ακτίνα εκπομπής, τον τύπο διαμόρφωσης που χρησιμοποιείται καθώς και τον ρυθμό μετάδοσης δεδομένων. Οι περισσότεροι ασύρματοι κόμβοι χρησιμοποιούν πομποδέκτες που λειτουργούν στην ISM (Industrial, Scientific, Medical) μπάντα, στις συχνότητες των 433.5 – 437.9 MHz, 868.0 – 868.6 MHz και 2400 – 2483.5 MHz. Χρησιμοποιούν δημοφιλείς μεθόδους διαμόρφωσης όπως OOK (On/Off Key), ASK (Amplitude Shift Key) και FSK (Frequency Shift Key). Η κατανάλωση ισχύος είναι σχεδόν ίδια κατά την εκπομπή και τη λήψη δεδομένων και κυμαίνεται στα 15 – 300 mW, ανάλογα με το ολοκληρωμένο, τη συχνότητα και τη μέθοδο που χρησιμοποιεί για την επικοινωνία. Η ακτίνα επικοινωνίας κυμαίνεται από τα 25 – 200 m, με το μέγιστο να παρατηρείται σε εξωτερικούς χώρους, με οπτική επαφή. Οι ρυθμοί μετάδοσης που επιτυγχάνονται είναι της τάξης των 10 – 250 Kbps και πρόσφατα έως και 500Kbps[**3**].

Το *υποσύστημα παροχής ισχύος* αποτελείται από μία μπαταρία και συνήθως και από ένα dc-dc converter. Σε αυτό το υποσύστημα εντοπίζεται η μεγάλη αδυναμία των ασύρματων δικτύων αισθητήρων. Από τη στιγμή, που θα εγκατασταθεί ένα WSN σε έναν χώρο, προβλέπεται να λειτουργήσει, για ένα μεγάλο χρονικό διάστημα, χωρίς την ανθρώπινη επίβλεψη. Αυτό σημαίνει, ότι θα πρέπει να διαχειριστεί τους ενεργειακούς του πόρους με τέτοιο τρόπο, ώστε να μεγιστοποιηθεί ο χρόνος ζωής του. Κύρια πηγή ενέργειας των κόμβων είναι οι μπαταρίες, οι οποίες συνήθως δεν είναι επαναφορτιζόμενες, αλλά έχουν προταθεί και μέθοδοι παραγωγής ενέργειας επάνω στον κόμβο και αποθήκευσής της σε επαναφορτιζόμενες μπαταρίες και

υπέρπυκνωτές, με σκοπό την παράταση του χρόνου ζωής του. Η ενεργειακή κατανάλωση είναι ο δείκτης εκείνος που θα καθορίσει την διάρκεια λειτουργίας του δικτύου.

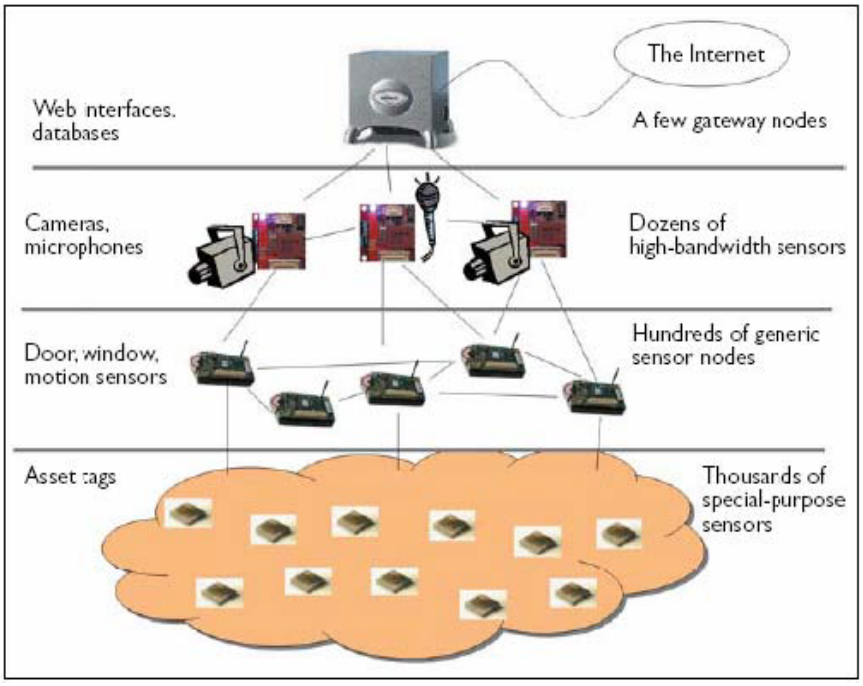
Εκτός από τις προαναφερόμενες βασικές υπομονάδες που συνθέτουν τους κόμβους-αισθητήρες, μπορεί οι κόμβοι να ενσωματώνουν και επιπρόσθετες υπομονάδες όπως γεννήτρια ισχύος (power generator), σύστημα κίνησης (mobilizer) και ένα σύστημα εντοπισμού θέσης (location finding system).

Οι παραπάνω υπομονάδες είναι πιθανό να πρέπει να τοποθετηθούν σε μία μονάδα με μικρό μέγεθος. Το μέγεθος ενός κόμβου, αυτό μπορεί να κυμαίνεται από τις διαστάσεις ενός παπουτσιού μέχρι και το μέγεθος ενός κόκκου σκόνης, παρότι πλήρως λειτουργικοί κόμβοι με αμιγώς μικροσκοπικές διαστάσεις δεν έχουν ακόμα κατασκευαστεί.

Το κόστος των αισθητήρων κόμβων παρουσιάζει και αυτό μεγάλη διακύμανση και εξαρτάται άμεσα από το μέγεθος του δικτύου και την πολυπλοκότητα αυτής καθεαυτής της συσκευής. Οι περιορισμοί στο κόστος και στο μέγεθος των αισθητήρων κόμβων έχουν άμεση επίδραση στον περιορισμό των ενεργειακών τους πόρων, της μνήμης, της υπολογιστικής ισχύος τους και του εύρους ζώνης που χρησιμοποιούν.

2.1.2 Κατηγορίες πλατφορμών

Η εμπειρία από την αρχική τους ανάπτυξη, έδειξε ότι τα συστήματα δικτύων αισθητήρων απαιτούν μια ιεράρχηση των κόμβων, που να ξεκινάει από χαμηλού επιπέδου αισθητήρες και να συνεχίζει σε υψηλού επιπέδου μονάδες με δυνατότητες συλλογής δεδομένων, ανάλυσης και αποθήκευσης (Εικόνα 2.3).



Εικόνα 2.3 Ιεραρχική ανάπτυξη ενός ασύρματου δικτύου αισθητήρων. Κάθε κατηγορία πλατφορμών διαχειρίζεται αίσθηση διαφορετικού τύπου

Αυτή η βαθμωτή αρχιτεκτονική είναι κοινή σε όλα σχεδόν τα δίκτυα αισθητήρων και γίνεται εύκολα κατανοητή με ένα παράδειγμα. Ας θεωρήσουμε ένα δίκτυο αισθητήρων ενός προηγμένου συστήματος ασφαλείας, στο οποίο η πλειονότητα των αισθητήρων καλύπτει σπάσιμο τζαμιών, κλείσιμο επαφών και ανίχνευση κίνησης. Το πλήθος των αισθητήρων και των κατάλληλων θέσεων τους απαιτούν να τροφοδοτούνται από μπαταρία. Συμπληρώνονται από μερικούς περισσότερο εξελιγμένους αισθητήρες, όπως είναι οι κάμερες και οι ανιχνευτές ήχων και χημικών, τοποθετημένοι σε καίρια σημεία. Τα απλά και τα σύνθετα δεδομένα των αισθητήρων δρομολογούνται μαζί, μέσω ενός δικτύου, σε μια μονάδα παρακολούθησης και ελέγχου του κτιρίου, που παρέχει τη δυνατότητα συνεχούς παρακολούθησης. Οι αισθητήρες που είναι τοποθετημένοι σε παράθυρα και πόρτες για ανίχνευση εισβολής είναι παραδείγματα γενικευμένων μονάδων αισθητήρων *(generic sensing devices)*. Η λειτουργία τους είναι απλή και συγκεκριμένη και απαιτεί την τροφοδοσία από μπαταρία μεγάλης διάρκειας. Επιπλέον, οι ρυθμοί επεξεργασίας και επικοινωνίας που διαθέτουν, είναι οι ελάχιστοι. Αντίθετα, οι αισθητήρες ήχου, εικόνας και χημικών είναι παραδείγματα μονάδων μεγάλου εύρους *ζώνης*, που απαιτούν επικοινωνία και μεγαλύτερη υπολογιστική ισχύ. Μπορεί σε κάποιες περιπτώσεις να απαιτούν τροφοδότηση από μπαταρία αλλά συχνά χρειάζεται να συνδεθούν με το δίκτυο παροχής ηλεκτρικής τάσης, για να λειτουργήσουν σε μακρά διάρκεια.

Επιπλέον των παραδοσιακών εφαρμογών ασφαλείας, τα ασύρματα δίκτυα αισθητήρων είναι σχεδιασμένα να παρακολουθούν κινητά αντικείμενα αξίας (mobile assets), μέσω μικροσκοπικών, χαμηλού κόστους συσκευών ασφαλείας (security tags-mini motes). Αυτοί οι *κόμβοι αισθητήρων ειδικού σκοπού* είναι συνώνυμοι μικροσκοπικών διατάξεων με απαίτηση ελάχιστης τροφοδοσίας. Θα μπορούσαν να ενεργοποιήσουν τον συναγερμό όταν ένα αντικείμενο απομακρυνθεί χωρίς εξουσιοδότηση. Επίσης πρέπει να είναι πλήρως ολοκληρωμένοι και σχετικά φτηνοί.

Στα συστήματα ασφαλείας, το δίκτυο αισθητήρων είναι πιθανό να έχει ένα ή περισσότερα τελικά σημεία, που περιλαμβάνουν μια βάση δεδομένων ή άλλο λογισμικό συλλογής δεδομένων, σχεδιασμένο να επεξεργάζεται και να αποθηκεύει ενδείξεις ανεξάρτητων αισθητήρων. Αυτές οι *μονάδες πύλης* (gateway nodes) παρέχουν μια διεπαφή (interface) σε πολλά υπάρχοντα είδη δικτύων.

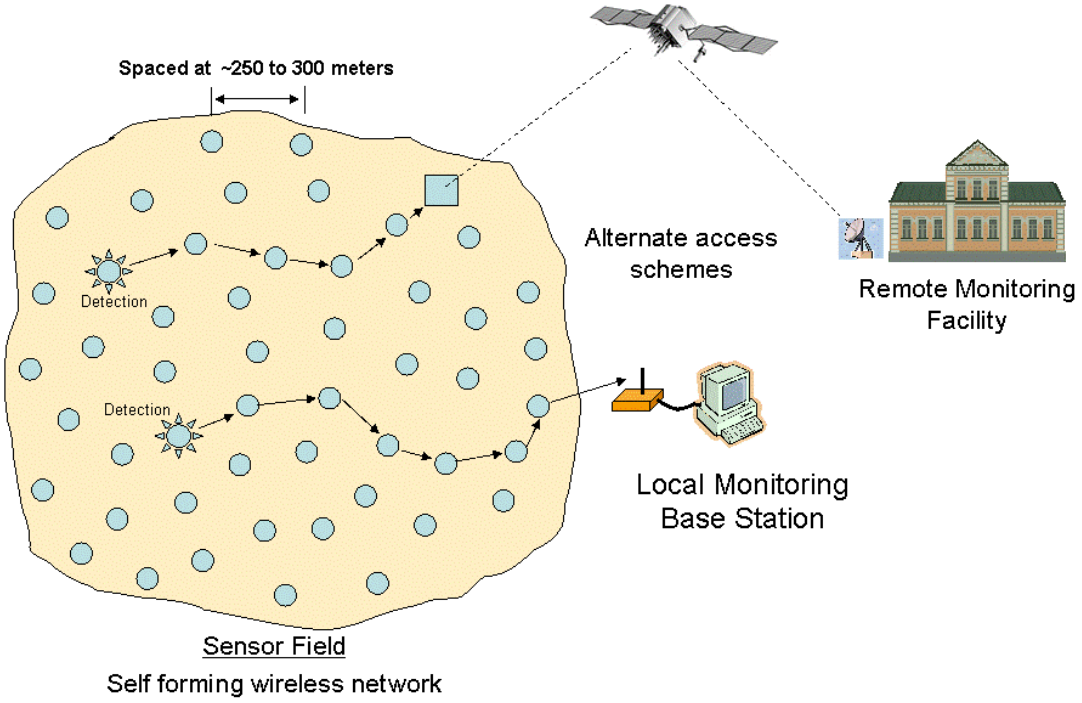
Στον Πίνακα 2.1 παρατίθενται τα τυπικά χαρακτηριστικά λειτουργίας των τεσσάρων κατηγοριών των μονάδων-κόμβων: πλατφόρμα-αισθητήρας **ειδικού σκοπού (specialized sensing platform)**, πλατφόρμα-αισθητήρας **γενικού σκοπού (generic sensing platform),** πλατφόρμα-αισθητήρας **μεγάλου εύρους ζώνης (high bandwidth sensing)** και **πύλη (gateway)** -όλες κατασκευασμένες με τεχνολογία αιχμής.

Πίνακας 2-1 Τυπικά χαρακτηριστικά λειτουργίας των τεσσάρων κατηγοριών κόμβων

Node Type	Sample "Name" and Size	Typical Application Sensors	Radio Bandwidth (Kbps)	MIPS Flash RAM	Typical Active Energy (mW)	Typical Sleep Energy (uW)	Typical Duty Cycle (%)
Specialized sensing platform	Spec mm^3	Specialized low-bandwidth sensor or advanced RF tag	<50Kbps	<5 <0.1Mb <4Kb	1.8V*10–15mA	1.8V *1uA	0.1–0.5%
Generic sensing platform	Mote 1-10cm^3	General-purpose sensing and communications relay	<100Kbps	<10 <0.5Mb <10Kb	3V*10–15mA	3V *10uA	1–2%
High-bandwidth sensing	Imote 1-10cm^3	High-bandwidth sensing (video, acoustic, and vibration)	~500Kbps	<50 <10Mb <128Kb	3V*60mA	3V *100uA	5–10%
Gateway	Stargate >10cm^3	High-bandwidth sensing and communications aggregation Gateway node	>500Kbs–10 Mbps	<100 <32Mb <512Kb	3V*200mA	3V *10mA	>50%

2.2 Ασύρματα Δίκτυα Αισθητήρων

2.2.1 Γενικά



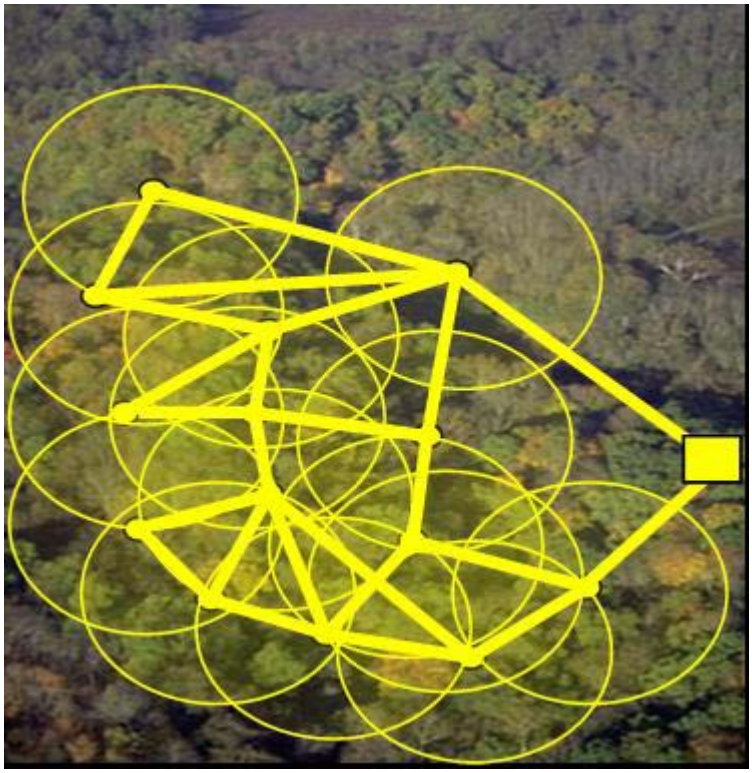
Εικόνα 2.4 Ασύρματο Δίκτυο Αισθητήρων

Ένα δίκτυο αισθητήρων αποτελείται από ένα μεγάλο αριθμό μονάδων-αισθητήρων κατάλληλα τοποθετημένων είτε μέσα στο φαινόμενο παρατήρησης ή πολύ κοντά σε αυτό. Η πυκνότητα των κόμβων στο πεδίο παρατήρησης εξαρτάται άμεσα τόσο από τις απαιτήσεις της εφαρμογής όσο και από αυτές του προς παρατήρηση φαινομένου. Επίσης η θέση αυτή των αισθητήρων μπορεί να μην είναι σχεδιασμένη ή προκαθορισμένη. Αυτό επιτρέπει την 'τυχαία' ανάπτυξη τους σε περιβάλλοντα μη προσπελάσιμα από τον άνθρωπο ή σε επιχειρήσεις αντιμετώπισης καταστροφών. Από την άλλη πλευρά, αυτό προϋποθέτει ότι τα πρωτόκολλα και οι αλγόριθμοι των δικτύων αυτών έχουν την ικανότητα να οργανώνονται από μόνα τους (self-organization).

Ένα ακόμη σημαντικό χαρακτηριστικό των δικτύων αισθητήρων είναι και η συνεχής επικοινωνία μεταξύ των κόμβων-αισθητήρων. Ο επεξεργαστής κάθε κόμβου του δίνει την δυνατότητα, αντί να στείλει κατευθείαν τα δεδομένα σε έναν καθορισμένο κόμβο που έχει αναλάβει τη μίξη τους, να εκτελεί καθορισμένους απλούς υπολογισμούς και στη συνέχεια να αποστέλλει μόνο τα απαραίτητα και μερικώς επεξεργασμένα δεδομένα. Η ενσωμάτωση της δυνατότητας τοπική επεξεργασίας και αποθήκευσης δεδομένων επιτρέπει στις μονάδες να

εκτελέσουν πολύπλοκες λειτουργίες σύμφωνα με την εκάστοτε εφαρμογή που υλοποιούν. Επίσης, η δυνατότητα επικοινωνίας μεταξύ τους επιτρέπει όχι μόνο τη μεταφορά και τον έλεγχο των δεδομένων κατά μήκος του δικτύου, αλλά και τη συνεργασία μεταξύ των μονάδων-κόμβων προς επίτευξη πολύπλοκων αλγορίθμων και εργασιών, όπως είναι η συνάθροιση δεδομένων (data aggregation), η στατιστική δειγματοληψία και η παρακολούθηση της κατάστασης και της υγείας ενός συστήματος.

Ο τελικός προορισμός των μετρήσεων στα δίκτυα αισθητήρων είναι συνήθως ένα κέντρο συλλογής (base-station) με μεγάλα ενεργειακά αποθέματα και με δυνατότητα περαιτέρω επεξεργασίας των δεδομένων με απώτερο σκοπό την λήψη αποφάσεων. Η σταδιακή προώθηση προτιμάται σε σχέση με την απευθείας αποστολή διότι είναι ενεργειακά πιο αποδοτική. Τα μονοπάτια πολλαπλών βημάτων που δημιουργούνται μέσα στο δίκτυο καταλήγουν στον ίδιο προορισμό, με αποτέλεσμα να διαμορφώνεται μια δενδρική αρχιτεκτονική στο δίκτυο όπως και φαίνεται στο παρακάτω σχήμα.



Εικόνα 2.5 Η ανάπτυξη ενός ασύρματου δικτύου αισθητήρων σε δάσος

Το κέντρο συλλογής με την σειρά του επικοινωνεί με τον κόμβο διαχείρισης της εφαρμογής μέσω διαδικτύου ή δορυφόρου. Αξίζει επίσης να σημειωθεί πως τα μονοπάτια μέσα στο δίκτυο δεν είναι προκαθορισμένα και μπορεί να μεταβάλλονται ανάλογα με τις συνθήκες παρατήρησης ή τα ενεργειακά αποθέματα του κάθε κόμβου.

Τα θέματα που σχετίζονται με την διατήρηση και την αλλαγή της τοπολογίας μπορούν να χωριστούν σε τρεις φάσεις.

- *Φάση πριν την τοποθέτηση και φάση τοποθέτησης των κόμβων:* Οι κόμβοι μπορούν να ριφθούν ομαδικά ή να τοποθετηθούν ένας-ένας μέσα στο πεδίο παρατήρησης. Μπορούν να τοποθετηθούν με ρίψη από αεροπλάνο, να διανεμηθούν μέσα σε στρατιωτικά βλήματα ή και να τοποθετηθούν ένας-ένας από ανθρώπους ή ρομπότ.

- *Φάση μετά την τοποθέτηση:* Μετά την τοποθέτηση, οι αλλαγές στην τοπολογία μπορούν να οφείλονται σε λόγους όπως: αλλαγή της θέσης των αισθητήρων, μεταβολές στην ακτίνα μετάδοσης τους (λόγω μεγάλης κίνησης, θορύβου, κινούμενων εμποδίων κ.τ.λ.), διαθέσιμη ενέργεια, και άλλες περιπτώσεις δυσλειτουργίας.

- *Φάση τοποθέτησης επιπρόσθετων κόμβων:* Πρόσθετοι κόμβοι μπορούν να τοποθετηθούν οποιαδήποτε χρονική στιγμή για να αντικαταστήσουν τους κόμβους που δυσλειτουργούν ή εξαιτίας αλλαγών στη δυναμική του έργου.

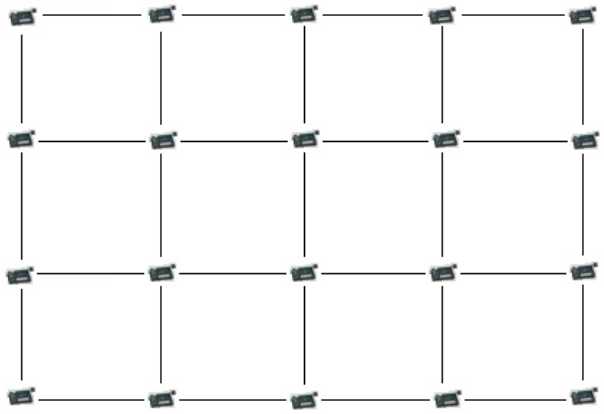
2.2.2 Αρχιτεκτονική Δικτύου

Η παράταξη (deployment) των κόμβων αισθητήρων αποτελεί θεμελιώδες ζήτημα για τα WSN. Ο στόχος των διαφόρων σχεδιασμών που έχουν αναπτυχθεί, αφορά την επίτευξη μιας επιθυμητής κάλυψης με έναν ελάχιστο αριθμό από κόμβους αισθητήρων, ενώ παράλληλα πρέπει να ικανοποιούνται οι προϋποθέσεις που εξασφαλίζουν αποδεκτά επίπεδα ποιότητας, κόστους, αξιοπιστίας και βάθμωσης (επέκτασης όσον αφορά τον αριθμό των κόμβων) μιας εφαρμογής.

Στα WSN, η κάλυψη έχει διπλή έννοια: *χωρική* και κάλυψη *εμβέλειας*. Η κάλυψη εμβέλειας αναφέρεται στο γεωμετρικό χώρο όπου είναι δυνατή η συλλογή δεδομένων («αίσθηση») που αφορούν παραμέτρους του περιβάλλοντος στο οποίο αναπτύσσεται το WSN, ενώ η χωρική κάλυψη αναφέρεται στις σχετικές θέσεις και αποστάσεις των κόμβων αισθητήρων έτσι ώστε να προσφέρουν ακριβείς μετρήσεις-δεδομένα.

Ανάλογα με την εφαρμογή, τώρα, οι στρατηγικές παράταξης των κόμβων αισθητήρων διαφοροποιούνται σημαντικά σε κάθε περίπτωση. Γενικά, οι στρατηγικές αυτές χωρίζονται σε τέσσερις κατηγορίες [23, 24]:

- *Προκαθορισμένη-τυποποιημένη παράταξη* : Συνήθως εφαρμόζεται σε περίπτωση που χρησιμοποιούνται πλέγματα κόμβων (*Σχήμα 2.6*) καθώς και σε περίπτωση που είναι γνωστή εκ των προτέρων η συμπεριφορά του περιβάλλοντος στο οποίο πρόκειται να αναπτυχθεί ένα WSN. Πλεονέκτημά της αποτελεί το γεγονός ότι μπορεί να παρέχει βέλτιστη λύση ως προς την επιθυμητή κάλυψη και ταυτόχρονα να επιτυγχάνει υψηλή ποιότητα και αποδοτικότητα ως προς το κόστος. Ωστόσο, όσον αφορά τη δεύτερη περίπτωση, η ακριβής γνώση της πλήρης συμπεριφοράς ενός περιβάλλοντος είναι αδύνατη, γι' αυτό και η στρατηγική της προκαθορισμένης παράταξης εφαρμόζεται κατά τμήματα.



Εικόνα 2.6 Πλέγμα κόμβων αισθητήρων.

- *Αυτορυθμιζόμενη παράταξη* : Η στρατηγική αυτή αναπτύχθηκε με σκοπό να ξεπεραστούν τα προβλήματα που αντιμετωπίζει η στρατηγική της προκαθορισμένης παράταξης των κόμβων. Συγκεκριμένα, οι κόμβοι αισθητήρων τοποθετούνται στο περιβάλλον κατά τέτοιο τρόπο έτσι ώστε να παρέχουν πλήρη κάλυψη αντιμετωπίζοντας περιπτώσεις ύπαρξης εμποδίων τόσο στην επικοινωνία όσο και στην «αίσθηση» με την αυτό-ρύθμιση κατάλληλων παραμέτρων τους (προσαρμογή). Επίσης, η στρατηγική αυτή είναι επεκτάσιμη ως προς την αύξηση του αριθμού των κόμβων, αλλά η ποσότητα των υπολογισμών μπορεί να γίνει απαγορευτική.

- *Τυχαία παράταξη* : Η συγκεκριμένη στρατηγική αποτελεί μια περισσότερο ρεαλιστική προσέγγιση για μεγάλης κλίμακας εφαρμογές WSN, όπως αυτές που αναπτύσσονται σε πεδία μάχης κ.α.. Οι κόμβοι αισθητήρων «σκορπίζονται» ομοιόμορφα σε μια περιοχή, χωρίς να έχουν προκαθορισμένες θέσεις [17, 21, 10, 11]. Αυτό το χαρακτηριστικό κάνει τη στρατηγική αυτή να είναι ιδιαίτερα επιθυμητή σε πολλές κοινές περιπτώσεις ανάπτυξης κάποιου WSN. Ωστόσο, η κάλυψη που παρέχεται μπορεί να περιορίζεται από τυχόν εμπόδια ή άλλες πηγές θορύβου ή μπορεί να απαιτείται μεγαλύτερη πυκνότητα κόμβων σε ορισμένες περιοχές αντί μιας ομοιόμορφης κατανομής, περιπτώσεις οι οποίες έχουν ως συνέπεια επίσης μείωση της ακρίβειας στην περιγραφή του περιβάλλοντος του δικτύου.

- *Προσαρμοσμένη παράταξη* : Η συγκεκριμένη στρατηγική εφαρμόζεται στις περιπτώσεις αυτές που μια ομοιόμορφη κατανομή των κόμβων δεν είναι ικανοποιητική ως προς την κάλυψη και πλήρη περιγραφή μιας περιοχής-περιβάλλοντος (τυχαία παράταξη), αλλά απαιτείται σε συγκεκριμένες περιοχές μεγαλύτερη ποσότητα πληροφορίας, ενώ αλλού πολύ μικρότερη [12]. Στο γεγονός αυτό συμβάλλει και το ότι η πληροφορία μερικές φορές που παρέχεται από κάποιον κόμβο μπορεί να είναι ελλιπής ή λανθασμένη, οπότε και απαιτείται η κάλυψή του από πληροφορίες γειτονικών κόμβων.

2.3 Εφαρμογές των ασύρματων δικτύων αισθητήρων

Τα WSN έχουν τη δυνατότητα να συλλέγουν πληροφορίες που αφορούν ένα ευρύ φάσμα φυσικών συνθηκών-φαινομένων, όπως [**4**]:

- Θερμοκρασία,
- Φως,
- Πίεση,
- Επίπεδο θορύβου,
- Υγρασία,
- Παρουσία ή κίνηση κάποιου αντικειμένου,
- Σύνθεση του εδάφους,
- Γενικά χαρακτηριστικά κάποιου αντικειμένου, όπως μάζα, διαστάσεις, ταχύτητα κίνησης, θέση κ.α.

Λόγω της αξιοπιστίας, της αυτοδυναμίας, της ευελιξίας και της ευκολίας της επέκτασης των WSN, υπάρχει μια μεγάλη ποικιλία από εφαρμογές. Σε αυτό συμβάλλει ιδιαίτερα και το ιδιαίτερο χαρακτηριστικό τους, ότι μπορούν εφαρμοστούν σε περιβάλλον με σχεδόν οποιεσδήποτε συνθήκες, όπως σε πεδία μάχης, σε ωκεανούς κ.α. Για το λόγο αυτό, πολλές φορές κρίνεται αναγκαία η χρήση προστατευτικών καλυμμάτων, όπως αυτά στην Εικόνα 2.7.

Εικόνα 2.7 Προστατευτικά καλύμματα κόμβων

Οι κόμβοι-αισθητήρες μπορούν να χρησιμοποιηθούν για τη διαρκή ανίχνευση κατάστασης και εντοπισμό συμβάντων καθώς και τον εντοπισμό θέσης. Η ιδέα της μικρό-αίσθησης και ασύρματης σύνδεσης αυτών των κόμβων υπόσχονται ένα πλήθος πεδίων εφαρμογών, μερικές από τις οποίες παραθέτονται παρακάτω:

2.3.1 Εφαρμογές ασφαλείας(Security Applications)

Τα ασύρματα δίκτυα αισθητήρων μπορούν να χρησιμοποιηθούν για ασφάλεια υποδομής και εφαρμογές αντιτρομοκρατίας. Κτίρια και υπηρεσίες όπως εργοστάσια παραγωγής ισχύος και κέντρα επικοινωνιών πρέπει να προστατεύονται από πιθανούς τρομοκράτες. Ολοκληρωμένα δίκτυα οπτικοακουστικών και άλλων αισθητήρων μπορούν να παραταχθούν γύρω από τέτοιες εγκαταστάσεις. Αυτοί οι αισθητήρες εγγυώνται έγκαιρη ανίχνευση επερχόμενου προβλήματος. Συγχωνεύοντας τα δεδομένα πολλαπλών αισθητήρων μπορεί να επιτευχθεί βελτιωμένη κάλυψη ανίχνευση, καθώς και μειωμένο σφάλμα ρυθμού ειδοποίησης. Τα ασύρματα δίκτυα αισθητήρων μπορούν επίσης να χρησιμοποιηθούν για να ανιχνευτούν βιολογικές, χημικές και πυρηνικές επιθέσεις και σε στρατιωτικές εφαρμογές. [5]

Εικόνα 2.8 Το στρατιωτικό σύστημα υποβοήθησης στρατιωτών SaS

Εικόνα 2.9 Στιγμιότυπο από το στρατιωτικό σύστημα υποβοήθησης SAS

2.3.2 Βιομηχανικός έλεγχος(Industrial Control)

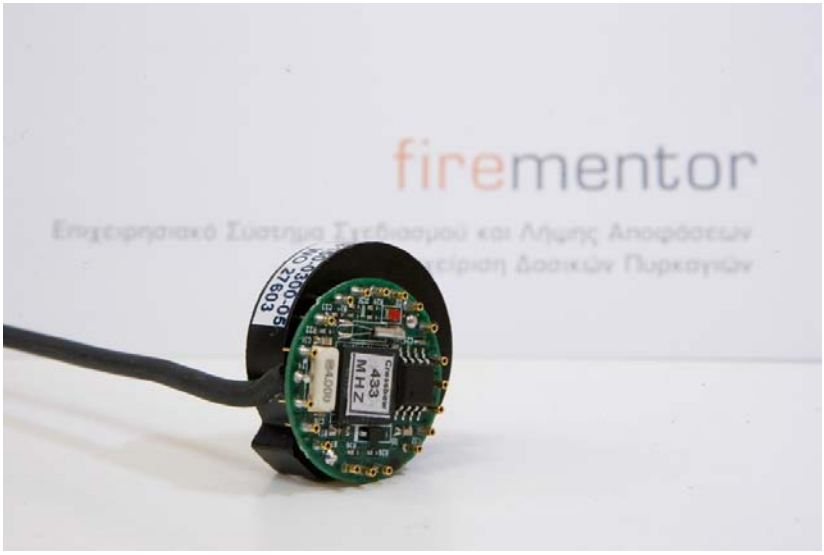
Η βιομηχανία έχει επιδείξει ενδιαφέρον στην ανίχνευση σαν μέσο μείωσης κόστους, βελτίωσης μηχανημάτων, απόδοσης χρήστη και συντηρησιμότητας. Σήμερα, είναι δυνατό να ελεγχθεί η κατάσταση μιας μηχανής μέσα από προσδιορισμό της δόνησης ή τα επίπεδα λίπανσης. Οι αισθητήρες μπορούν να εισαχθούν σε περιοχές μη ευπρόσιτες από άνθρωπο. Οι απομακρυσμένοι ασύρματοι αισθητήρες μπορεί να επιτρέψουν σε ένα εργοστάσιο να εξοπλιστεί, αφού εγγυώνται ότι τηρούνται οι νόμοι και οδηγίες ασφαλείας ενώ το κόστος της εγκατάστασης διατηρείται χαμηλό. Σε ένα βιομηχανικό περιβάλλον χρησιμοποιούνται συχνά αισθητήρες φάσματος. Οπτικοί αισθητήρες μπορούν να αντικαταστήσουν τα ήδη υπάρχοντα όργανα και να επιτελέσουν μετρήσεις στην υλική ιδιοκτησία. Η οπτική ανίχνευση διευκολύνεται και με τη σμίκρυνση. Ο σκοπός αυτής και άλλων βιομηχανικών εφαρμογών είναι να καταστήσουν εφικτή την πολύ-σημειακή ή ανίχνευση σε πλέγμα: δεδομένα από χιλιάδες αισθητήρες τροφοδοτούν βάσεις δεδομένων που μπορούν να ερωτηθούν με οποιοδήποτε τρόπο ώστε να δείξουν εμφανίσουν πληροφορία πραγματικού χρόνου σε μικρή ή μεγάλη κλίμακα.

Εικόνα 2.10 Sensor-based Real-Time Control with Industrial Robots from Mitsubishi Electric[6]

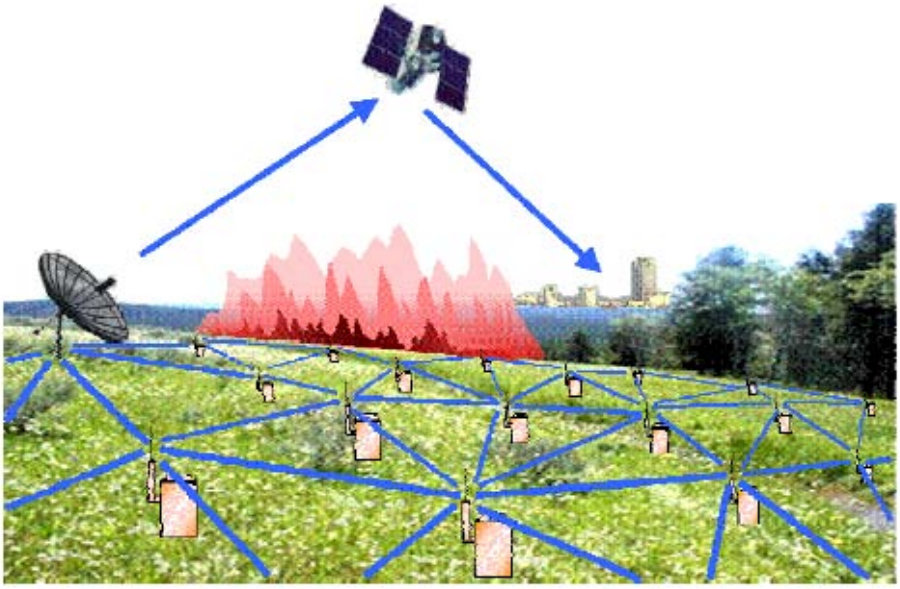
Εικόνα 2.11 Ultrasonic Level Sensors for Non-Contact Measurement [7]

2.3.3 Παρακολούθηση περιβαλλοντικών συνθηκών(Environmental Monitoring)

Οι αισθητήρες περιβάλλοντος μπορούν να χρησιμοποιηθούν για να μελετηθεί η απόκριση βλάστησης στις κλιματικές τάσεις και ασθένειες, και οι οπτικοακουστικοί αισθητήρες μπορούν να προσδιορίσουν, να αποτυπώσουν και να μετρήσουν τον πληθυσμό ζώων, για παράδειγμα πουλιών ή ειδών υπό εξαφάνιση. Επίσης σημαντικές εφαρμογές είναι και η πρόληψη και ανίχνευση εκδήλωσης φωτιάς σε υπαίθριους ή κλειστούς χώρους ή άλλες φυσικές καταστροφές και φαινόμενα και η παρακολούθηση της εξέλιξης γεωργικών καλλιεργειών. Ακόμα, δυνατή είναι και η καταγραφή του μικροκλίματος σε εργασιακούς χώρους και άλλες μεγάλες εγκαταστάσεις για τη βελτιστοποίηση της χρήσης των κλιματιστικών συστημάτων.



Εικόνα 2.12 Ασύρματος αισθητήρας τύπου crossbow για την εφαρμογή firementor

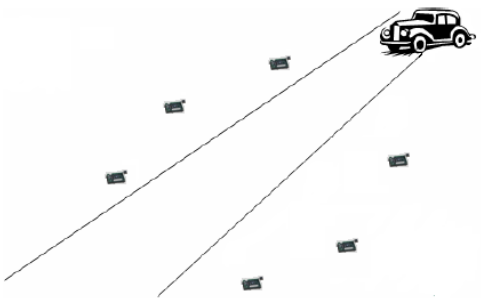


Εικόνα 2.13 Δίκτυο αισθητήρων για τη ανίχνευση πυρκαγιών σε υπαίθριο χώρο.

Εικόνα 2.14 Ασύρματο δίκτυο αισθητήρων για τη παρακολούθηση της γεωργικής καλλιέργειας.

2.3.4 Έλεγχος οδικής κυκλοφορίας(Traffic Control)

Τα ασύρματα δίκτυα αισθητήρων χρησιμοποιούνται σήμερα για την παρακολούθηση και τον έλεγχο της οδικής κυκλοφορίας των οχημάτων. Οι αισθητήρες αναλαμβάνουν να εντοπίσουν τα οχήματα και να ελέγξουν τα φανάρια. Βιντεοκάμερες χρησιμοποιούνται συχνά για την παρακολούθηση τμημάτων δρόμων με έντονη συμφόρηση, και τα βίντεο αποστέλλονται σε κεντρικές τοποθεσίες. Ωστόσο, αυτοί οι αισθητήρες και το δίκτυο που τους συνδέει είναι δαπανηρά, έτσι ο έλεγχος της κυκλοφοριακής κίνησης περιορίζεται σε μερικά κρίσιμα σημεία. Ολιγοδάπανα ασύρματα ad-hoc δίκτυα θα αλλάξουν ριζικά το σενάριο στον έλεγχο της οδικής κυκλοφορίας. Οικονομικοί αισθητήρες με ενσωματωμένη δικτυακή ικανότητα μπορούν να παραταχθούν σε οποιαδήποτε διασταύρωση για να ανιχνεύσουν και να μετρήσουν την συμφόρηση και να υπολογίσουν την ταχύτητα των οχημάτων. Οι αισθητήρες θα συνεργάζονται με γειτονικούς ώστε να σχηματίσουν μία καθολική εικόνα κυκλοφορίας των οχημάτων . Μία διαφορετική και πιο δραστική επανάσταση είναι οι αισθητήρες οι οποίοι προσαρμόζονται πάνω στα ίδια τα οχήματα. Καθώς τα οχήματα περνάει το ένα το άλλο ανταλλάσσουν περιληπτικές πληροφορίες σχετικές με την τοποθεσία και την ταχύτητα και την πυκνότητα της κίνησης, πληροφορίες που μπορεί να δημιουργούνται από αισθητήρες εδάφους. Αυτή η πληροφορία διαδίδεται από όχημα σε όχημα και χρησιμοποιείται από τους οδηγούς για να αποφεύγουν κυκλοφοριακή συμφόρηση και να οργανώνουν εναλλακτικές διαδρομές.



Εικόνα 2.15 Ιχνηλάτηση αντικειμένου.

2.3.5 Δομικά συστήματα παρακολούθησης υγείας (Structural Health Monitoring)

Τα δομικά συστήματα παρακολούθησης υγείας επιδιώκουν να ανιχνεύσουν και να εντοπίσουν τη ζημία στα κτίρια, τις γέφυρες, τα σκάφη, και τα αεροσκάφη. Το σχέδιο τέτοιων συστημάτων είναι ένας ενεργός και καθιερωμένος τομέας της έρευνας. Όταν χτίστηκαν, τέτοια συστήματα θα συμπέραιναν την ύπαρξη και τη θέση της ζημίας με τη μέτρηση της δομικής απάντησης στην περιβαλλοντική ή αναγκασμένη διέγερση. Τα ασύρματα δίκτυα αισθητήρων είναι φυσικός υποψήφιος για τα δομικά συστήματα παρακολούθησης υγείας, δεδομένου ότι επιτρέπουν πυκνή ανίχνευση και απλοποιούν την ανάπτυξη εγκατάστασης οργάνων. Τα ασύρματα δίκτυα διαδραματίζουν ένα αμεσότερο ρόλο στο δομικό έλεγχο. Οι πρόοδοι στη δομική εφαρμοσμένη

μηχανική εξαρτώνται από τη διαθεσιμότητα πολλών λεπτομερών συνόλων στοιχείων που καταγράφουν την απάντηση των διαφορετικών δομών στην περιβαλλοντική δόνηση.

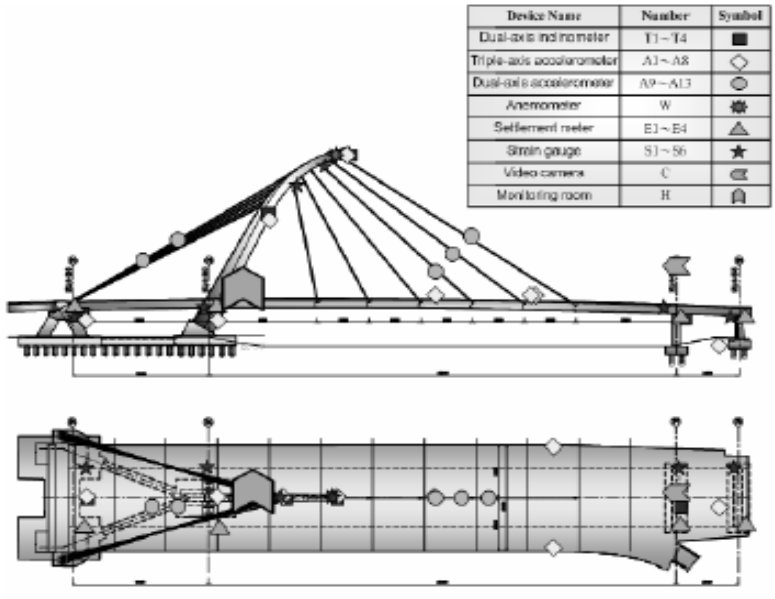


Figure 3. The layout diagram of sensors

Εικόνα 2.16 WSN για την παρακολούθηση καταστροφικών ζημιών σε γέφυρες

2.3.6 Εφαρμογές για έξυπνα σπίτια (Smart House). (Εικόνες 2.17 και 2.19)

Ένα έξυπνο σπίτι παρέχει στους κατοίκους του εξοικονόμηση ενέργειας, έλεγχο και ευκολία με τη χρήση ενός δικτύου ασύρματων αισθητήρων συνδεδεμένους σε ένα κεντρικό υπολογιστή και σύνδεση στο Διαδίκτυο. Οι δυνατότητες που παρέχει είναι:

-Αυτόματη ενεργοποίηση και απενεργοποίηση του φωτισμού σε χώρους που υπάρχει δραστηριότητα.

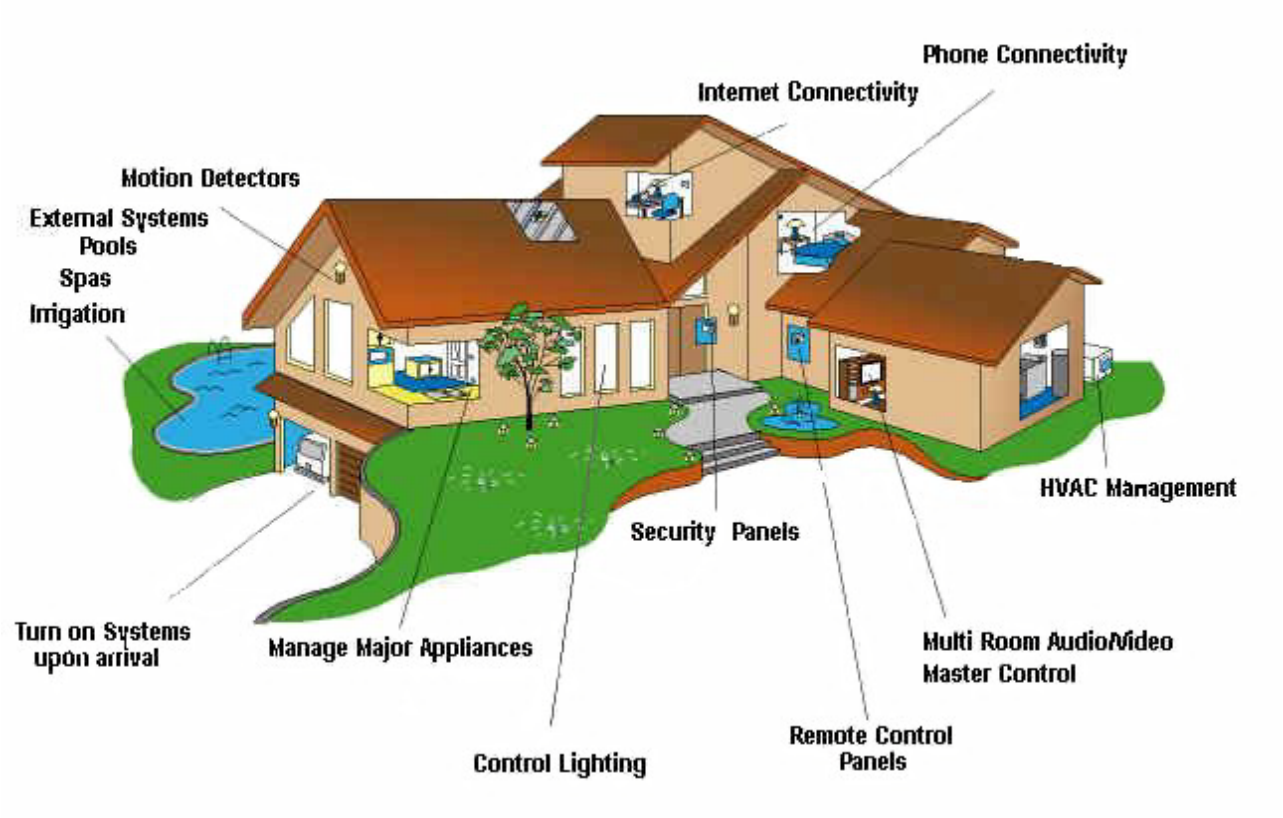
-Αυτόματη ρύθμιση της θερμοκρασίας ή της έντασης του φωτισμού ανάλογα με τις εξωτερικές κλιματολογικές συνθήκες.

-Απενεργοποίηση όλων των συσκευών με το πάτημα ενός κουμπιού.

-Απομακρυσμένο έλεγχο συσκευών και μηχανισμών.

-Προστασία από ηλεκτροπληξία και πλημμύρα.

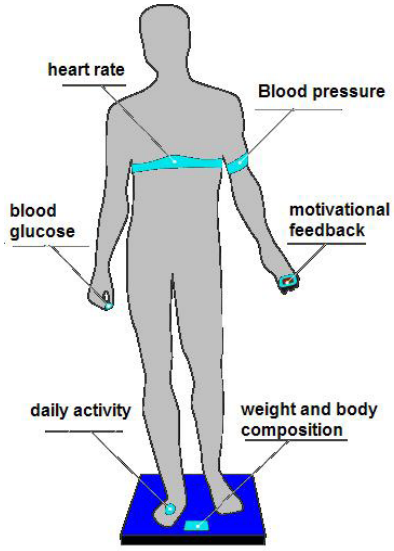
-Ειδοποίηση με γραπτό μήνυμα σε περίπτωση που ανοίξει κάποια πόρτα ή παράθυρο.



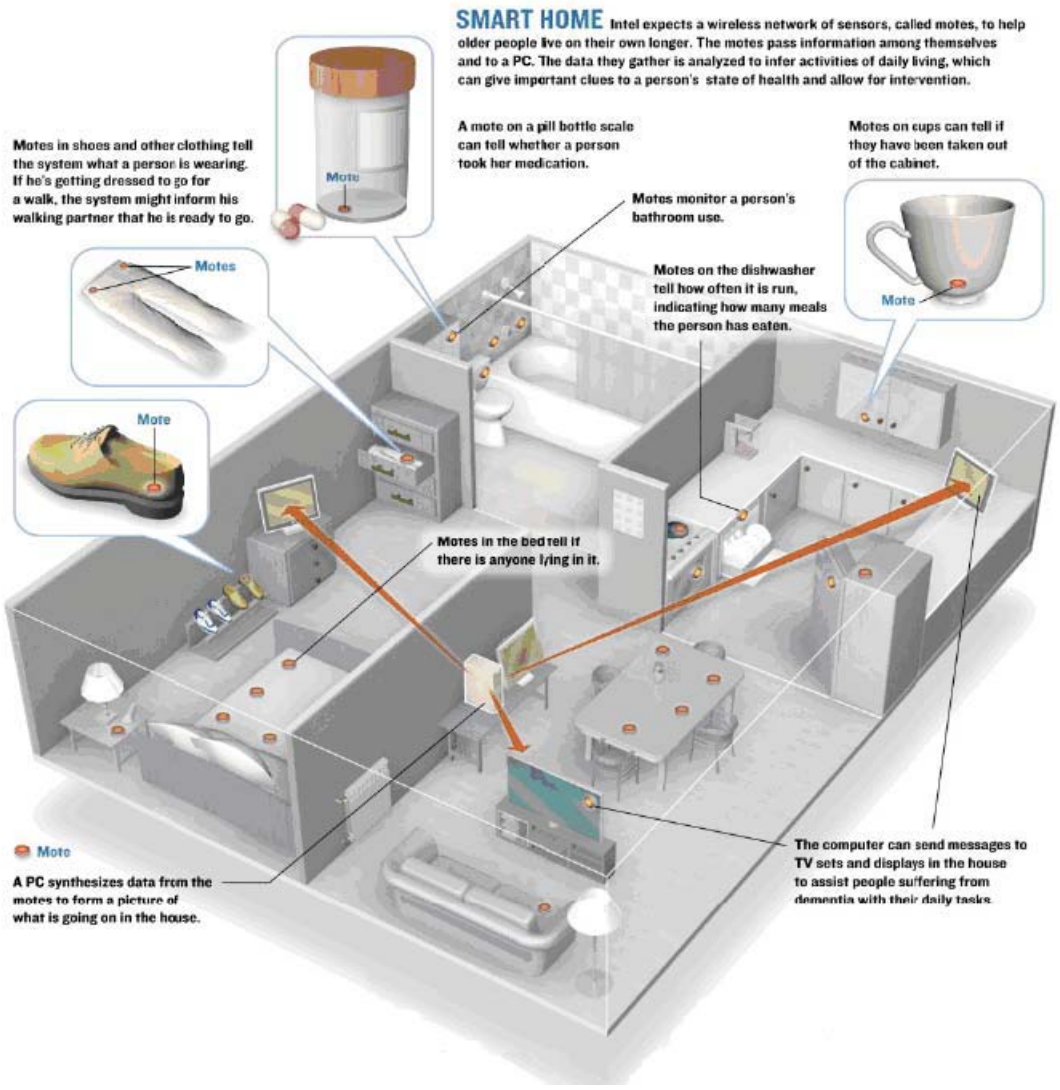
Εικόνα 2.17 Έξυπνο σπίτι [8]

2.3.7 Εφαρμογές στον χώρο της υγείας.[9]

Μια αναδυόμενη εφαρμογή των ασύρματων δικτύων αισθητήρων αφορά την ιατρική περίθαλψη. Ο εφοδιασμός κάθε ασθενή σε ένα νοσοκομείο ή κλινική με μικρούς, φορητούς αισθητήρες για καταγραφή βασικών ζωτικών παραμέτρων θα μπορούσε να δώσει τη δυνατότητα σε γιατρούς και νοσηλευτικό προσωπικό να παρακολουθούν συνεχώς την κατάσταση του ασθενή από μακριά και να επεμβαίνουν εγκαίρως σε περίπτωση όπου αυτό απαιτείται. Τέτοια συστήματα που ήδη χρησιμοποιούνται είναι τα συστήματα καταγραφής κρίσιμων βιοσημάτων: το ηλεκτροκαρδιογράφημα, το ηλεκτρομυογράφημα, το ηλεκτροεγκεφαλογράφημα, ο κορεσμός του οξυγόνου στο αίμα, η θερμοκρασία του ασθενούς, η αναπνοή, η πίεση κ.α. Η παρακολούθηση, μέσω ασύρματων αισθητήρων, καθημερινών δραστηριοτήτων του ανθρώπου, όπως είναι ο τρόπος που ντύνεται, που μαγειρεύει ή οδηγεί το αυτοκίνητό του, καταλήγει σε μια ανιαρή καταγραφή δεδομένων που όμως, μπορούν να οδηγήσουν στη διάγνωση νευρικών διαταραχών, όπως το Parkinson ή το Alzheimer, και την έγκαιρη υποβολή σε θεραπεία. Όμοια, με τη χρήση περιβαλλοντικών αισθητήρων είναι δυνατή η πραγματοποίηση επιδημιολογικών μελετών. Επίσης, ένα σύστημα αισθητήρων μπορεί να βοηθήσει τους ασθενείς να ακολουθήσουν τις οδηγίες των ιατρών τους στη λήψη των φαρμάκων τους. Κάθε χτύπημα στον νεροχύτη και κάθε μπουκάλι στο ψυγείο μπορούν να ελέγχονται μέσω mote για να εγγράφεται το ποσό της δόσης που λήφθηκε από τον ηλικιωμένο, ενώ κάθε μπουκάλι φαρμάκου στηρίζεται σε μια διάταξη συνδεδεμένη με mote.



Εικόνα 2.18 Εφαρμογές στον χώρο της υγείας.



Εικόνα 2.19 Έξυπνο σπίτι-Παρακολούθηση ατόμου μέσα στο σπίτι του με αισθητήρες.

Στις περισσότερες από αυτές τις παραπάνω εφαρμογές υπάρχει μια καθαρή διαφορά μεταξύ των κόμβων που αισθάνονται τα δεδομένα (source) και των κόμβων-βάσεων στους οποίους θα σταλούν τα δεδομένα (base-station). Τα base - stations μερικές φορές είναι μέρος του ίδιου του δικτύου αισθητήρων και κάποιες φορές δεν αποτελούν μέρος του δικτύου, αλλά είναι καθαρά συστήματα έξω από αυτό (π.χ. PDA κ.α.). Στις πλείστες φορές οι κόμβοι αισθητήρων (source nodes) είναι αρκετά πιο πολλοί από τα base-stations. Τα πιο τυπικά σχέδια αλληλεπίδρασης μεταξύ κόμβων αισθητήρων και κόμβων base-stations είναι:

- Event detection: Οι κόμβοι αισθητήρων πρέπει να αναφέρουν στα base-stations μόλις ανιχνεύσουν την πραγματοποίηση ενός συγκεκριμένου γεγονότος.

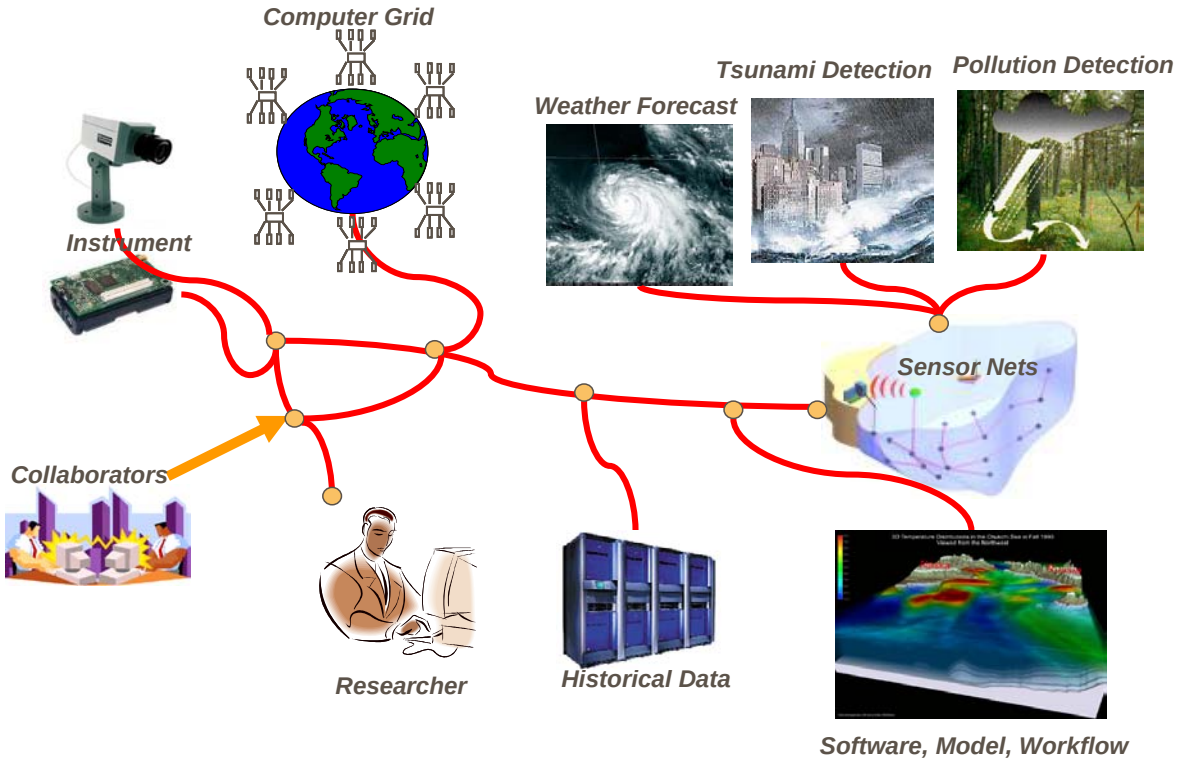
- Periodic measurements: Οι κόμβοι αισθητήρων ρυθμίζονται έτσι ώστε να στέλνουν περιοδικά τις μετρήσεις που ανιχνεύουν στα base-stations.

- Function approximation and edge detection: Ο τρόπος που μια φυσική παράμετρος, όπως η θερμοκρασία, αλλάζει από μια τοποθεσία σε άλλη, μπορεί να θεωρηθεί ανάλογος της τοποθεσίας. Ένα ασύρματο δίκτυο αισθητήρων μπορεί να χρησιμοποιηθεί για να υπολογίσει τη σχέση της τοποθεσίας με τη θερμοκρασία χρησιμοποιώντας περιορισμένο αριθμό δειγμάτων που λαμβάνονται από κάθε μεμονωμένο κόμβο αισθητήρων.

- Tracking: Η πηγή ενός γεγονότος μπορεί να είναι κινητή (π.χ. ένας εισβολέας στα σενάρια επιτήρησης). Ένα ασύρματο δίκτυο αισθητήρων μπορεί να χρησιμοποιηθεί για να αναφέρει τις αναπροσαρμογές στη θέση της πηγής γεγονότος στο base-station, ενδεχομένως με τις εκτιμήσεις για την ταχύτητα και την κατεύθυνση. Χαρακτηριστικά οι κόμβοι αισθητήρων πρέπει να συνεργαστούν προτού να μπορέσουν οι αναπροσαρμογές να αναφερθούν στο base-station.

Τα ασύρματα δίκτυα αισθητήρων κυμαίνονται από καλά οργανωμένα, με σταθερή ανάπτυξη των κόμβων αισθητήρων, μέχρι και σε τυχαία ανάπτυξη π.χ. ρίχνοντας μεγάλη ποσότητα κόμβων από ένα αεροπλάνο σε ένα δάσος το οποίο καίγεται. Ακόμη οι αισθητήρες μπορεί να είναι κινητοί για να αντισταθμιστεί η ανεπάρκεια στη διαδικασία επέκτασης με την κίνηση προς τις θέσεις όπου δεν υπάρχει κάλυψη, έτσι ώστε οι στόχοι τους να μπορούν να εκπληρωθούν καλύτερα. Οι κόμβοι μπορούν να γίνουν κινητοί εάν είναι συνδεδεμένοι με άλλα κινούμενα αντικείμενα και το δίκτυο πρέπει να προσαρμοστεί στη νέα τοποθεσία τους ανά πάσα στιγμή.

Η ανάπτυξη των WSN θα προσφέρει τη δυνατότητα για ακόμα περισσότερες εφαρμογές στο μέλλον (Εικόνα 2.20).



Εικόνα 2.20 Το όραμα Sensor Web

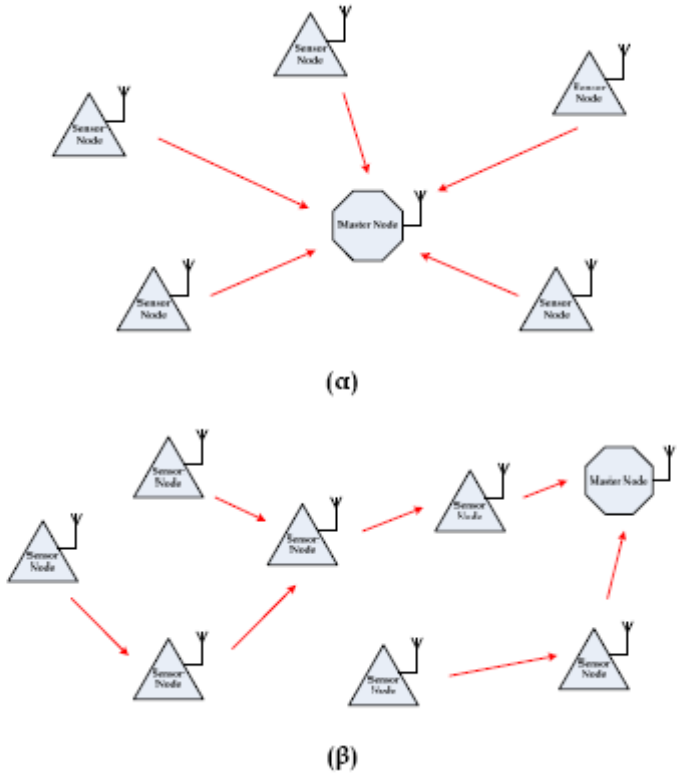
2.4 Επικοινωνία μεταξύ αισθητήρων

2.4.1 Singe και multi-hop μετάδοση πληροφορίας

Ουσιαστική ανάγκη ενός WSN αποτελεί η ύπαρξη στοιχειώδους ασύρματης επικοινωνίας των κόμβων με κάποιον κεντρικό κόμβο ή και μεταξύ τους. Δεδομένης της πυκνής εγκατάστασης των κόμβων, η μετάδοση της πληροφορίας μεταξύ τους μπορεί να γίνει με δύο κυρίως τρόπους: Α) Με ένα βήμα (single-hop). Β) Με πολλαπλά βήματα (multi-hop) (Εικόνα 2.21). Η πρώτη περίπτωση εφαρμόζεται όταν ο κόμβος – αποδέκτης είναι εντός της εμβέλειας όλων των κόμβων. Η εκπομπή πιθανώς να πρέπει να γίνει με την μέγιστη δυνατή ισχύ, για να επιτευχθεί η μέγιστη δυνατή εμβέλεια. Η επικοινωνία με πολλαπλά βήματα είναι πιο ρεαλιστική και προσφέρει περισσότερες δυνατότητες στον σχεδιαστή του δικτύου. Χρησιμοποιώντας την εμβέλεια του κάθε κόμβου, μπορεί να δημιουργηθεί ένα επεκτάσιμο δίκτυο, που θα καλύπτει μια μεγάλη περιοχή και οι πληροφορίες θα μετακινούνται από κόμβο σε κόμβο μέχρι τον τελικό προορισμό τους. Η μέθοδος αυτή αυξάνει κατά πολύ την συνολική κάλυψη του δικτύου, μειώνει την απαιτούμενη ενέργεια, αλλά απαιτεί πιο εξειδικευμένα πρωτόκολλα επικοινωνίας και ισχυρότερους κόμβους από πλευράς υλικού [19].

Οι περισσότεροι Ασύρματοι Κόμβοι χρησιμοποιούν ασύρματους πομποδέκτες που λειτουργούν στην ISM (Industrial, Scientific, Medical) μπάντα, στις συχνότητες των 433.5 – 437.9 MHz, 868.0 – 868.6 MHz και 2400 – 2483.5 MHz. Χρησιμοποιούν δημοφιλείς μεθόδους διαμόρφωσης όπως OOK (On/Off Key), ASK (Amplitude Shift Key) και FSK (Frequency Shift Key). Η

κατανάλωση ισχύος είναι σχεδόν ίδια κατά την εκπομπή και τη λήψη δεδομένων και κυμαίνεται στα 15 – 300 mW, ανάλογα με το ολοκληρωμένο, τη συχνότητα και τη μέθοδο που χρησιμοποιεί για την επικοινωνία. Η ακτίνα επικοινωνίας κυμαίνεται από τα 25 – 200 m, με το μέγιστο να παρατηρείται σε εξωτερικούς χώρους, με οπτική επαφή. Οι ρυθμοί μετάδοσης που επιτυγχάνονται είναι της τάξης των 10 – 100 Kbps και ο χρόνος που χρειάζονται συνήθως τα ολοκληρωμένα, από την στιγμή που θα τροφοδοτηθούν μέχρι να είναι έτοιμα για επικοινωνία, είναι της τάξης των μερικών ms (πάνω από 5 ms θεωρείται απαράδεκτο). Την επικοινωνία των ασύρματων κόμβων μεταξύ τους αναλαμβάνουν τηλεπικοινωνιακά πρωτόκολλα, που φροντίζουν για την ομαλή διεξαγωγή των επικοινωνιών [20],[21],[22].



Εικόνα 2.21 α)Επικοινωνία με ένα βήμα(Single-hop) β) Επικοινωνία με πολλαπλά βήματα(multi-hop)

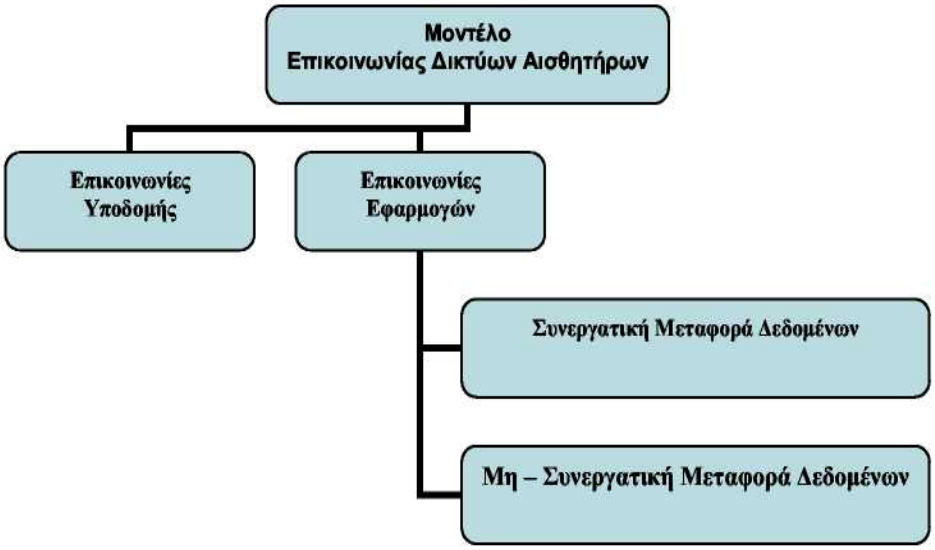
2.4.2 Μοντέλο Επικοινωνίας Ασύρματων Δικτύων Αισθητήρων

Στη βιβλιογραφία υπάρχουν αρκετά προτεινόμενα μοντέλα δικτύου. Σε όλα τα μοντέλα διακρίνονται δύο βασικές κατηγοριοποιήσεις στα μηνύματα που εκπέμπονται ανάμεσα στους κόμβους, ενώ ταυτόχρονα υπάρχει συμφωνία στα επίπεδα του δικτύου που χρησιμοποιούνται.

Μελετώντας όμως το επικοινωνιακά πρότυπα (patterns) που διακινούνται, συστηματικά, ο σχεδιαστής του δικτύου είναι σε θέση να επιλέξει εκείνη την δικτυακή υποδομή και πρωτόκολλο

το οποίο θα συνδυάζει τη βέλτιστη απόδοση και επίδοση με το χαμηλότερο κόστος ανάπτυξης και το μέγιστο χρόνο ζωής.

Νοηματικά, οι μορφές επικοινωνίας σε ένα δίκτυο αισθητήρων χωρίζονται σε δύο κατηγορίες: στην επικοινωνία, που εξυπηρετεί τις εφαρμογές και στην επικοινωνία υποδομής, που εξυπηρετεί την λειτουργία και διαχείριση του δικτύου (Εικόνα 2.22).

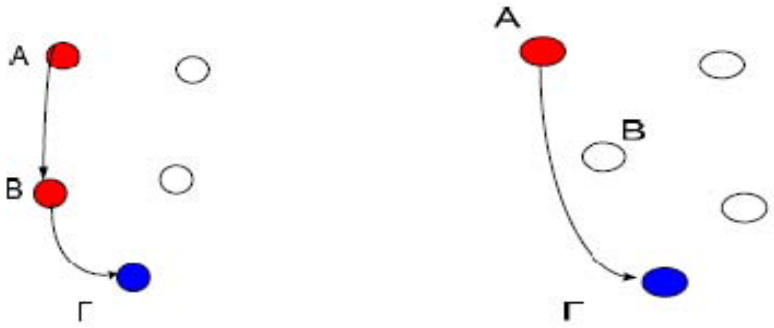


Εικόνα 2.22 Μοντέλο Επικοινωνίας Δικτύων Αισθητήρων

Η επικοινωνία για την εξυπηρέτηση των εφαρμογών σχετίζεται με τη μεταφορά των δεδομένων που έχουν καταγραφεί με σκοπό την ενημέρωση του εκάστοτε ενδιαφερομένου για το φαινόμενο που παρακολουθείται. Δύο είναι τα μοντέλα που κυριαρχούν σε αυτή τη μορφή επικοινωνίας. Το πρώτο απαιτεί τη συνεργασία μεταξύ διαφορετικών κόμβων για τη μεταφορά της πληροφορίας στους ενδιαφερόμενους ενώ, το δεύτερο δεν απαιτεί τη συνεργασία για τη μεταφορά πληροφορίας (Εικόνα 2.23).

Η επικοινωνία υποδομής αναφέρεται σε όλες τις εκπομπές που πρέπει να γίνουν ανάμεσα στους κόμβους και έχουν ως σκοπό την ρύθμιση, συντήρηση και βελτιστοποίηση της λειτουργίας του δικτύου.

Ειδικότερα, εξαιτίας της φύσης των δικτύων αισθητήρων, απαιτείται από τους κόμβους να μπορούν να εντοπίζουν άλλους κόμβους και να αποκαθιστούν επικοινωνία. Έτσι η επικοινωνία υποδομής απαιτείται για τη διατήρηση των συνδέσεων του δικτύου, για την εξασφάλιση του συγχρονισμού ανάμεσα στους κόμβους αλλά και για τη βελτιστοποίηση της συνολικής απόδοσης του δικτύου.

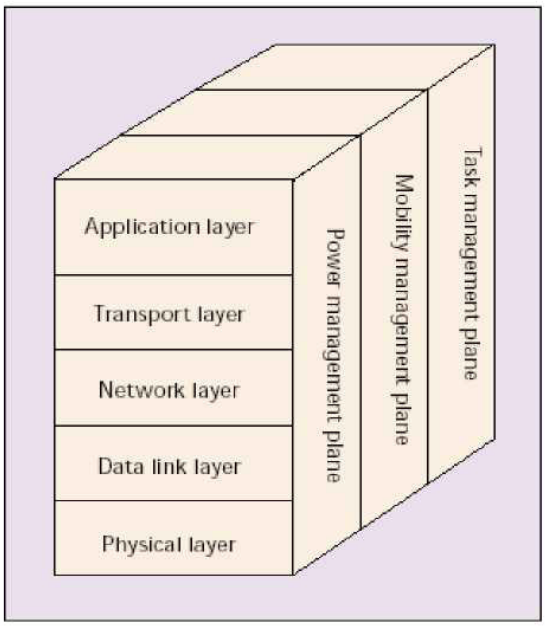


Εικόνα 2.23 (α) Συνεργατική (β) Μη -Συνεργατική Μέθοδος Μεταφοράς δεδομένων.

2.4.3 Τα layers στα Ασύρματα Δίκτυα Αισθητήρων

Για την υλοποίηση του παραπάνω μοντέλου συνεργατικής επικοινωνίας απαιτείται η συνεργασία πρωτοκόλλων σε όλο το εύρος της στοίβας που προβλέπει το μοντέλο OSI και ορισμένων επιπλέον 'επιπέδων'. Επιπλέον, στην Εικόνα 2.24 βλέπουμε πώς διαμορφώνεται το πρότυπο του OSI και πώς προστίθενται κάθετα σε αυτό οι επιπλέον στοίβες της ισχύος, της κινητικότητας και της διαχείρισης της κατανομής του φόρτου ανάμεσα στους κόμβους του δικτύου.

Οι στοίβες αυτές διατρέχουν όλα τα επίπεδα και βοηθούν στο συντονισμό των αισθητήρων με απώτερο σκοπό τη συνολική μείωση της καταναλισκόμενης ενέργειας και στην αύξηση της αποδοτικότητας του δικτύου.



Εικόνα 2.24 Τελική μορφή μοντέλου OSI -Τα Layers στα Ασύρματα Δίκτυα Αισθητήρων [18]

Το physical layer είναι υπεύθυνο για την επιλογή της συχνότητας, την παραγωγή συχνότητας μετάδοσης, την ανίχνευση σήματος, τη διαμόρφωση του και για την κρυπτογράφηση των δεδομένων. Μέχρι σήμερα, είναι από όλους αποδεκτές οι συχνότητες των 816 MHz, 915 MHz και 2.4 GHz ISM για τα ασύρματα δίκτυα αισθητήρων.

2.4.3.1 Data link layer

Το data link layer είναι υπεύθυνο να κάνει multiplexing της ροής των δεδομένων, να ανιχνεύει τα data frames καθώς και για medium access και error control. Επίσης, διαφυλάσσει αξιόπιστες point to point και point to multipoint επικοινωνίες στο δίκτυο. Το πρωτόκολλο MAC, σε ένα ασύρματο multihop και αυτό-οργανωτικό δίκτυο αισθητήρων, έχει ως στόχους τη δημιουργία της υποδομής του δικτύου και το δίκαιο και αποδοτικό διαχωρισμό των πόρων επικοινωνίας μεταξύ των κόμβων του δικτύου. Οι δύο γνωστοί μέθοδοι για error control είναι οι forward error correction (FEC) και automatic repeat request (ARQ). Η χρησιμότητα του ARQ σε multihop δίκτυα αισθητήρων είναι περιορισμένη από το επιπλέον κόστος ενέργειας αναμετάδοσης και το overhead. Από την άλλη, η πολυπλοκότητα αποκωδικοποίησης του FEC είναι μεγαλύτερη αφού απαιτείται η ενσωμάτωση ικανοτήτων error control. Λαμβάνοντας αυτό υπόψη, οι απλοί κώδικες error control με μικρή πολυπλοκότητα κωδικοποίησης και αποκωδικοποίησης παρουσιάζονται ως οι καλύτερες λύσεις στα ασύρματα δίκτυα αισθητήρων.

2.4.3.2 Network layer

Παραδοσιακές ad-hoc τεχνικές δρομολόγησης συνήθως δεν ταιριάζουν στις απαιτήσεις των ασύρματων δικτύων αισθητήρων. Το network layer των δικτύων αισθητήρων είναι σχεδιασμένο σύμφωνα με τις πιο κάτω αρχές:

ο Πρέπει να διαχειρίζονται αποδοτικά τα διαθέσιμα ποσά ενέργειας τους.

ο Πρέπει να είναι data-centric.

ο Η συνάθροιση των δεδομένων είναι χρήσιμη μόνο όταν δεν εμποδίζεται η συνεργασία μεταξύ των κόμβων αισθητήρων.

ο Σε ένα ιδανικό δίκτυο αισθητήρων, οι διευθύνσεις των κόμβων είναι attribute-based και δεν γνωρίζουν την ακριβή τοποθεσία τους οι κόμβοι.

Τα δρομολόγια τα οποία είναι energy – efficient μπορούν να βρεθούν βάση της διαθέσιμης εναπομένουσας ενέργειας σε κάθε κόμβο (available power - PA) ή της ενέργειας που απαιτείται για τη μετάδοση των δεδομένων μέσω των δρομολογίων.

Οι πιο γνωστοί αλγόριθμοι για την επιλογή του πιο αποδοτικού δρομολογίου όσον αφορά την ενέργεια είναι οι πιο κάτω:

ο **Maximum PA route:** Επιλέγεται το δρομολόγιο το οποίο έχει τη μεγαλύτερη συνολική εναπομένουσα ενέργεια, η οποία υπολογίζεται με το άθροισμα της εναπομένουσας ενέργειας του κάθε κόμβου ο οποίος ανήκει στο συγκεκριμένο δρομολόγιο.

ο **Minimum energy (ME) route:** Επιλέγεται το δρομολόγιο το οποίο απαιτεί την ελάχιστη ενέργεια για τη μεταφορά των πακέτων μεταξύ των base - stations και των κόμβων αισθητήρων.

ο **Minimum hop route (MH):** Επιλέγεται το δρομολόγιο το οποίο θα έχει τον ελάχιστο αριθμό hops μέχρι το base - station.

ο **Maximum minimum PA note route:** Επιλέγεται το δρομολόγιο του οποίου η ελάχιστη εναπομένουσα ενέργεια θα είναι μεγαλύτερη από την ελάχιστη εναπομένουσα ενέργεια των υπολοίπων δρομολογίων.

Μια από τις υπευθυνότητες του network layer είναι να παρέχει διασύνδεση με εξωτερικά δίκτυα (πχ. άλλα δίκτυα αισθητήρων ή το διαδίκτυο). Σε ένα σενάριο το base - station μπορεί να χρησιμοποιηθεί ως το gateway προς άλλα δίκτυα.

2.4.3.3 Transport Layer

Αυτό το layer είναι χρήσιμο κυρίως όταν το σύστημα είναι σχεδιασμένο έτσι ώστε να μπορεί να έχει πρόσβαση στο διαδίκτυο ή σε άλλα εξωτερικά δίκτυα. Μια προσέγγιση παρόμοια με αυτήν του TCP μπορεί να φανεί χρήσιμη για την επικοινωνία με άλλου είδους δίκτυα όπως το διαδίκτυο. Η επικοινωνία μεταξύ κόμβων του δικτύου δεν μπορεί να υποστηριχθεί με πρωτόκολλα των οποίων η προσέγγιση είναι παρόμοια με του πρωτοκόλλου TCP, λόγω της περιορισμένης μνήμης που διαθέτουν οι κόμβοι αισθητήρων.

2.4.3.4 Application Layer

Παρόλο που έχουν οριστεί πολλές περιοχές εφαρμογών για ασύρματα δίκτυα αισθητήρων, δεν έχει ερευνηθεί μεγάλος αριθμός πρωτοκόλλων για το application layer. Μερικά πρωτόκολλα του application layer τα οποία βρίσκονται υπό έρευνα είναι:

ο Sensor Management Protocol (SMP)

ο Task Assignment and Data advertisement Protocol (TADAP)

ο Sensor Query and Data Dissemination Protocol (SQDDP)

Power management plane: Είναι υπεύθυνο για τη διαχείριση της ενέργειας ενός κόμβου αισθητήρα. Για παράδειγμα, για την αποφυγή παραλαβής διπλών μηνυμάτων, ο κόμβος αισθητήρα μπορεί να κλείσει το δέκτη του μετά από παραλαβή ενός μηνύματος από κάποιο από τους γείτονες του. Ακόμη, όταν ο κόμβος αισθητήρα δεν έχει μεγάλα αποθέματα ενέργειας, ενημερώνει τους γείτονες του για να μην συμμετάσχει σε μελλοντική δρομολόγηση πακέτων και έτσι η εναπομένουσα ενέργεια που διαθέτει θα χρησιμοποιείται μόνο για την ανίχνευση γεγονότων.

Mobility management plane: Εντοπίζει και καταγράφει την κίνηση των κόμβων αισθητήρων έτσι ώστε η δρομολόγηση πίσω προς το χρήστη να είναι πάντα δυνατή και έτσι ώστε οι κόμβοι να γνωρίζουν ποιοι είναι οι γείτονές τους.

Task management plane: Ισοζυγίζει και προγραμματίζει τις διεργασίες για αίσθηση σε μια συγκεκριμένη περιοχή.

Η ευελιξία, η μεγάλη ανοχή σε λάθη, η υψηλή αξιοπιστία αίσθησης, το μικρό κόστος και η γρήγορη ανάπτυξη ενός δικτύου αισθητήρων είναι μερικοί από τους σημαντικότερους παράγοντες που δημιουργούν πολλές καινούριες και συναρπαστικές περιοχές για ανάπτυξη εφαρμογών. Σύντομα αυτό το μεγάλο εύρος εφαρμογών θα κάνει τα ασύρματα δίκτυα αισθητήρων μέρος της καθημερινής μας ζωής.

2.5 Τεχνικές προκλήσεις και απαιτήσεις των ασύρματων δικτύων αισθητήρων

Ο σχεδιασμός των WSN στηρίζεται σε μια ή περισσότερες από τις παρακάτω τεχνικές προκλήσεις [13, 14]:

- *Εκτενής και ταυτόχρονα τυχαία παράταξη κόμβων αισθητήρων* : Τα περισσότερα WSN αποτελούνται από ένα μεγάλο αριθμό κόμβων αισθητήρων, οι οποίοι απλώνονται τυχαία στις περιοχές ενδιαφέροντος ή παρατάσσονται πυκνά σε συγκεκριμένα κρίσιμα σημεία (απρόσιτες περιοχές). Το σύστημα αυτορυθμίζεται πριν την έναρξη της εφαρμογής που εξυπηρετεί.

- *Πλεονασμός δεδομένων* : Η πυκνή παράταξη των κόμβων αισθητήρων συμβάλλει στον υψηλό συσχετισμό των δεδομένων που παρέχονται από κάποιον από αυτούς με τα δεδομένα των γειτονικών του.

- *Περιορισμένοι πόροι* : Ο σχεδιασμός και η υλοποίηση των WSN περιορίζονται από τέσσερις τύπους πόρων: ενέργεια, υπολογιστική ισχύ, μνήμη και εύρος ζώνης. Λόγω του μικρού μεγέθους των κόμβων αισθητήρων, η μόνη δυνατή πηγή ενέργειας είναι οι μπαταρίες. Επιπλέον, τα WSN αναπτύσσονται με τέτοιο τρόπο που έχει ως αποτέλεσμα οι κόμβοι τους να μην είναι προσπελάσιμοι, οπότε οι μπαταρίες των τελευταίων δεν μπορούν να επαναφορτισθούν ή να αντικατασταθούν. Συγχρόνως, οι μνήμες τους είναι περιορισμένες με συνέπεια να μπορεί να επιτευχθεί επίσης περιορισμένη υπολογιστική ισχύ, ενώ το εύρος ζώνης στο ασύρματο μέσο επικοινωνίας είναι σημαντικά μικρό.

- *Ad hoc αρχιτεκτονική και αυτόνομη λειτουργία*: Η έλλειψη συγκεκριμένης υποδομής και της ανθρώπινης επέμβασης στη λειτουργία των WSN κάνουν το σύστημα να σχεδιάζεται έτσι ώστε να δημιουργεί, να διατηρεί και γενικά να ελέγχει από μόνο του τις συνδέσεις μέσα στο δίκτυο (αυτόνομο).

- *Δυναμικές τοπολογίες και περιβάλλον* : Αφ' ενός, η τοπολογία και η συνδεσιμότητα ενός WSN συχνά ποικίλει εξαιτίας της περιορισμένης αξιοπιστίας των κόμβων αισθητήρων ως

προς τις λειτουργίες τους. Για παράδειγμα, μπορεί ένας κόμβος να σταματήσει να λειτουργεί λόγω εξάντλησης της παρεχόμενης ενέργειας οποιαδήποτε στιγμή, χωρίς να ενημερώσει κατάλληλα τους άλλους κόμβους εκ των προτέρων. Επίσης, νέοι κόμβοι μπορούν να προστεθούν σε τυχαία χρονική στιγμή και θέση σε μια περιοχή χωρίς να υπάρχει προγενέστερη δήλωση του συνόλου των κόμβων που συμμετέχουν στο δίκτυο. Αφ' ετέρου, τώρα, το περιβάλλον που παρακολουθούν τα WSN ποικίλει και αυτό, πράγμα το οποίο μπορεί να κάνει μερικούς κόμβους αισθητήρων να δυσλειτουργήσουν.

- *Επιρρεπές σε λάθη ασύρματο μέσο* : Οι κόμβοι αισθητήρων συνδέονται μεταξύ τους μέσω του ασύρματου μέσου επικοινωνίας, στο οποίο συμβαίνουν αρκετά λάθη. Για παράδειγμα, σε μερικές εφαρμογές το περιβάλλον επικοινωνίας είναι αρκετά θορυβώδες και μπορεί να προκαλέσει αλλοιώσεις στα δεδομένα των μεταδιδόμενων σημάτων.

- *Ποικιλία από εφαρμογές* : Όπως έχει αναφερθεί και σε προηγούμενη ενότητα, τα WSN μπορούν να χρησιμοποιηθούν στην ανάπτυξη ποικίλων εφαρμογών, όπως στόχευση, ανίχνευση αντικειμένων, παρακολούθηση συνθηκών περιβάλλοντος κ.λπ. Οι απαιτήσεις για τις διάφορες εφαρμογές ποικίλουν και αυτές σημαντικά.

- *Ασφάλεια και απόκρυψη δεδομένων από τρίτους* : Η ασφάλεια και η απόκρυψη των μεταδιδόμενων δεδομένων αποτελεί δύο από τα σημαντικότερα χαρακτηριστικά που πρέπει να λαμβάνονται υπόψη κατά τον σχεδιασμό των WSN, όπως στις περιπτώσεις των στρατιωτικών εφαρμογών και αλλού. Ωστόσο, η ασφάλεια φαίνεται να είναι ένα σημαντικά δύσκολο πρόβλημα προς επίλυση στα WSN εξαιτίας του γεγονότος ότι απαιτείται η χρησιμοποίηση περισσότερων πόρων, οι οποίοι ως γνωστόν είναι περιορισμένοι στα δίκτυα αυτά.

- *Ποιότητα συστήματος* : Η ποιότητα που παρέχεται από ένα WSN αναφέρεται στην ακρίβεια με την οποία τα δεδομένα που λαμβάνονται από τους κόμβους αισθητήρων αντιπροσωπεύουν αυτό που πραγματικά συμβαίνει στο περιβάλλον τους. Σε αντίθεση με άλλα δίκτυα, ένα WSN στηρίζεται στη συνάθροιση των λαμβανόμενων δεδομένων από κάθε κόμβο για την περιγραφή, για παράδειγμα, ενός φαινομένου και όχι σε μεμονωμένα δεδομένα. Κατά συνέπεια, το πλήθος ή η πυκνότητα των κόμβων προσδιορίζει και την αντίστοιχη ποιότητα του συστήματος. Ωστόσο, μία άλλη παράμετρος που επηρεάζει την ποιότητα είναι το γεγονός ότι σε τέτοια συστήματα, όπου χρησιμοποιούνται τα WSN, υπάρχει σημαντική καθυστέρηση - λανθάνουσα κατάσταση που κάνει τα δεδομένα κάθε χρονική στιγμή να μην αντιπροσωπεύουν την παρούσα κάθε φορά κατάσταση του περιβάλλοντος του δικτύου, αλλά μία προηγούμενη ή μεταβατική, γεγονός το οποίο μπορεί πολλές φορές να οδηγήσει σε λανθασμένες αντιδράσεις από μέρους των χειριστών του συστήματος.

2.5.1 Στόχοι σχεδιασμού των ασύρματων δικτύων αισθητήρων

Οι στόχοι σύμφωνα με τους οποίους αναπτύσσεται ο σχεδιασμός των WSN, ώστε να ανταποκρίνονται στις προκλήσεις και να ικανοποιούν τις απαιτήσεις των διάφορων εφαρμογών [13, 14, 15, 16, 17], αναφέρονται παρακάτω:

- *Μικρές και φθηνές συσκευές κόμβων αισθητήρων* : Το χαμηλό κόστος και το μικρό μέγεθος των συσκευών αισθητήρων αποτελούν σημαντικούς παράγοντες που συμβάλλουν στην εκτενή και τυχαία ανάπτυξη και επέκταση των WSN σε μια περιοχή ενδιαφέροντος. Για μια μεγάλης κλίμακας WSN εφαρμογή, το κόστος των συσκευών αισθητήρων συμβάλλει σημαντικά στο συνολικό κόστος της εφαρμογής αυτής. Εκτός αυτού, όσο μικρότερη είναι η συσκευή, τόσο μικρότερη αλλοίωση δημιουργείται στα δεδομένα των αισθητήρων και τόσο πιο εύκολα γίνεται η τοποθέτησή τους για το σχηματισμό ενός WSN δικτύου.

- *Κλιμακωτές και ευέλικτες αρχιτεκτονικές και πρωτόκολλα* : Η αρχιτεκτονική των WSN πρέπει να είναι κλιμακωτή και ευέλικτη έτσι ώστε να μπορεί να επιτευχθεί διεύρυνσή τους. Οι μέθοδοι που αποσκοπούν στην ανάπτυξη των δύο αυτών χαρακτηριστικών περιλαμβάνουν την ομαδοποίηση (clustering), τη multi-hop μετάδοση δεδομένων και τη συγκέντρωση της υπολογιστικής ισχύος ανά ομάδα κόμβων.

- *Τοπική επεξεργασία και συνάθροιση δεδομένων* : Προκειμένου να αποβληθεί κάπως ο πλεονασμός στα δεδομένα, οι κόμβοι αισθητήρων μπορούν να συνεργαστούν έτσι ώστε να αναπτύξουν ποικίλη τοπική επεξεργασία. Έτσι, αντί να στέλνουν τα μη επεξεργασμένα δεδομένα κατευθείαν στον προορισμό, οι κόμβοι αισθητήρων μπορούν τοπικά να τα αθροίζουν-ομαδοποιούν και να τα επεξεργάζονται, σύμφωνα με τις απαιτήσεις της κάθε εφαρμογής, και ύστερα να τα στέλνουν κατάλληλα μορφοποιημένα.

- *Αποδοτικότητα σχεδιασμού ως προς τη χρήση των πόρων* : Στα WSN, η αποδοτικότητα ως προς τη χρήση των πόρων είναι εξαιρετικά κρίσιμη και επιθυμητή παρά την πολυπλοκότητα που μπορεί να απαιτεί. Προ πάντων, τα αποδοτικά ως προ ς την κατανάλωση της ενέργειας πρωτόκολλα είναι ιδιαίτερα επιθυμητά προκειμένου να επεκταθεί η διάρκεια ζωής του συστήματος. Πράγματι, η μείωση της κατανάλωσης ενέργειας πρέπει να επιτυγχάνεται σε κάθε κομμάτι του δικτύου με διάφορους μηχανισμούς, όπως εξοικονόμηση ενέργειας στο MAC επίπεδο επικοινωνίας, περιορισμένης ενέργειας δρομολόγηση δεδομένων στο επίπεδο δικτύου κ.λπ.. Επιπλέον, προσπάθειες πρέπει να καταβάλλονται προκειμένου να επιτευχθεί αύξηση της αποδοτικότητας των συστημάτων ως προς τη χρησιμοποίηση άλλων πόρων. Για παράδειγμα, χρησιμοποιώντας αλγορίθμους με μικρή πολυπλοκότητα μπορεί να μειωθεί ο χρόνος υπολογισμών με αποτέλεσμα να μειωθούν επίσης, τόσο η κατανάλωση ενέργειας όσο και η καθυστέρηση διάδοσης δεδομένων. Οι αποδοτικές ως προς το εύρος ζώνης αρχιτεκτονικές και τα πρωτόκολλα μπορούν επίσης να μειώσουν την καθυστέρηση διάδοσης δεδομένων.

- *Αυτορυθμιζόμενα δίκτυα* : Οι κόμβοι αισθητήρων πρέπει να μπορούν να ρυθμιστούν από μόνοι τους ώστε να δημιουργηθούν οι απαραίτητες συνδέσεις στο δίκτυο και να αρχίσει η λειτουργία του όλου συστήματος. Για το λόγο αυτό, τα WSN είναι ιδιαίτερα δυναμικά κατά τη διάρκεια ζωής τους. Οι καταστάσεις, τώρα, από τις οποίες διέρχονται οι κόμβοι

με σκοπό την εξοικονόμηση ενέργειας είναι: Απενεργοποίησης, «Ύπνου», Εκκίνησης, Αναμονής, Εκπομπής, Λήψης και Αποτυχίας. Κατά συνέπεια, τα πρωτόκολλα των WSN πρέπει να έχουν την ικανότητα να δημιουργούν συνδέσεις αυτόνομα και ανεξάρτητα από την κατάσταση των κόμβων αισθητήρων, αποσκοπώντας παράλληλα στην επίτευξη των ιδιαίτερων χαρακτηριστικών που έχουν, όπως εξοικονόμηση ενέργειας, μείωση της καθυστέρησης διάδοσης των δεδομένων κ.λπ..

- *Προσαρμοστικότητα* : Για να ανταπεξέλθουν στις δυναμικές-ποικίλες συνθήκες λειτουργίας τους, τα WSN πρέπει να προσαρμόζονται στη μεταβαλλόμενη κατάσταση των συνδέσεων και της ροής δεδομένων κατά τη διάρκεια του χρόνου. Για παράδειγμα, στην περίπτωση περιβάλλοντος με μεγάλο θόρυβο επικοινωνίας, θα πρέπει να «θυσιασθεί» ενέργεια προκειμένου να αυξηθεί η ακρίβεια, αυξάνοντας το λόγο σήματος προς θόρυβο.

- *Αξιοπιστία και ανοχή σφαλμάτων* : Σε πολλές εφαρμογές, τα δεδομένα πρέπει να μεταδοθούν αξιόπιστα μέσω ενός θορυβώδους, επιρρεπούς σε λάθη και χρονικά μεταβαλλόμενου ασύρματου καναλιού. Σε τέτοιες περιπτώσεις, η επαλήθευση και η διόρθωση των δεδομένων σε κάθε επίπεδο επικοινωνίας του δικτύου είναι κρίσιμα για την παροχή ακριβέστερων αποτελεσμάτων.

- *Σχεδιασμός ανάλογα με την εφαρμογή* : Επειδή κανένα πρωτόκολλο δεν ικανοποιεί της απαιτήσεις όλων των εφαρμογών των WSN, ο σχεδιασμός τους είναι σε πολλές περιπτώσεις υψηλά εξαρτώμενος από την εφαρμογή.

- *Ασφάλεια* : Η απόκρυψη και η ασφάλεια της επικοινωνίας στα WSN είναι ιδιαίτερα σημαντικές κατά το σχεδιασμό των δικτύων αυτών.

- *Ποιότητα του συστήματος με περιορισμένους πόρους* : Όπως αναφέρθηκε προηγουμένως, η ποιότητα ενός συστήματος με WSN περιγράφεται από την ακρίβεια και την εγκυρότητα των δεδομένων που μεταδίδονται. Γενικά, το ποσό των δεδομένων που διακινούνται μέσα στο δίκτυο καθορίζουν το επίπεδο της ακρίβειας, αλλά από την άλλη υψηλό ποσό δεδομένων συνεπάγεται μεγάλη κατανάλωση του εύρους ζώνης και περισσότερο ανταγωνισμό κατά τη διάρκεια της μετάδοσης μεταξύ των κόμβων, που σημαίνει με τη σειρά του αυξημένη κατανάλωση ενέργειας και καθυστέρηση διάδοσης των δεδομένων (μειώνεται η εγκυρότητα). Κάποια λύση μπορεί να δώσει η τοπική επεξεργασία δεδομένων περιορίζοντας το ποσό των δεδομένων που μεταδίδονται, αλλά η περιπλοκότητα των υπολογισμών και η χρήση της μνήμης μπορούν να προκαλέσουν μεγάλη καθυστέρηση της μετάδοσής τους και αύξηση της ενέργειας που καταναλώνεται.

Γενικά, είναι αδύνατο να επιτευχθούν όλοι οι στόχοι σε ένα σύστημα με WSN. Οι περισσότεροι σχεδιασμοί συστημάτων με WSN εξαρτώνται σημαντικά από την εφαρμογή που υλοποιούν και δίνουν διαφορετική βαρύτητα στον κάθε στόχο. Κατά συνέπεια, τα πρωτόκολλα αντίστοιχα πρέπει να σχεδιάζονται έτσι ώστε να ικανοποιούν τις απαιτήσεις της κάθε εφαρμογής και να γίνεται μια κατάλληλη επιλογή των διαφόρων παραμέτρων που αφορούν το σχεδιασμό σύμφωνα με τη βαρύτητα που δίνεται σε κάθε στόχο. Ένα ασύρματο δίκτυο αισθητήρων πρέπει να βασιστεί σε μεγάλο βαθμό στην εκμετάλλευση διαφόρων trade-offs μεταξύ αμοιβαία

αντιφατικών στόχων που αφορούν το σχεδιασμό του συστήματος και των πρωτοκόλλων. Ο *Πίνακας 2.2* συνοψίζει τις τεχνικές προκλήσεις και απαιτήσεις των WSN και τους αντίστοιχους στόχους σχεδιασμού.

Πίνακας 2-2 Σύνοψη των τεχνικών προοπτικών και απαιτήσεων και των στόχων

Τεχνικές προοπτικές και απαιτήσεις	**Στόχοι σχεδιασμού**
Εκτενής και ταυτόχρονα τυχαία παράταξη κόμβων αισθητήρων	Μικρές και φθηνές συσκευές κόμβων αισθητήρων, κλιμακωτές και ευέλικτες αρχιτεκτονικές και πρωτόκολλα
Πλεονασμός δεδομένων	Τοπική επεξεργασία και συνάθροιση δεδομένων
Περιορισμένοι πόροι	Αποδοτικότητα σχεδιασμού ως προς τη χρήση των πόρων
Ad hoc αρχιτεκτονική και αυτόνομη λειτουργία	Αυτορυθμιζόμενα δίκτυα
Δυναμικές τοπολογίες και περιβάλλον	Προσαρμοστικότητα
Επιρρεπές σε λάθη ασύρματο μέσο	Αξιοπιστία και ανοχή σφαλμάτων
Ποικιλία από εφαρμογές	Σχεδιασμός ανάλογα με την εφαρμογή
Ασφάλεια και απόκρυψη δεδομένων από τρίτους	Ασφάλεια
Ποιότητα συστήματος	Ποιότητα του συστήματος με περιορισμένους πόρους

2.6 Ενέργεια στα Ασύρματα Δίκτυα Αισθητήρων

Από τη στιγμή, που θα εγκατασταθεί ένα Ασύρματο Δίκτυο Αισθητήρων σε έναν χώρο, προβλέπεται να λειτουργήσει, για ένα μεγάλο χρονικό διάστημα, χωρίς την ανθρώπινη επίβλεψη. Αυτό σημαίνει, ότι θα πρέπει να διαχειριστεί τους ενεργειακούς του πόρους με τέτοιο τρόπο, ώστε να μεγιστοποιηθεί ο χρόνος ζωής του. Κύρια πηγή ενέργειας των κόμβων είναι οι μπαταρίες, οι οποίες συνήθως δεν είναι επαναφορτιζόμενες, αλλά έχουν προταθεί και μέθοδοι

παραγωγής ενέργειας επάνω στον κόμβο και αποθήκευσής της σε επαναφορτιζόμενες μπαταρίες και υπέρ-πυκνωτές, με σκοπό την παράταση του χρόνου ζωής του [28]. Η ενεργειακή κατανάλωση είναι ο δείκτης εκείνος που θα καθορίσει την διάρκεια λειτουργίας του δικτύου.

Ο σχεδιασμός του δικτύου και των κόμβων, που το αποτελούν, θα πρέπει να είναι τέτοιος, ώστε να ελαχιστοποιείται η κατανάλωση ενέργειας και κάθε υποσύστημα του κόμβου θα πρέπει να είναι κατασκευασμένο με υλικό χαμηλής κατανάλωσης ενέργειας. Ειδική μέριμνα θα πρέπει να ληφθεί και σε επίπεδο λογισμικού, αλγορίθμων επεξεργασίας σήματος και πρωτοκόλλων επικοινωνίας. Μόνο ένας καθολικός σχεδιασμός χαμηλής ενέργειας θα προσφέρει στο δίκτυο την αναμενόμενη απόδοση και μακροζωία [20],[25]-[29].

2.6.1 Μετρικές για την κατανάλωση ενέργειας

Διάφορες μετρικές για την κατανάλωση ενέργειας είναι οι πιο κάτω:

- **Energy per correctly received bit:** Η ποσότητα ενέργειας που καταναλώνεται για τη μεταφορά ενός bit δεδομένων από το source στο base - station

- **Energy per reported (unique) event:** Η μέση ποσότητα ενέργειας που καταναλώνεται για την αναφορά ενός γεγονότος. Έχοντας υπόψη ότι το ίδιο γεγονός μπορεί να αναφερθεί από πολλά sources ταυτόχρονα, συνηθίζεται να γίνεται κοινωνικοποίηση της μετρικής και λαμβάνονται υπόψη μόνο τα μοναδικά γεγονότα.

- **Delay/energy trade-offs:** Μερικές εφαρμογές πρέπει όταν ανιχνεύσουν ένα γεγονός, να το στείλουν "επειγόντως" στο base - station. Αυτό απαιτεί αύξηση της κατανάλωσης της ενέργειας των κόμβων για την ταχεία αποστολή της αναφοράς στο base - station.

- **Network lifetime:** Ο χρόνος που το δίκτυο είναι λειτουργικό, δηλαδή μπορεί να ολοκληρώσει την αποστολή του με επιτυχία. Σο δίκτυο μπορεί να θεωρηθεί μη λειτουργικό, είτε όταν πάψει να λειτουργεί ο πρώτος κόμβος, είτε αν 50% (ή ένα άλλο ποσοστό) των κόμβων του ξεμείνουν από ενέργεια, είτε αν το δίκτυο διασπαστεί σε μικρότερα ανεξάρτητα κομμάτια τα οποία δεν επικοινωνούν μεταξύ τους γιατί ο κόμβος που ενώνει τα κομμάτια αυτά είναι νεκρός. Μέσα σε αυτή την κατηγορία μπορεί να θεωρηθεί και ο υπολογισμός του χρόνου μέχρι το δίκτυο να πάψει να καλύπτει ένα κομμάτι της περιοχής για την οποία είναι υπεύθυνο.

Όλες αυτές οι μετρικές μπορούν να υπολογιστούν μόνο υπό κάποιες υποθέσεις που αφορούν τα χαρακτηριστικά κατανάλωσης ενέργειας για τον κάθε κόμβο, το πραγματικό φορτίο το οποίο μπορεί να αντέξει το δίκτυο και την συμπεριφορά του καναλιού των συχνοτήτων που επικοινωνούν οι κόμβοι.

2.6.2 Τεχνικές για τη μείωση της κατανάλωσης ενέργειας

2.6.2.1 Χαμηλού duty-cycle λειτουργία

Γενικά, το περιβάλλον που υπόκειται σε παρατήρηση με τη χρήση κάποιου WSN, δεν μεταβάλλεται ακαριαία (π.χ. ομαλή μεταβολή της θερμοκρασίας κ.λπ.), οπότε ο κάθε κόμβος αισθητήρων και κατά συνέπεια και τα επιμέρους τμήματα από τα οποία αποτελείται πρέπει να μεταβαίνουν με εναλλασσόμενο τρόπο σε ενεργές και ανενεργές καταστάσεις με σκοπό τη μείωση της καταναλισκόμενης ενέργειας. Οι δύο σημαντικότεροι παράγοντες στους οποίους οφείλεται η κατανάλωση της ενέργειας σε έναν κόμβο αισθητήρων είναι η επεξεργασία και η εκπομπή-λήψη δεδομένων.

Η ενέργεια που καταναλώνεται από την μονάδα του επεξεργαστή μπορεί να μοντελοποιηθεί ως εξής [18, 19]:

$$E_{total} = E_{switch} + E_{leakage} = C_{total} \times V_{dd}{}^2 + (V_{dd} \times t) I_0 e^{\frac{V_{dd}}{n' V_T}}$$

Η συνολική καταναλισκόμενη ενέργεια, δηλαδή, αποτελείται από δύο μέρη: ενέργεια εναλλαγής καταστάσεων (switching) και ενέργεια διαρροών (leakage). Η ενέργεια εναλλαγής καταστάσεων καθορίζεται από την εφαρμοζόμενη τάση, V_{dd}, και τη συνολική χωρητικότητα που εναλλάσσει κατάσταση κατά τη διάρκεια κάποιας διεργασίας-υπολογισμού, C_{total}. Η ενέργεια διαρροών αναφέρεται στις περιόδους εκείνες όπου δε διενεργείται καμιά διεργασία-υπολογισμός. Επίσης, V_T είναι η θερμική τάση και I_0 το ρεύμα διαρροής. Σύμφωνα, λοιπόν, με την παραπάνω εξίσωση, ένας αποτελεσματικός τρόπος να περιοριστεί η κατανάλωση ενέργειας στον επεξεργαστή είναι να περιοριστεί η ενέργεια που χάνεται κατά την περίοδο που δε διενεργούνται καθόλου διεργασίες, η ενέργεια, δηλαδή, διαρροών.

Όσον αφορά τώρα τη μονάδα επικοινωνίας, ένας πιθανός τρόπος μείωσης της καταναλισκόμενης ενέργειας είναι η απενεργοποίηση των αντίστοιχων κυκλωμάτων (συνθέτες συχνοτήτων, μίκτες κ.λπ.) κατά τη διάρκεια περιόδων αδράνειας και ενεργοποίηση αυτών όταν επιζητείται η επικοινωνία με κάποιον κόμβο. Η μέση ισχύς που καταναλώνεται από τη μονάδα επικοινωνίας μπορεί να μοντελοποιηθεί ως εξής [18]:

$$P_{aver} = N_{tx} \times [P_{tx} \times (T_{tx-on} + T_{start}) + P_{out} \times T_{tx-on}] + N_{rx} \times [P_{rx} (T_{rx-on} + T_{start})]$$

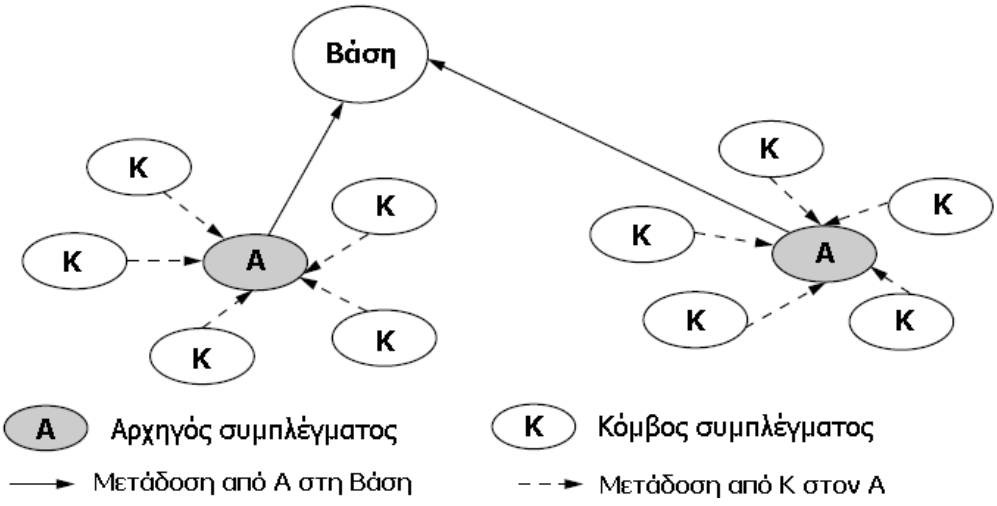
όπου $N_{tx/rx}$ είναι ο μέσος αριθμός των περιόδων ανά δευτερόλεπτο που η μονάδα επικοινωνίας είναι ενεργή, $P_{tx/rx}$ είναι η ισχύς που καταναλώνεται κατά την εκπομπή/λήψη στη μονάδα επικοινωνίας, P_{out} είναι η ισχύς εκπομπής, T_{start} είναι ο χρόνος εκκίνησης της μονάδας επικοινωνίας και $T_{tx\text{-}on/rx\text{-}on}$ είναι ο ενεργός χρόνος αποστολής/λήψης δεδομένων, ο οποίος είναι $T_{tx-on/rx-on} = L / R$, όπου L είναι το μήκος των μεταδιδόμενων πακέτων σε bit και R είναι ο ρυθμός μετάδοσης δεδομένων σε bit ανά δευτερόλεπτο. Όπως είναι φανερό, μείωση της καταναλισκόμενης ενέργειας σε αυτή τη μονάδα μπορεί να επιτευχθεί με την απενεργοποίησή της σε περίπτωση που δεν υπάρχει κάποια επικοινωνία και την ολοκλήρωση της διαδικασίας εκπομπής/λήψης σε σύντομο χρονικό διάστημα (μείωση δηλαδή του duty-cycle). Ωστόσο, η μετάβαση από ενεργή σε ανενεργή κατάσταση και αντίστροφα αυξάνει την καθυστέρηση μετάδοσης δεδομένων και την κατανάλωση ενέργειας. Για το λόγο αυτό, είναι απαραίτητη η ανάπτυξη σχεδιασμών οι οποίοι να λαμβάνουν υπόψη τους το επίπεδο της μετάδοσης δεδομένων

(ποσότητα και ρυθμός μετάδοσης) στο ασύρματο μέσο, ώστε να γίνεται εναλλαγή καταστάσεων με βέλτιστο τρόπο.

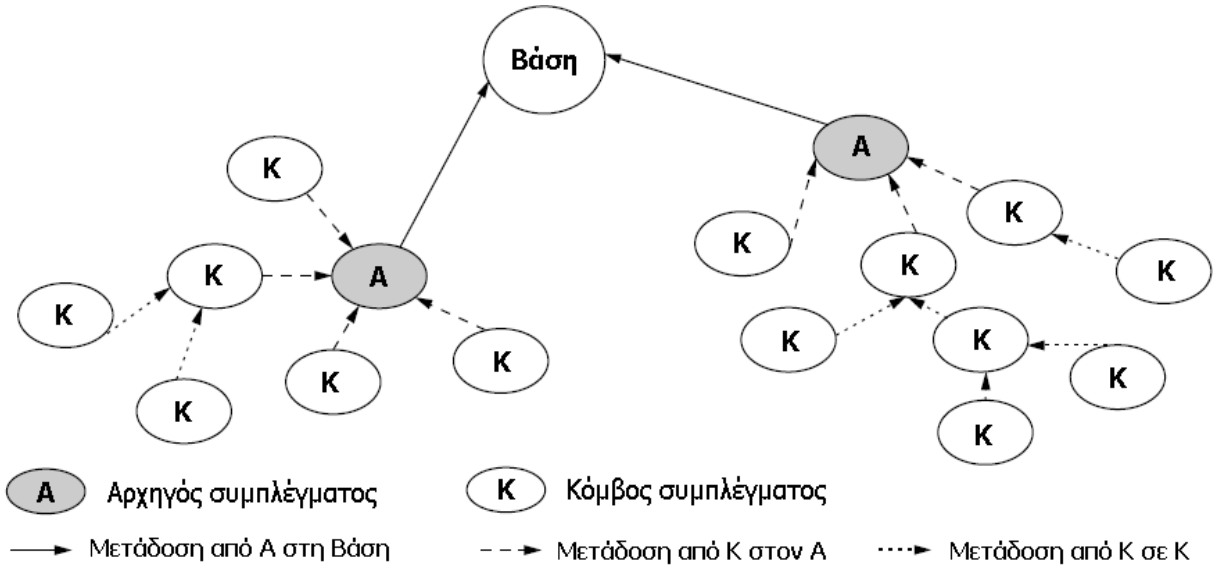
2.6.2.2 Διαίρεση του συστήματος σε συμπλέγματα

Γενικά, αναφέρεται ότι η ενέργεια που καταναλώνεται για την επικοινωνία είναι πολύ υψηλότερη (από 1000 έως 10000 φορές) από αυτή για την λειτουργία των αισθητήρων και τους υπολογισμούς-επεξεργασία δεδομένων [31], σε σημείο που οι δύο τελευταίες μπορούν να αμεληθούν και να θεωρηθεί η πρώτη ως η συνολική καταναλισκόμενη ενέργεια στα WSN. Επίσης, η ενέργεια αυτή αυξάνεται εκθετικά με την αύξηση της εμβέλειας μετάδοσης, πράγμα το οποίο μαζί με τα παραπάνω οδηγεί στο συμπέρασμα ότι μείωση της ποσότητας των δεδομένων και της εμβέλειας των επικοινωνιών μεταξύ των κόμβων αισθητήρων μπορεί να παρατείνει σημαντικά τη διάρκεια της ζωής ενός συστήματος με WSN.

Από την άλλη μεριά, ένα WSN συνήθως αποτελείται από ένα μεγάλο αριθμό κόμβων τοποθετημένων σε μια ευρεία περιοχή, με αποτέλεσμα η βάση, όπου συλλέγονται όλα τα δεδομένα του συστήματος, να είναι αρκετά μακριά από τους περισσότερους κόμβους. Κατά συνέπεια, διαίρεση του συστήματος σε *συμπλέγματα* (*clusters*) μπορεί να αντικαταστήσει την μεγάλης απόστασης άμεση μετάδοση μηνυμάτων (one-hop) μεταξύ ενός κόμβου και της βάσης με μια μικρής απόστασης αναμετάδοση (multi-hop) μεταξύ των κόμβων μέχρι τη βάση. Αυτό έχει ως αποτέλεσμα, επίσης, τη μείωση της καταναλισκόμενης ενέργειας και τα πλεονεκτήματα της εξισορρόπησης του φόρτου λειτουργίας-επιβάρυνσης των κόμβων και της εύκολης επεκτασιμότητας, με πρόσθεση κόμβων, ενός WSN. Ζητήματα που αφορούν, τώρα, αυτήν την προσέγγιση, αποτελούν η επιλογή των κόμβων που θα οδηγούν τα συμπλέγματα («αρχηγοί» - leaders) και ο τρόπος οργάνωσης των συμπλεγμάτων. Η αρχιτεκτονική αυτή χωρίζεται σε δύο κατηγορίες: *single-hop* και *multi-hop συμπλέγματα* [32], που διακρίνονται με βάση την απόσταση των «αρχηγών» με τους υπόλοιπους κόμβους, όπως φαίνεται στα *Σχήματα 2.25* και *2.26*, αντίστοιχα.

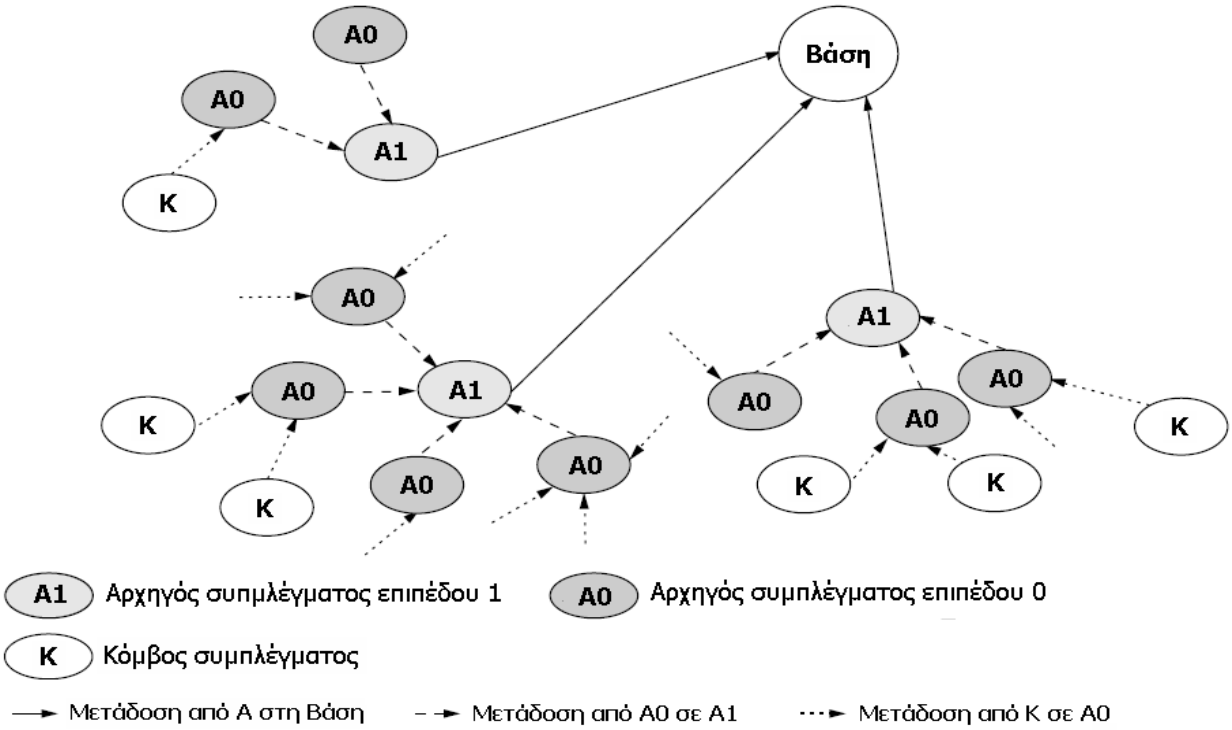


Εικόνα 2.25 Single-hop αρχιτεκτονική συμπλεγμάτων

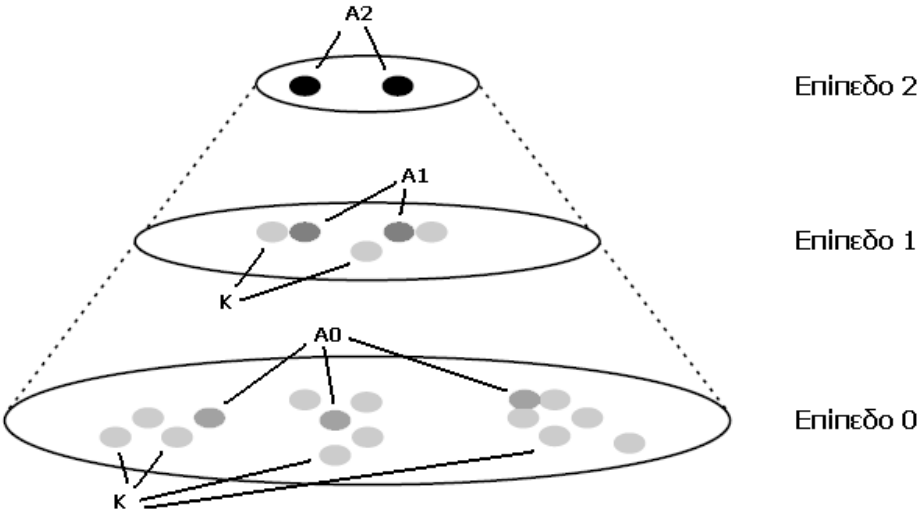


Εικόνα 2.26 Multi-hop αρχιτεκτονική συμπλεγμάτων.

Σύμφωνα, τώρα, με την ιεραρχική δομή των συμπλεγμάτων, οι αντίστοιχες αρχιτεκτονικές μπορούν να χωριστούν σε άλλες δύο κατηγορίες: *ενός επιπέδου* και *πολλαπλών επιπέδων αρχιτεκτονικές* συμπλεγμάτων. Τα *Σχήματα 2.25* και *2.26* αναπαριστούν ενός επιπέδου αρχιτεκτονικές. Μια πολλαπλών επιπέδων αρχιτεκτονική [33] φαίνεται στα *Σχήμα 2.27* και *2.28*.



Εικόνα 2.27 Δομή μιας πολλαπλών επιπέδων αρχιτεκτονικής συμπλεγμάτων.



Εικόνα 2.28 Επίπεδα μιας πολλαπλών επιπέδων αρχιτεκτονικής συμπλεγμάτων.

Γενικά, λαμβάνοντας υπόψη τα χαρακτηριστικά της κάθε αρχιτεκτονικής που βασίζεται σε συμπλέγματα, έχουν αναπτυχθεί διάφορες τεχνικές που αποσκοπούν στη μείωση της κατανάλωσης ενέργειας στο WSN και στη βέλτιστη οργάνωση του όλου δικτύου. Γνωστή, για παράδειγμα, είναι η τεχνική της «περιστροφής του αρχηγού» σε κάθε σύμπλεγμα (LEACH [22]) ώστε να υπάρχει μια ισορροπημένη επιβάρυνση λειτουργίας μεταξύ των κόμβων του συμπλέγματος, ενώ ο κατάλληλος υπολογισμός των αποστάσεων μεταξύ των κόμβων είναι και αυτός απαραίτητος κατά την επιδίωξη της μείωσης της καταναλισκόμενης ενέργειας. Φυσικά, σε WSN μεγάλης κλίμακας είναι απαραίτητη η χρήση μιας multi-hop, πολλαπλών επίπεδων αρχιτεκτονικής με σκοπό την μείωση της απόστασης επικοινωνίας μεταξύ των κόμβων αισθητήρων.

3. Η ΠΛΑΤΦΟΡΜΑ TMOTE SKY

3.1 Γενικά

Το Tmote Sky είναι μια μονάδα ασύρματης επικοινωνίας εξαιρετικά χαμηλής ενεργειακής κατανάλωσης, για χρήση σε δίκτυα αισθητήρων, εφαρμογές παρακολούθησης, και την ταχεία εφαρμογή πρωτοτύπων με σκοπό τόσο την ανεκτικότητα στο θόρυβο όσο και την ευκολία περαιτέρω ανάπτυξης και αξιοποίησης. Το Tmote Sky αξιοποιεί πρότυπα όπως το USB και το IEEE 802.15.4 για να επικοινωνεί άψογα με άλλες συσκευές. Χρησιμοποιώντας βιομηχανικά πρότυπα και ενσωματώνοντας μετρητές υγρασίας, θερμοκρασίας, αισθητήρες φωτός, και παρέχοντας ευέλικτες διασυνδέσεις με περιφερειακά, το Tmote Sky βοηθά στην υλοποίηση ενός μεγάλου φάσματος εφαρμογών δικτύου πλέγματος. Το Tmote Sky είναι ο αντικαταστάτης του επιτυχημένου Telos και περιλαμβάνει αυξημένες επιδόσεις, λειτουργικότητα, και επεκτασιμότητα. Με απόλυτη υποστήριξη από το TinyOS, το Tmote Sky αξιοποιεί τα νέα πρωτόκολλα ασύρματης δικτύωσης και το κίνημα του ανοιχτού λογισμικού. Το Tmote Sky είναι μέρος μιας σειράς πλατφορμών που αναπτύχθηκαν από το Πανεπιστήμιο του Berkeley, California και τα οποία διαθέτουν on-board αισθητήρες για να αυξηθεί η ισχύς, ενώ ταυτόχρονα μειώνεται το κόστος και το μέγεθος του πακέτου.

Εικόνα 3.1 Η ασύρματη μονάδα Tmote sky.

3.2 Βασικά γνωρίσματα

Τα κυριότερα γνωρίσματα και τεχνικά χαρακτηριστικά του Tmote Sky φαίνονται περιληπτικά παρακάτω:

- Ασύρματος πομποδέκτης 250kbps 2.4GHz IEEE 802.15.4 Chipcon,
- Διαλειτουργικότητα με άλλες συσκευές IEEE 802.15.4,
- Μικροελεγκτής 8MHz Texas Instruments MSP430 (10k RAM, 48k Flash),
- Ολοκληρωμένος ADC, DAC, Supply Voltage Supervisor και ελεγκτής DMA,
- Υλοποιημένη onboard κεραία με εμβέλεια 50m σε εσωτερικούς χώρους/125m σε εξωτερικούς,
- Ενσωματωμένοι αισθητήρες υγρασίας, θερμοκρασίας και φωτός,
- Εξαιρετικά χαμηλή κατανάλωση ρεύματος,
- Γρήγορη αφύπνιση (<6μs),
- Κωδικοποίηση και πιστοποίηση αυθεντικότητας στο στρώμα ζεύξης υλικού,
- Προγραμματισμός και συλλογή δεδομένων μέσω USB,
- Υποστήριξη επέκτασης 16pin και προαιρετικός συνδετήρας SMA για εξωτερική κεραία,
- Υποστήριξη λειτουργικού συστήματος TinyOS, πλέγμα εφαρμογής δικτύωσης και επικοινωνίας,
- Συμμορφώνεται με τους κανονισμούς FCC Part 15,
- Φιλικό προς το περιβάλλον - είναι σύμφωνη με τους κανονισμούς RoHS.

3.3 Τεχνικά χαρακτηριστικά

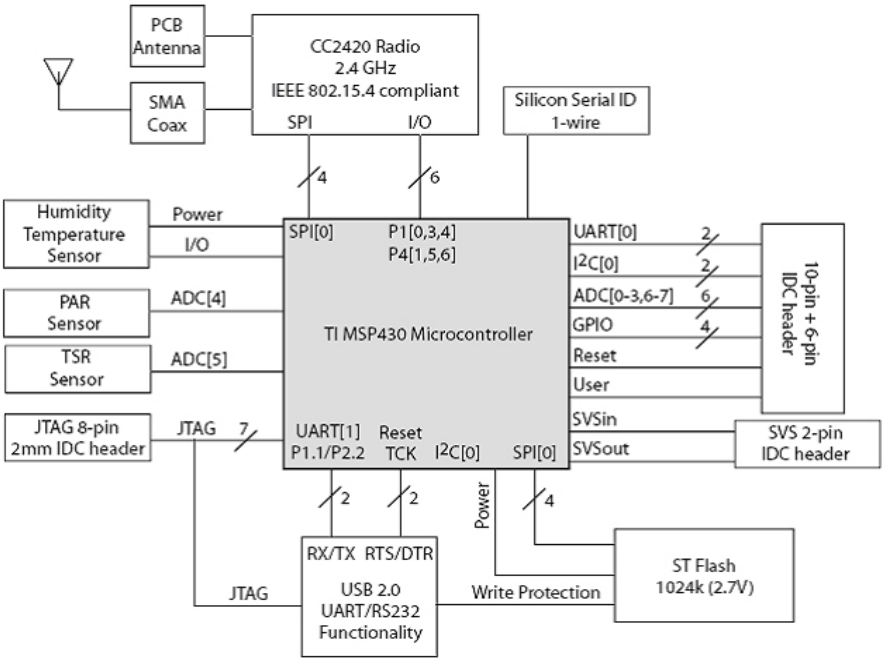
Πίνακας 3-1 Τεχνικά χαρακτηριστικά Tmote Sky

CPU	
Bus Speed	8 MHz
RAM	10 kB
Program Space	48 kB
External Flash	1024KB
Serial Communications	DIO,SPI,I2C,UART
Current (active w/ Radio on)	19 mA
Current (sleep)	5.1 uA
Startup Time	6 us
Voltage	1.8-3.6 V
Radio	
Frequency	2400-2483 MHz
Data rate	250 kbps
Output Power Startup Time	-25 to 0 dBm 580 us
Antenna Type	Inverted-F or SMA Coax
Humidity Sensor	
Humidity Accuracy	3.5% RH
Temperature Accuracy	0.5 °C
Sampling Rate	90 Hz

Η υλοποίηση και ανάπτυξη του tmote στηρίχθηκε σε τρεις βασικούς στόχους: την ακόμα χαμηλότερη κατανάλωση ενέργειας σε σχέση με τις προηγούμενες γενιές πλατφόρμων, την ευκολία χρήσης και την ευκολία περαιτέρω ανάπτυξης και πειραματισμού. Ο σχεδιασμός της μονάδας Tmote Sky στηρίζεται στην ακόλουθη βασική αρχή: Η μονάδα-κόμβος βρίσκεται σε αδράνεια στο σύνολο του χρόνου, αφυπνίζεται άμεσα με την ύπαρξη ενός συμβάντος, επεξεργάζεται το συμβάν και επιστρέφει σε αδράνεια. Η ολοκληρωμένη σχεδίαση του προσφέρει όμως κάτι παραπάνω από απλά χαμηλή κατανάλωση ενέργειας κατά τη λειτουργία του. Επιτρέπει στους σχεδιαστές να εκμεταλλευτούν την αυξημένη λειτουργικότητά του και να αναπτύξουν πιο εύρωστα συστήματα.

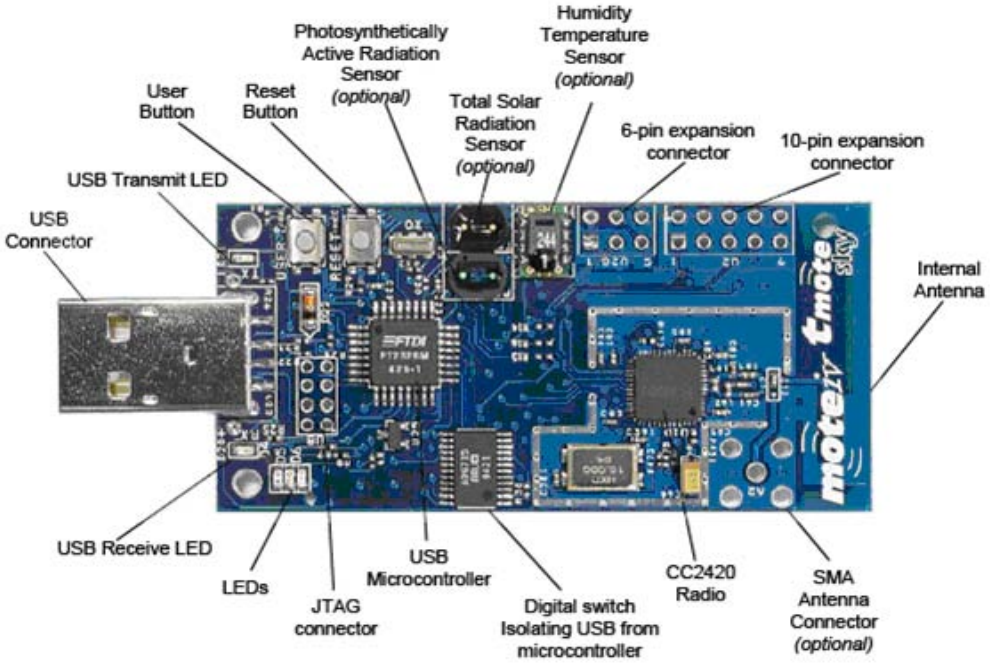
Η μονάδα tmote χρησιμοποιεί τον μικροελεγκτή MSP430, ο οποίος έχει χαμηλή κατανάλωση ενέργειας σε καταστάσεις αδράνειας και ενεργής λειτουργίας και λειτουργεί με ελάχιστη τάση 1.8 V. Η απαίτηση χαμηλών τιμών τάσης είναι σημαντική για την εξαγωγή όλης της ενέργειας από μια πηγή τάσης. Για παράδειγμα, οι μπαταρίες τύπου AA έχουν τάση αποκοπής στα 0.9V. Αν χρησιμοποιηθούν 2 μπαταρίες σε σειρά, η τάση αποκοπής του συστήματος είναι 1.8V, ακριβώς η ίδια με την ελάχιστη τάση που απαιτεί ο MSP430. Ο MSP430 μεταβαίνει από την κατάσταση αναμονής (standby 1μA) στην κατάσταση λειτουργίας το πολύ σε 6 μs. Επίσης, διαθέτει έναν ελεγκτή DMA (Direct Memory Access controller) προσφέροντας τη δυνατότητα μείωσης του φορτίου στον πυρήνα του μικροελεγκτή και της κατανάλωση ενέργειας, καθώς και αύξησης της απόδοσης.

3.4 Ολοκληρωμένη σχεδίαση

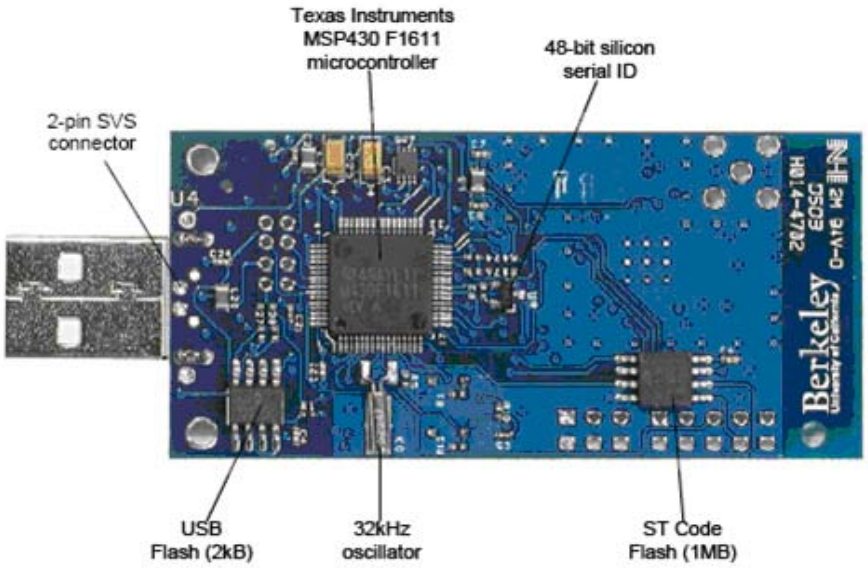


Εικόνα 3.2 Λειτουργικό μπλοκ διάγραμμα του Tmote

Το tmote-sky είναι μια μονάδα που συνδυάζει ενσωματωμένους αισθητήρες, δυνατότητα ασύρματης επικοινωνίας, κεραία, μικροελεγκτή και προγραμματιστικές δυνατότητες. Η ολοκληρωμένη σχεδίασή του παρέχει μια εύχρηστη μονάδα-κόμβο με αυξημένη στιβαρότητα. Τα τμήματα απ' τα οποία αποτελείται η μονάδα αυτή φαίνονται στην εικόνα 3.2.



Εικόνα 3.3 Εμπρόσθια όψη του Tmote sky

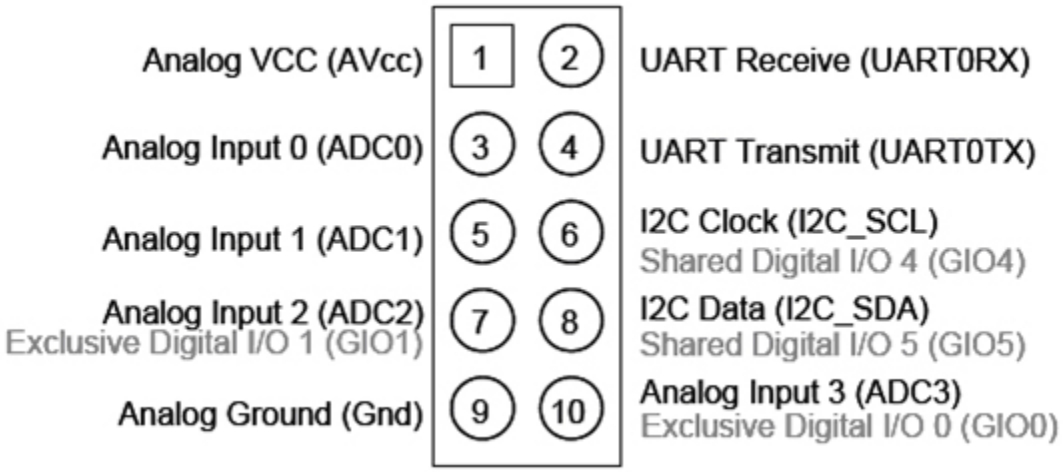


Εικόνα 3.4 Οπίσθια όψη του Tmote sky

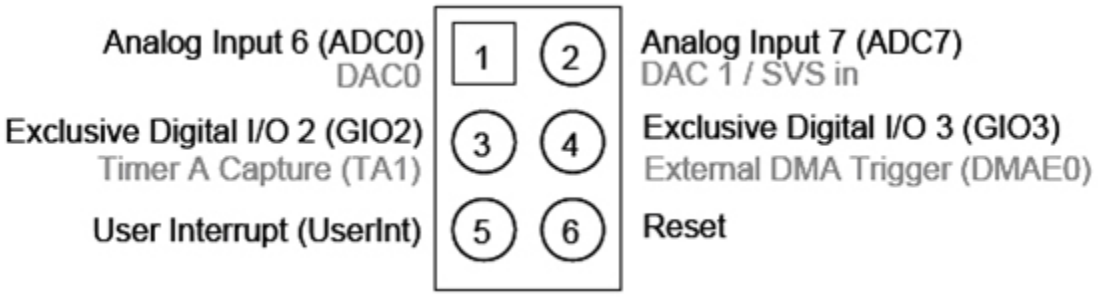
Το Tmote Sky χρησιμοποιεί μια ενσωματωμένη κεραία στα 2.4GHz , η οποία είναι μια μικροταινία σε σχήμα ανεστραμμένου F (Planar Inverted Folded Antenna – PIFA) και η οποία βρίσκεται τυπωμένη στην άκρη της πλακέτας, όπως φαίνεται και στην παραπάνω εικόνα (εμπρός όψη). Η κεραία αυτή επιτυγχάνει εμβέλεια 50 μέτρων σε εσωτερικούς χώρους και μπορεί να φτάσει μέχρι και τα 125 μέτρα σε ανοιχτούς. Μια προαιρετική SMA coax σύνδεση μπορεί να χρησιμοποιηθεί αντί της εσωτερικής κεραίας. Η ενσωμάτωση της κεραίας χαμηλώνει το συνολικό κόστος του mote αφού δεν απαιτείται άλλο ακριβό σύστημα εξωτερική κεραίας. Ο προγραμματισμός της μονάδας γίνεται μέσω σύνδεσης με τη θύρα USB ενός υπολογιστή. Για αυτό το λόγο ενσωματώνει πάνω του το κατάλληλο βύσμα USB που το απαλλάσσει από την ανάγκη χρήσης εξωτερικών καρτών διεπαφών.

3.5 Συνδετήρας επέκτασης

Το tmote έχει δυο συνδετήρες επέκτασης, έναν των 10 ακροδεκτών (10-pin IDC header) και έναν των 6 ακροδεκτών (6-pin IDC header) οι οποίοι μπορούν να διαμορφωθούν κατάλληλα ώστε να συνδεθούν επιπλέον συσκευές, όπως αναλογικοί αισθητήρες, οθόνες LCD και άλλες περιφερειακές συσκευές, οι οποίες και θα ελέγχονται από τη μονάδα. Ο συνδετήρας των 10 pin παρέχει τόσο ψηφιακές εισόδους και εξόδους όσο και αναλογικές. Ένας δεύτερος συνδετήρας των 6pin δίνει πρόσβαση σε επιπλέον δυνατότητες του sky mote. Οι λειτουργίες που υποστηρίζουν οι ακροδέκτες φαίνονται στα παρακάτω σχήματα :



Εικόνα 3.5 Συνδετήρας επέκτασης 10 θέσεων(pins)



Εικόνα 3.6 Συνδετήρας επέκτασης 6 θέσεων (pins).

3.6 Ισχύς

Η συσκευή λειτουργεί με δύο μπαταρίες τύπου AA οι οποίες μπορούν να χρησιμοποιηθούν για λειτουργία στο εύρος τάσης 2.1V με 3.6V DC. Εάν η συσκευή τοποθετηθεί στη θύρα USB για προγραμματισμό ή επικοινωνία με τον H/Y τότε μπορεί να τροφοδοτηθεί μέσω της θύρας αυτής. Στην περίπτωση αυτή η τάση τροφοδοσίας είναι 3V και δεν είναι απαραίτητη η χρήση μπαταρίας.

Ο 16-pin συνδετήρας επέκτασης μπορεί να παρέχει ενέργεια στη μονάδα. Οποιαδήποτε από τις συνδέσεις των ακροδεκτών μπαταρίας μπορεί επίσης να παρέχει ενέργεια για την στη μονάδα. Ποτέ δεν θα πρέπει η τάση εισόδου να υπερβαίνει τα 3.6V - κάτι τέτοιο μπορεί να βλάψει το μικροελεγκτή ή άλλα εξαρτήματα.

Πίνακας 3-2 Παροχή Τάσης

	MIN	NOM		UNIT
Supply voltage	2.1		3.6	V
Supply voltage during flash memory	2.7		3.6	V
Operating free air temperature	-40		85	^{o}C
Current Consumption: MCU on, Radio RX		21.8	23	mA
Current Consumption: MCU on, Radio TX		19.5	21	mA
Current Consumption: MCU on, Radio off		1800	2400	μA
Current Consumption: MCU idle, Radio off		54.5	1200	μA
Current Consumption: MCU standby		5.1	21	μA

Το tmote είναι η πρώτη μονάδα που περιλαμβάνει ‘προστασία εγγραφής’ στο υλικό (hardware write-protection). Όταν συνδέεται στη USB θύρα, η προστασία εγγραφής απενεργοποιείται και ο πρώτος τομέας της μνήμης flash μπορεί να εγγραφεί. Όταν λειτουργεί με μπαταρίες (χωρίς USB), ο τομέας αυτός έχει προστασία εγγραφής. Η προστασία εγγραφής είναι σημαντική σε συστήματα που μπορεί να αναπρογραμματιστούν ασύρματα. Και αυτό γιατί από τη στιγμή που θα έχει εγγραφεί στην προστατευμένη μνήμη μια ‘εικόνα’ ενός λειτουργικού προγράμματος θα υπάρχει πάντα ένας μηχανισμός επαναφοράς σε περίπτωση που χρειαστεί. Επίσης, κάθε στοιχείο-κομμάτι του υλικού είναι απομονωμένο. Η τροφοδοσία του κυκλώματος μπορεί να ανοίξει ή να κλείσει ανεξάρτητα από την υπόλοιπη πλατφόρμα. Η απομόνωση αυτή παρέχει μια στιβαρότητα έτσι ώστε σε περίπτωση αποτυχίας, τα στοιχεία που παρουσίασαν πρόβλημα να μπορούν να απενεργοποιηθούν ελαχιστοποιώντας την επίδραση τους στο κύκλωμα. Το κίνητρο για αυτή τη σχεδίαση ήρθε από την εμπειρία των πραγματικών δικτύων αισθητήρων στο Great Duck Island (GDI). Εκεί, ένας από τους κύριους λόγους αποτυχίας ενός κόμβου ήταν η ύπαρξη λάθους σε κάποιον αισθητήρα. Αφού το λάθος μπορεί να αναγνωριστεί από το λογισμικό, η δυνατότητα της διακοπής της τροφοδοσίας στο συγκεκριμένο τμήμα της πλακέτας θα μπορούσε να διασώσει το όλο σύστημα.

3.7 Μικρό-ελεγκτής

Η χαμηλής κατανάλωσης λειτουργία του Sky Tmote οφείλεται στην εξαιρετικά χαμηλής ισχύος μικροελεγκτή Texas Instruments MSP430 F1611 που διαθέτει 10kB RAM, 48KB flash, και 128B διαθέσιμα για αποθήκευση πληροφοριών. Αυτός ο 16-bit επεξεργαστής (ονομάζεται RISC) διαθέτει εξαιρετικά χαμηλές καταναλώσεις ενέργειας όταν είναι ενεργός καθώς και κατά τη διάρκεια του sleep mode που επιτρέπει στο Tmote να τρέχει για χρόνια με ένα ζευγάρι κοινές μπαταρίες AA. Ο MSP430 διαθέτει έναν εσωτερικό ψηφιακά ελεγχόμενο ταλαντωτή (DCO) που μπορεί να λειτουργήσει μέχρι 8MHz. Ο DCO μπορεί να ενεργοποιηθεί από την κατάσταση νάρκης σε 6μs, ωστόσο τα 292ns είναι μια τυπική τιμή για θερμοκρασία δωματίου. Όταν ο ελεγκτής είναι απενεργοποιημένος, το MSP430 λειτουργεί με ένα αιώνιο 32768Hz κρυσταλλικό ρολόι. Παρά το γεγονός ότι η DCO δε λειτουργεί σε μια σταθερή συχνότητα, αλλά αλλάζει καθώς αλλάζουν η τάση και η θερμοκρασία, μπορεί να βαθμονομηθεί με τη χρήση του ταλαντωτή των 32kHz.

Εκτός από το DCO, η MSP430 έχει 8 εξωτερικές θύρες ADC και 8 εσωτερικές θύρες ADC. Οι εσωτερικές θύρες μπορούν να χρησιμοποιηθούν για να διαβαστεί το εσωτερικό θερμίστορ ή για παρακολούθηση της τάσης της μπαταρίας. Μια ποικιλία από περιφερειακά είναι διαθέσιμα, όπως SPI, UART, ψηφιακές θύρες I / O, χρονόμετρο watchdog καθώς και άλλα χρονόμετρα. Η F1611 περιλαμβάνει επίσης μια δίθυρη12-bit DAC μονάδα, Supply Voltage Supervisor, και τρίθυρο ελεγκτή DMA (3 port).

3.8 Κατανάλωση ενέργειας

Πίνακας 3-3 Τυπικές συνθήκες λειτουργίας

Typical Operating Conditions	MIN	NOM	MAX	UNIT
Supply voltage during program execution	1.8		3.6	V
Supply voltage during flash memory programming	2.7		3.6	V
Operating free air temperature	-40		85	C
Low frequency crystal frequency		32768		kHz
Active current at Vcc = 3V, 1MHz		500	600	μA
Sleep current in LPM3 Vcc = 3V, 32.768kHz active		2.6	3	μA
Wake up from LPM3 (low power mode)			6	μS

Η κατανάλωση ενέργειας ενός αισθητήρα δεν αφορά μόνο τον μικροελεγκτή και/ή τον πομποδέκτη, αλλά επίσης και τα βοηθητικά στοιχεία από τα οποία αποτελείται. Λόγω της ολοκληρωμένης σχεδίασης του, ένα επιπλέον ρεύμα της τάξης των 3μA καταναλώνεται, σε κατάσταση αδράνειας, σε διακόπτες και ενδιάμεσες μνήμες για την προστασία από ροή του ρεύματος προς αποσυνδεδεμένα στοιχεία, και κυρίως το κύκλωμα της USB. Παρά το μικρό trade off, η συνολική κατανάλωση ενέργειας σε έναν κύκλο λειτουργίας (αφύπνιση, δειγματοληψία, μετάδοση και αδράνεια) είναι χαμηλή. Η κατανάλωση ενέργειας ισούται με το συνολικό χρόνο ενεργής λειτουργίας της μονάδας, πολλαπλασιασμένο με το ρεύμα που καταναλώνεται σε αυτό το χρόνο. Αφού το Tmote Sky έχει χαμηλότερη κατανάλωση ρεύματος, χαμηλότερο χρόνο αφύπνισης και χαμηλότερη τάση λειτουργίας, μπορεί να επιτύχει μεγαλύτερη διάρκεια ζωής από προηγούμενους σχεδιασμούς. Η χαμηλότερη κατανάλωση ενέργειας δε σημαίνει ότι το Tmote Sky εμφανίζει μικρότερη λειτουργικότητα, καθώς όλο και ισχυρότερα στοιχεία μικροεπεξεργαστών ενσωματώνονται στους μικροελεγκτές.

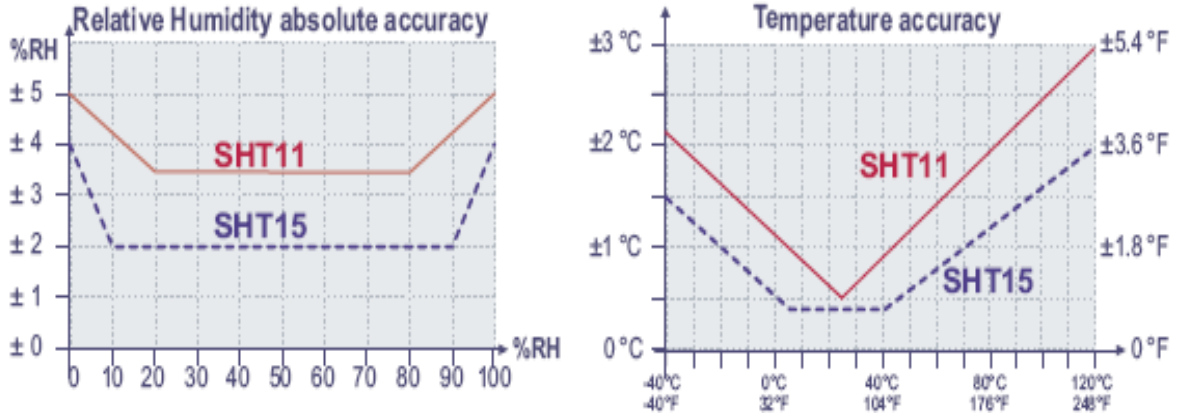
Η μονάδα Tmote Sky ενσωματώνει επίσης έναν ελεγκτή DMA ο οποίος λειτουργεί ενώ ο πυρήνας της μονάδας του μικροελεγκτή (MCU) βρίσκεται σε κατάσταση αδράνειας. Ο ελεγκτής DMA επιτρέπει σε εφαρμογές να εκτελούν διεργασίες όπως η δειγματοληψία του ADC, η έξοδος ενός σήματος στον ψηφιακό-αναλογικό μετατροπέα DAC καθώς και η ασύρματη μετάδοση δεδομένων χωρίς τη μεσολάβηση της MCU. Ο ελεγκτής DMA χρησιμοποιείται παραδοσιακά για την αύξηση της επίδοσης, αλλά στην περίπτωση των ενσωματωμένων

συστημάτων χαμηλής ισχύος, αυτό που κάνει στην πραγματικότητα είναι να χαμηλώνει το duty cycle, επιτρέποντας στον πυρήνα του μικροελεγκτή να παραμένει σε κατάσταση αδράνειας για περισσότερο χρόνο και να εξυπηρετεί λιγότερες διακοπές υλικού (hardware interrupts). Η βελτίωση των επιδόσεων λόγω του DMA μας επιτρέπουν να φτάσουμε σε ρυθμό δειγματοληψίας μέχρι και 200ksamples/sec σε σύγκριση με τη μέγιστη δυνατότητα των 10ksamples/sec σε μικροελεγκτές χωρίς DMA.

3.9 Αισθητήρες υγρασίας/θερμοκρασίας και φωτός

Πάνω στην πλακέτα του sky mote βρίσκονται ενσωματωμένοι αισθητήρες υγρασίας/ θερμοκρασίας και φωτός, οι οποίοι μπορούν να χρησιμοποιηθούν σε πλήθος εφαρμογών. Ο αισθητήρας υγρασίας/θερμοκρασίας, κατασκευασμένος από την εταιρεία Sensirion AG, παράγεται χρησιμοποιώντας μια CMOS επεξεργασία και συνδυάζεται με έναν 14bit αναλογικό/ψηφιακό μετατροπέα (A/D converter).

Οι αισθητήρες SHT11/SHT15 βαθμονομούνται και παράγουν μια ψηφιακή έξοδο. Οι συντελεστές βαθμονόμησης αποθηκεύονται στην EEPROM του αισθητήρα. Η διαφορά μεταξύ της SHT11 και SHT15 είναι ότι η SHT15 παράγει μεγαλύτερη ακρίβεια μετρήσεων, όπως φαίνεται στην Εικόνα 3.7.



Εικόνα 3.7 Ακρίβεια μετρήσεων αισθητήρων SHT11 και SHT15

Ο αισθητήρας φωτός χρησιμοποιεί φωτοδιόδους, στη συγκεκριμένη περίπτωση κατασκευασμένους από την Hamamatsu Corporation, οι οποίοι ‘αντιδρούν’ στην ακτινοβολία φωτός. Πρόκειται για το S1087 για ανίχνευση της φωτοσυνθετικά ενεργή ακτινοβολία και το S1087-01 για ανίχνευση ολόκληρου του ορατού φάσματος, συμπεριλαμβανομένου του υπερύθρου. Κάθε φωτοδίοδος με παρόμοια φυσικές διαστάσεις μπορεί να χρησιμοποιηθεί με το Tmote Sky.

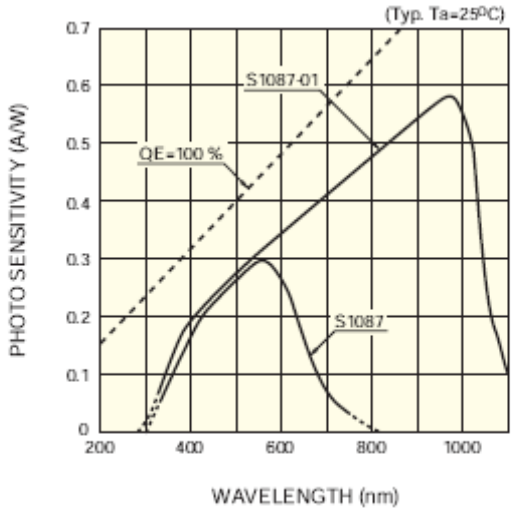
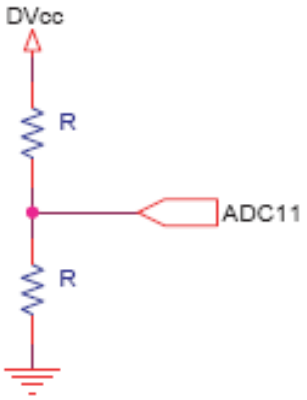


Figure 19 : Photo Sensitivity of the Light sensors on Tmote Sky(from Hamamatsu)

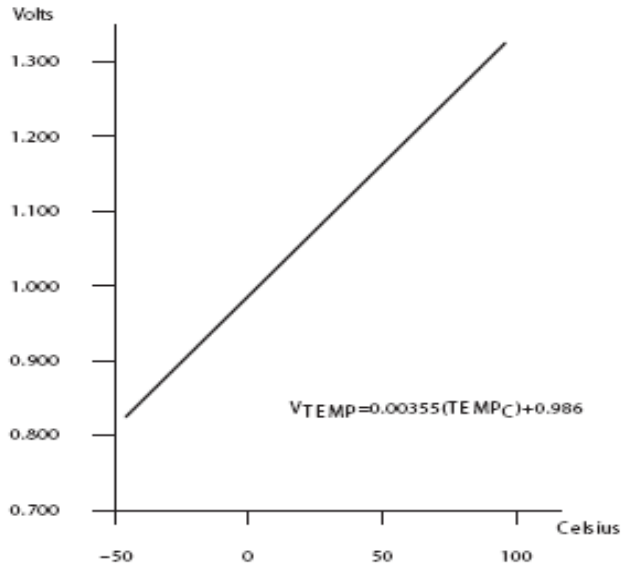
Εικόνα 3.8 Φώτο-ευαισθησία των φωτοδιόδων του Tmote Sky

Τέλος, ο μικροελεγκτής MPS430 διαθέτει και εσωτερικούς αισθητήρες θερμοκρασίας και τάσης οι οποίοι μπορούν να χρησιμοποιηθούν μέσω της διεπαφής ADC του μικροελεγκτή. Η θύρα τάσης (είσοδος 11) στον 12-bit ADC καταγράφει την έξοδο από έναν διαχωριστή τάσης.



Εικόνα 3.9 Διάγραμμα του κυκλώματος αισθητήρα θερμοκρασίας

Η είσοδος για τη θερμοκρασία είναι μια δίοδος θερμοκρασίας συνδεδεμένη στην εσωτερική θύρα 10 του ADC. Η τυπική απόκριση του αισθητήρα φωτός φαίνεται στην επόμενη εικόνα:



Εικόνα 3.10 Παράστασης τυπικής απόκλισης του εσωτερικού αισθητήρα φωτός.

Είναι ευνόητο ότι οι μετρήσεις όπως καταγράφονται από τις εξόδους του ADC είναι απλά ένα ενδιάμεσο βήμα για την απόκτηση χρήσιμης πληροφορίας σχετικής με δεδομένα αισθητήρων και είναι αναγκαία η μετατροπή των τιμών αυτών σε μορφή αναγνωρίσιμη στις γνωστές σε όλους SI μονάδες, σύμφωνα με τον πίνακα 3-4.

Πίνακας 3-4 Μονάδες μετρούμενων μεγεθών από Tmote Sky

ΥΠΟ ΜΕΤΡΗΣΗ ΜΕΓΕΘΟΣ	ΤΥΠΟΣ ΑΙΣΘΗΤΗΡΑ	ΜΟΝΑΔΕΣ ΣΤΟ SI
ΘΕΡΜΟΚΡΑΣΙΑ (Temperature)	Sensirion SHT11	^{o}C
ΥΓΡΑΣΙΑ (Humidity)	Sensirion SHT11	%RH
ΟΛΙΚΗ ΗΛΙΑΚΗ ΑΚΤΙΝΟΒΟΛΙΑ (TSR)	Hamamatsu TSR Light Sensor (S-1087)	lux
ΕΝΕΡΓΗ ΦΩΤΟΣΥΝΘΕΤΙΚΑ ΑΚΤΙΝΟΒΟΛΙΑ (PAR)	Hamamatsu PAR Light Sensor(S1087-01)	lux
ΤΑΣΗ ΤΡΟΦΟΔΟΣΙΑΣ	TI MSP430 Internal	Volts
ΕΣΩΤΕΡΙΚΗ ΘΕΡΜΟΚΡΑΣΙΑ	TI MSP430 Internal	^{o}C

Από μελέτη των [34],[35],[36],[37] προκύπτουν τα παρακάτω:

3.9.1 Θερμοκρασία

Οι αισθητήρες θερμοκρασίας και υγρασίας είναι τοποθετημένοι στο εξωτερικό του αισθητήρα Sensirion.

Για τη θερμοκρασία, το Oscilloscope επιστρέφει μία 14-bit τιμή που μετατρέπεται σε βαθμούς Κελσίου από τον ακόλουθο τύπο:

$$T = -39.60 + 0.01 * SOt$$

όπου SOt είναι η μη επεξεργασμένη τιμή της θερμοκρασίας.

3.9.2 Υγρασία

Η υγρασία είναι μία 14-bit τιμή και υπολογίζεται από τους παρακάτω τύπους :

$$Humidity_{linear} = -4 + 0.405 * SOrh + (-2.8 * 10 \wedge -6) * (SOrh \wedge 2)$$

όπου SOrh η μη επεξεργασμένη τιμή του αισθητήρα υγρασίας ο οποίος εξαρτάται από τη θερμοκρασία .

Χρησιμοποιώντας το παραπάνω αποτέλεσμα και την τιμή της θερμοκρασίας από την 3.9.1, με βάση τον ακόλουθο τύπο παίρνουμε την πραγματική τιμή της υγρασίας:

$$Humidity_{true} = (Tc - 25) * (0.01 + 0.00008 * SOrh) + humidity$$

όπου Tc η θερμοκρασία σε βαθμούς Κελσίου έτσι όπως υπολογίστηκε στο 3.9.1.

3.9.3 Εσωτερική Τάση

Ο αισθητήρας εσωτερικής τάσης χρησιμοποιεί το 12-bit ADC του μικροελεγκτή του Mote. Στη θύρα τάσης του ADC απεικονίζεται το αποτέλεσμα από ένα διαιρέτη τάσης. Άρα για τη μετατροπή σε φυσικές μονάδες χρησιμοποιούμε τον ακόλουθο τύπο:

$$Vcc = \frac{RawValue}{4096} * Vref * \frac{2R}{R}$$

όπου Vref = 1.5V η τάση των μπαταριών που χρησιμοποιεί το mote, RawValue η τιμή τάσης του ADC και ο αριθμός $4096 = 2^{12}$ γιατί έχουμε 12-bit ADC.

3.9.4 Εσωτερική Θερμοκρασία

Παρόμοια με την εσωτερική τάση, ο αισθητήρας εσωτερικής θερμοκρασίας είναι μη-βαθμονομημένος και δειγματοληπτείται με τη χρήση του 12-bit ADC που βρίσκεται στον 16-bit microcontroller. Αν και μη βαθμονομημένος ο ακόλουθος τύπος (εικόνα 3.10) λειτουργεί σωστά για αρκετές εφαρμογές, ωστόσο όμως κάποιες φορές το σφάλμα μπορεί να είναι αρκετά μεγάλο.

$$T = \frac{Vtemp - 0.986}{0.0035}$$

όπου Vtemp είναι η τιμή της τάσης που προέρχεται από 3.9.3 διαιρεμένη με το 2.Αυτό γιατί υπολογίζουμε την εσωτερική θερμοκρασία συναρτήσει της τιμής τάσης που εμφανίζεται στη γραμμή τάσης του ADC(το αποτέλεσμα που προέρχεται από το διαιρέτη τάσης που εξηγήσαμε προηγουμένως).

3.9.5 Ολική Ηλιακή Ακτινοβολία (TSR)

Οι TSR και PAR αισθητήρες μετρώνται με τον 12-bit ADC με Vref=1.5V. Οι φωτοδίοδοι δημιουργούν ρεύμα κατά μήκος μιας αντίστασης 100KOhm.

Το ρεύμα της φωτοδιόδου του αισθητήρα TSR σε lux προκύπτει από τον τύπο

S1087 lx = 0.625 * 1e6 * I * 1000

Όπου

$$I = \frac{V_{sensor}}{100,000}$$

(από V_{sensor}=I*R , με I=100KOhm)

με

$$V_{sensor} = \frac{RawValue}{4096} * Vref$$

3.9.6 Ενεργή Φωτοσυνθετικά Ακτινοβολία (PAR)

Το ρεύμα της φωτοδιόδου του PAR προκύπτει από τον τύπο:

S1087-01 lx = 0.769 * 1e5 * I * 1000

με I όμοια με το 3.9.5.

3.10 Ασύρματος πομποδέκτης

3.10.1 Εισαγωγή στους Ραδιοπομποδέκτες

Υπάρχουν δύο τύποι ραδιοπομπών χαμηλής ισχύος, χαμηλού ρυθμού δεδομένων: οι στενής ζώνης (narrowband) και οι ευρυζωνικοί (wideband). Αρκετοί narrowband πομποδέκτες παρέχουν πολύ γρήγορους χρόνους εκκίνησης (startup times) καθώς συγχρονίζονται από τον μικροελεγκτή (MCU) αλλά, έχουν απλά σχήματα διαμόρφωσης, δεν έχουν εξάπλωση κώδικα και είναι ευάλωτοι στο θόρυβο. Οι wideband πομποδέκτες έχουν την ανάγκη ελέγχου από υψηλής ταχύτητας ταλαντωτές. Τα βελτιωμένα σχήματα διαμόρφωσης που εμφανίζονται σε αυτούς τους πομποδέκτες, όπως είναι οι διαμορφώσεις DSSS και O-QPSK, παρέχουν στιβαρότητα στο σήμα απέναντι στο θόρυβο και την παρεμβολή. Οι narrowband ραδιοπομποδέκτες λειτουργούν συνήθως σε χαμηλότερες συχνότητες και με χαμηλούς ρυθμούς δεδομένων, σε αντίθεση με τους wideband που λειτουργούν συνήθως στη συχνότητα των 2.4GHz και προσφέρουν υψηλότερους ρυθμούς μετάδοσης. Η επιλογή του κατάλληλου πομποδέκτη στηρίζεται σε ορισμένα κριτήρια που ο σχεδιαστής ενός συστήματος πρέπει να λάβει υπόψη του, όπως είναι η επίδραση του θορύβου, η ευελιξία που διατίθεται στην τελική εφαρμογή, η ευκολία επικοινωνίας με άλλες συσκευές, η κατανάλωση ενέργειας και το διαθέσιμο εύρος ζώνης δεδομένων.

Παρουσιάζονται συνοπτικά στον παρακάτω πίνακα τα χαρακτηριστικά κοινών ραδιοπομποδεκτών. Κανένας από τους αναγραφόμενους δεν είναι γενικά ο καλύτερος. Η εκλογή του κατάλληλου πομποδέκτη πρέπει να βασίζεται κάθε φορά στις απαιτήσεις της εφαρμογής.

Πίνακας 3-5 Χαρακτηριστικά σύγχρονων πομποδεκτών (COTS radios, commercial -shelf) ιδανικών για δίκτυα ασύρματων αισθητήρων

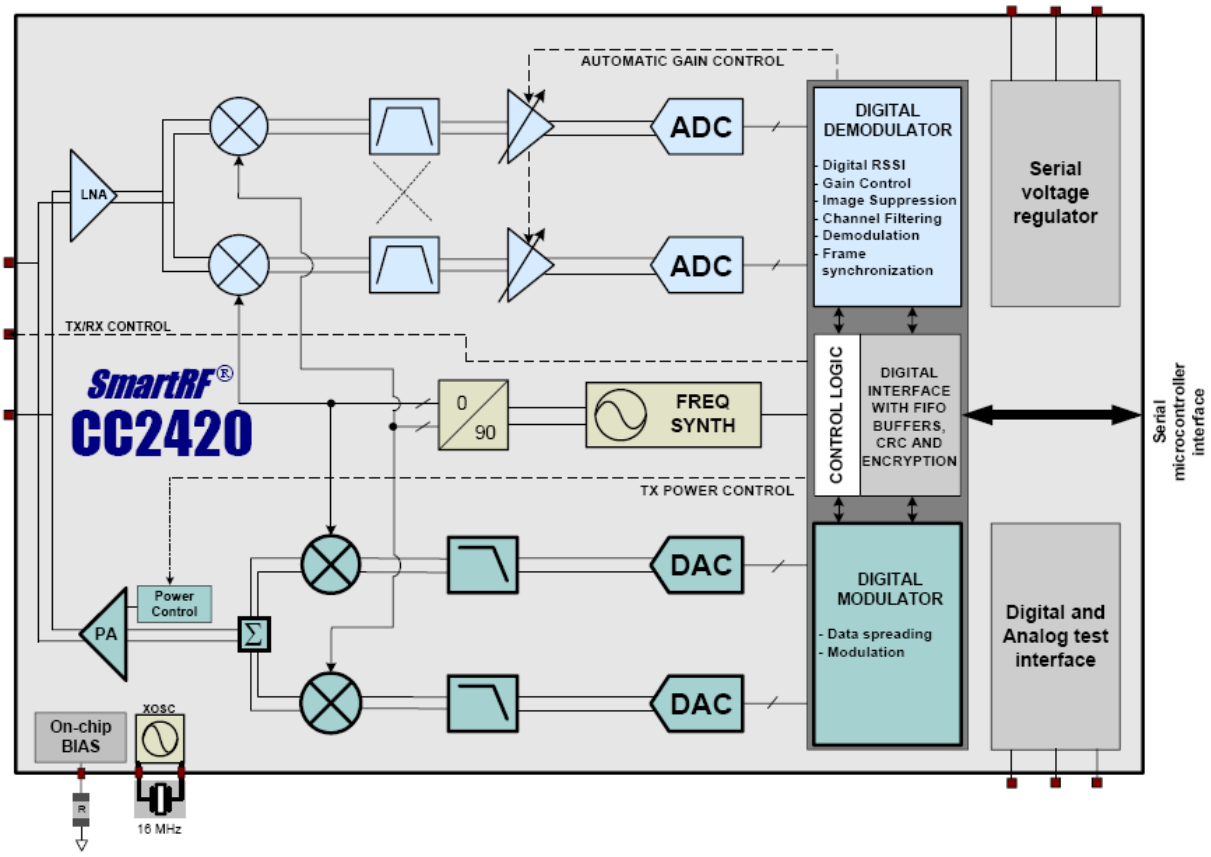
Type	**Narrowband**				**Wideband**		
Vendor Part no.	**RFM TR1000**	**Chipcon CC1000**	**Chipcon CC2400**	**Nordic nRF2401**	**Chipcon CC2420**	**Motorola MC13191/92**	**Zeevo ZV4002**
Max Data rate (kbps)	115.2	76.8	1000	1000	250	250	723.2
RX power (mA)	3.8	9.6	24	18 (25)	19.7	37(42)	65
TX power (mA/dBm)	12/ 1.5	16.5 / 10	19/0	13/0	17.4 / 0	34(30)/ 0	65/ 0
Powerdown power (μA)	1	1	1.5	0.4	1	1	140

Type	Narrowband				Wideband		
Vendor Part no.	**RFM TR1000**	**Chipcon CC1000**	**Chipcon CC2400**	**Nordic nRF2401**	**Chipcon CC2420**	**Motorola MC13191/92**	**Zeevo ZV4002**
Turn on time (ms)	0.02	2	1.13	3	0.58	20	*
Modulation	OOK/ASK	FSK	FSK,GFSK	GFSK	DSSS-O-QPSK	DSSS-O-QPSK	FHSS-GFSK
Packet detection	no	no	programmable	yes	yes	yes	yes
Address decoding	no	no	no	yes	yes	yes	yes
Encryption support	no	no	no	no	128-bit AES	no	128-bit SC
Error detection	no	no	yes	yes	yes	yes	yes
Error correction	no	no	no	no	yes	yes	yes
Acknowledgments	no	no	no	no	yes	yes	yes
Interface	bit	byte	packet/ byte	packet/byte	packet/byte	packet/ byte	packet
Buffering (bytes)	no	1	32	16	128	133	yes *
Time-sync	bit	SFD/ byte	SFD/ packet	packet	SFD	SFD	Bluetooth
Localization	RSSI	RSSI	RSSI	no	RSSI/LQI	RSSI/LQI	RSSI

3.10.2 Tmote Sky- Πομποδέκτης Chipcon CC2420

Η μονάδα tmote χρησιμοποιεί το πρότυπο IEEE 802.15.4 και υποστηρίζει το πρωτόκολλο ZigBee. Χρησιμοποιώντας έναν τυποποιημένο πομποδέκτη, το tmote μπορεί να επικοινωνήσει με οποιοδήποτε αριθμό συσκευών που μοιράζονται το ίδιο φυσικό στρώμα, συμπεριλαμβάνοντας και συσκευές άλλων κατασκευαστών. Το Tmote Sky χρησιμοποιεί τον πομποδέκτη Chipcon CC2420 (εικόνα 3.13) στα 2.4 GHz, έναν ευρυζωνικό πομποδέκτη με διαμόρφωση O-QPSK με DSSS στα 250Kbps. Ο υψηλότερος ρυθμός δεδομένων επιτρέπει μικρότερες περιόδους λειτουργίας μειώνοντας επιπλέον την κατανάλωση ενέργειας. Ο CC2420 είναι ένας πομποδέκτης με αυξημένη ευαισθησία και χαμηλή ισχύ λειτουργίας, ο οποίος παρέχει αξιόπιστη ασύρματη επικοινωνία. Η λειτουργία του ελέγχεται μέσω του TI MSP430 ενώ, και η ισχύς εξόδου μπορεί να προγραμματιστεί σύμφωνα με τις ανάγκες της εκάστοτε εφαρμογής.

Εικόνα 3.11 Πομποδέκτης Chipcon CC2420



Εικόνα 3.12 Block Διάγραμμα του πομποδέκτη Chipcon CC2420

Ο CC2420 παρέχει επίσης ένα σύνολο από επιταχυντές υλικού προς βελτίωση της απόδοσης. Αυτοί περιλαμβάνουν κρυπτογράφηση και επαλήθευση, υποστήριξη χειρισμού πακέτων, αυτόματες γνωστοποιήσεις (auto acknowledgments) και αποκρυπτογράφηση διευθύνσεων (address decoding). Απ' τη στιγμή όμως που οι επιταχυντές υλικού είναι ενσωματωμένοι στον πομποδέκτη αντί στον μικροελεγκτή, δεν μπορούν να χρησιμοποιηθούν για λειτουργίες γενικού σκοπού. Για παράδειγμα, ένα σύνολο δεδομένων μπορεί να είναι κρυπτογραφημένο και αποθηκευμένο σε μια μνήμη flash αλλά, από τη στιγμή που δε στέλνεται κάπου ασύρματα μέσω του πομπού, η μονάδα κρυπτογράφησης του υλικού του πομπού δεν μπορεί να χρησιμοποιηθεί.

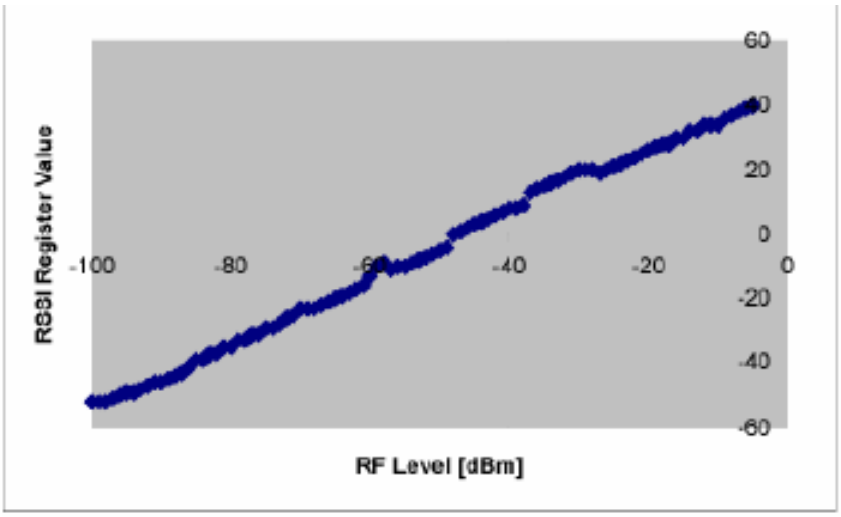
Οι τυπικές συνθήκες λειτουργίας του ασύρματου πομποδέκτη φαίνονται στον επόμενο πίνακα.

Πίνακας 3-6 Τυπικές συνθήκες λειτουργίας ασύρματου πομποδέκτη Chipcon CC2420.

	MIN	NOM	MAX	Μονάδα
Τάση λειτουργίας κατά την ασύρματη εκπομπή (Vreg on)	2.1		3.6	V
Θερμοκρασία λειτουργίας	-40		85	°C
Εύρος συχνοτήτων RF	2400		2484	MHz
Ρυθμός μετάδοσης δεδομένων	250		250	kbps
Ονομαστική ισχύς εξόδου	-3	0		dBm
Προγραμματιζόμενο εύρος ισχύς εξόδου		40		dBm
Ευαισθησία δέκτη	-90	-94		dBm
Κατανάλωση ρεύματος: ασύρματη μετάδοση σε 0 dBm		17.4		mA
Κατανάλωση ρεύματος: ασύρματη Λήψη		19.7		mA
Κατανάλωση ρεύματος: Radio on, ταλαντωτής on		365		μA
Κατανάλωση ρεύματος: κατάσταση αδράνειας, ταλαντωτής off		20		μA
Κατανάλωση ρεύματος: κατάσταση μη λειτουργίας, Vreg off			1	μA
Ρεύμα ρυθμιστή τάσης	13	20	29	μA
Χρόνος εκκίνησης ασύρματου ταλαντωτή		580	860	μs

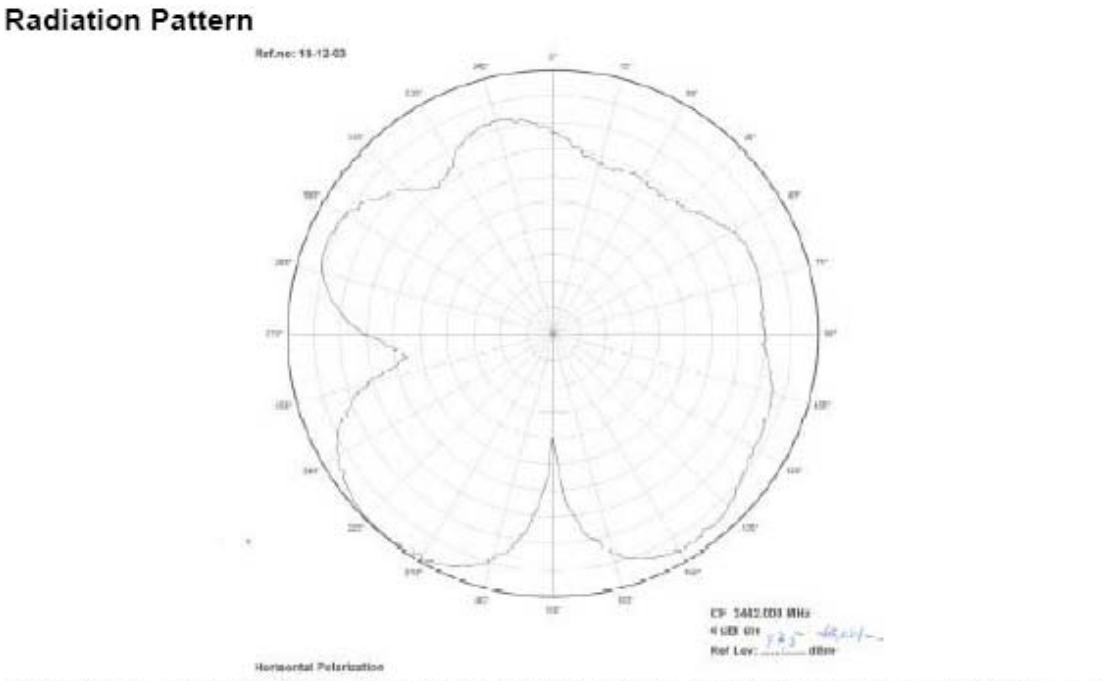
Το CC2420 παρέχει μια ψηφιακή ένδειξη ισχύος του σήματος (RSSI), που μπορεί να διαβαστεί οποιαδήποτε στιγμή. Επιπλέον, σε κάθε υποδοχή πακέτου, το CC2420 διαβάζει δείγματα από τα πρώτα οκτώ chips, υπολογίζει το ποσοστό σφάλματος (error rate), και παράγει μια ένδειξη της ποιότητας της σύνδεσης (LQI).

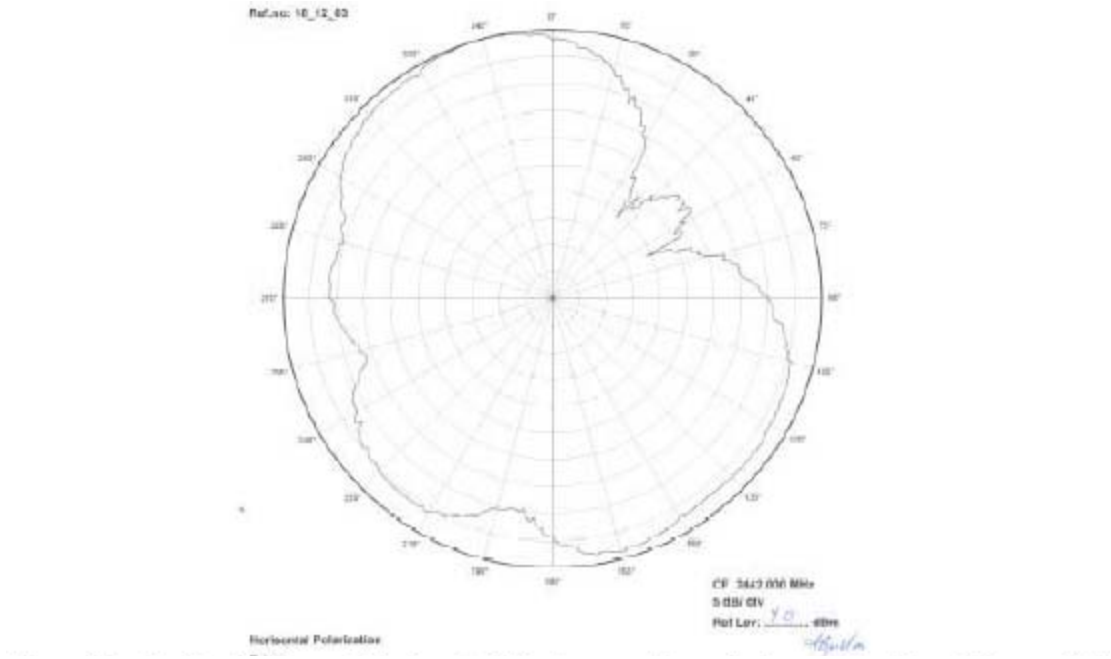
Μια χαρτογράφηση του RSSI στο RF επίπεδο σε dBm φαίνεται στο επόμενο Σχήμα.



Εικόνα 3.13 χαρτογράφηση του RSSI στο RF επίπεδο σε dBm

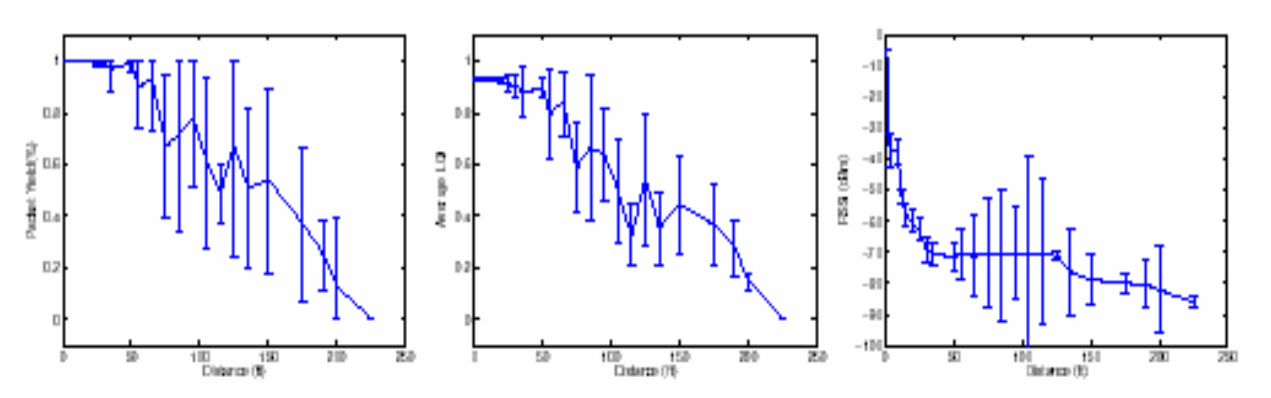
Η ενσωματωμένη στο tmote κεραία εμφανίζει κυρίως ομοιοκατευθυντικό διάγραμμα ακτινοβολίας.





Εικόνα 3.14 Διάγραμμα ακτινοβολίας της κεραίας του Tmote Sky

Από μετρήσεις που έχουν γίνει για την επίδραση της απόστασης στην ισχύ του ληφθέντος σήματος (Received Signal Strength Indicator, RSSI), στο ρυθμό επιτυχών πακέτων και στην ποιότητα της ζεύξης (Link Quality Indicator, LQI) προέκυψαν τα παρακάτω διαγράμματα για τις μέσες τιμές αυτών:



Εικόνα 3.15 Ποσοστό Ποσοστό ληφθέντων πακέτων (αριστερά), δείκτης ποιότητας ζεύξης (κέντρο) και ισχύς λαμβανομένου σήματος (δεξιά), σε εξωτερικό χώρο χρησιμοποιώντας τη μονάδα Tmote Sky και εσωτερική κεραία. Παρουσιάζεται ο μέσος όρος αποτελεσμάτων για 10 συνυπάρχοντες δέκτες.

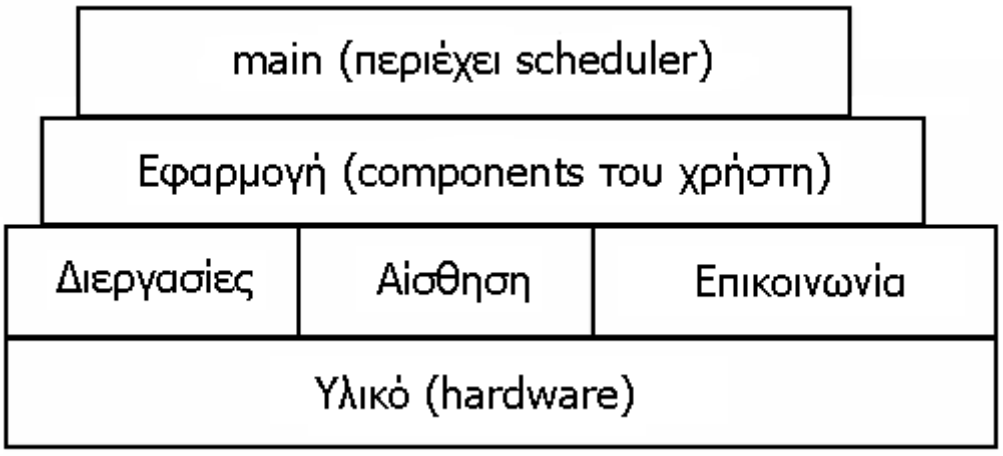
Ο δείκτης LQI καθιερώθηκε στο 802.15.4 και μετράει το σφάλμα στην ενδοδιαμόρφωση των επιτυχώς ληφθέντων πακέτων (πακέτα που πέρασαν τον CRC έλεγχο). Ο LQI του πομποδέκτη πλησιάζει σχηματικά το ρυθμό επιτυχών πακέτων. Ο RSSI ακολουθεί εκθετική μείωση καθώς ο ρυθμός επιτυχών πακέτων είναι υψηλός. Μετά από 18,29m (60 feet), το σήμα είναι πιο θορυβώδες και μειώνεται στην ελάχιστη ευαισθησία του πομποδέκτη. Επίσης, σε πειράματα που έγιναν σε ένα δίκτυο αποτελούμενο από τριάντα μονάδες-κόμβους Tmote Sky ώστε να μετρηθεί το πραγματικό εύρος ζώνης, προέκυψε ότι μια μονάδα (mote) είναι ικανή να χρησιμοποιήσει σχεδόν το μισό ενός πλήρους εύρους ζώνης δεδομένων του καναλιού ή 125kbps. Όταν και οι 30 κόμβοι μεταδίδουν όσο γρηγορότερα γίνεται, το Tmote Sky περιορίζεται σε ένα μέσο ρυθμό λήψης των 150kbps. Η απόδοση βέβαια, όπως είπαμε μπορεί να αυξηθεί χρησιμοποιώντας τον ελεγκτή DMA για την απευθείας μετάδοση δεδομένων χωρίς την μεσολάβηση της MCU καθώς και τη μείωση των συμβάντων διακοπής υλικού και υπερχείλισης της ενδιάμεσης μνήμης.

4. ΤΟ ΛΕΙΤΟΥΡΓΙΚΟ TINYOS

4.1 Εισαγωγή

Το TinyOS αναπτύχθηκε και εξελίχθηκε από το πανεπιστήμιο του Berkley και είναι λογισμικό ανοιχτού κώδικα[40],[41],[42]. Πρόκειται για ένα λειτουργικό σύστημα μικρό σε μέγεθος και οικονομικό ενεργειακά στη διαχείριση των αισθητήρων. Παρέχει ένα σύνολο από δομικές μονάδες λογισμικού από τις οποίες ο προγραμματιστής μπορεί να διαλέξει τα κατάλληλα συστατικά (components). Το μέγεθος τέτοιων αρχείων είναι τις τάξης των 200 bytes έτσι το μέγεθος του συνολικού προγράμματος παραμένει το ελάχιστο δυνατό. Το λειτουργικό σύστημα αυτό διαχειρίζεται τόσο το υλικό όσο και το ασύρματο δίκτυο εκτελώντας τις μετρήσεις των αισθητήρων, παίρνοντας αποφάσεις δρομολόγησης και ελέγχοντας την κατανάλωση ενέργειας.

Λόγω των περιορισμών πο υ επέβαλλε η φύση των αισθητήρων, μια νέα γλώσσα προγραμματισμού αναπτύχθηκε, η nesC, που υλοποιεί τις δομικές σχεδιαστικές ανάγκες και την επαναχρησιμοποίηση κώδικα του TinyOS για μικροσκοπικούς αισθητήρες. Για την υλοποίηση της επαναχρησιμοποίησης το TinyOS εφαρμόζει μια αρχιτεκτονική κατανομής σε επιμέρους συστατικά (component – based). Επιπρόσθετα για την βελτιστοποίηση της διαχείρισης ενέργειας χρησιμοποιεί ένα μοντέλο εκτέλεσης βασισμένο σε γεγονότα (event – based) όπου τα γεγονότα οδηγούν τα προγράμματα και οι σχετικοί πόροι αποδεσμεύονται με το πέρας της χρήσης τους. Επίσης, το TinyOS έχει βελτιστοποιηθεί ως προς την χρήση της μνήμης και προσφέρει υψηλή αποδοτικότητα ως προς την κατανάλωση ενέργειας. Επιπλέον, παρέχει διεπαφές (interfaces) μεταξύ των επιμέρους τμημάτων που το συνθέτουν και που ανήκουν σε γειτονικά στρώματα της αρχιτεκτονικής του. Ένα τέτοιο διάγραμμα στρωμάτων αυτής της αρχιτεκτονικής φαίνεται στην Εικόνα 4.1.



Εικόνα 4.1 Δομικό διάγραμμα στρωμάτων της αρχιτεκτονικής του TinyOS

Η προσαρμογή του TinyOS σε διάφορα είδη πλατφόρμων είναι εύκολη εξαιτίας της εύχρηστης και αφαιρούμενης διαστρωμάτωσης σε επίπεδο υλικού στον πυρήνα. Η διαστρωμάτωση αυτή επίσης διευκολύνει πολύ την ανάπτυξη των ασύρματων δικτύων . Το TinyOS ακόμα προσφέρει μια σειρά από εφαρμογές και εργαλεία ανάπτυξης όπως το Tossim (προσομοιωτής δικτύων του TinyOS) ,το deluge και το TinyDB που βοηθούν στην ανάπτυξη και ερεύνα των WSN. Λόγω της αποδοτικής σχεδίασης, της μεγάλης κοινότητας υποστήριξης και του ανοικτού κώδικα το TinyOS έγινε το πλέον διαδεδομένο λειτουργικό σύστημα για τα WSN.

4.2 Η γλώσσα προγραμματισμού NesC

4.2.1 Εισαγωγή

Τα πρώτα ασύρματα ενσωματωμένα συστήματα αισθητήρων έτρεχαν πάνω σε προσωπικούς υπολογιστές και χρησιμοποιούσαν κυρίως προγράμματα Linux. Όταν η ανάπτυξη αυτών των δικτύων πέρασε από τους μικροεπεξεργαστές (microprocessors) στους μικροελεγκτές (microcontrollers), το Linux είχε πάψει πια να αποτελεί την κατάλληλη επιλογή. Οι εφαρμογές των συστημάτων της εποχής εκείνης αναπτύσσονταν κυρίως σε τυπική γλώσσα C ή κατευθείαν σε γλώσσα assembly. Ο προγραμματισμός όμως σε αυτή τη γλώσσα είναι δύσκολο να αναλυθεί και επίσης μπορεί εύκολα να καταλήξει εκτός ελέγχου όταν η πολυπλοκότητα της εφαρμογής αυξηθεί. Σε πολυκλιμακωτά συστήματα το πρόβλημα μπορεί να επιληφθεί με αντικειμενοστραφή προγραμματισμό (object-oriented programming), ο οποίος καθιστά ευκολότερο το διαχωρισμό πολύπλοκων προγραμμάτων σε ανεξάρτητα, ευκολοσύνθετα συστατικά. Αλλά ο προγραμματισμός με προσανατολισμό στο αντικείμενο απαιτεί δυναμική παραχώρηση μνήμης και τείνει να απαιτεί περισσότερους προγραμματιστικούς πόρους, κάτι το οποίο τον κρίνει ακατάλληλο για ενσωματωμένα συστήματα (embedded systems). Η γλώσσα NesC, η οποία αναπτύχθηκε από ερευνητές του Πανεπιστημίου UC Berkeley, αντιπροσωπεύει ένα πολλά υποσχόμενο πεδίο για τους σχεδιαστές εφαρμογών. Είναι κατάλληλα σχεδιασμένη για ενσωματωμένα συστήματα δικτύων και υποστηρίζει ένα προγραμματιστικό μοντέλο που ενσωματώνει αντιδραστικότητα με το περιβάλλον, ταυτοχρονισμό και δυνατότητα επικοινωνίας.

Ένα βασικός άξονας επικέντρωσης της NesC είναι η ο ολιστικός σχεδιασμός συστημάτων. Οι εφαρμογές των μονάδων-αισθητήρων (motes) είναι βαθιά συνδεδεμένες στο υλικό και κάθε μονάδα τρέχει μια εφαρμογή κάθε φορά. Έτσι υπάρχει ένας αριθμός μοναδικών προκλήσεων που η γλώσσα NesC πρέπει να επιληφθεί:

- Οδήγηση από την αλληλεπίδραση με το περιβάλλον: Σε αντίθεση με τα παραδοσιακά συστήματα υπολογιστών, τα motes χρησιμοποιούνται για τη συλλογή δεδομένων και τον έλεγχο του τοπικού περιβάλλοντος, παρά για γενικής φύσεως υπολογισμούς. Αυτή η ιδιαιτερότητα οδηγεί σε δυο παρατηρήσεις: Η πρώτη είναι ότι τα motes είναι στοιχειωδώς οδηγούμενα από συμβάντα (event driven), αντιδρώντας σε αλλαγές του περιβάλλοντος (άφιξη ενός μηνύματος, ανάκτηση δεδομένων από αισθητήρες) παρά οδηγούμενα από διαδραστική (interactive) ή κατά δεσμίδες (batch) επεξεργασία. Η δεύτερη παρατήρηση είναι ότι η ‘άφιξη’ ενός συμβάντος ή η επεξεργασία δεδομένων είναι συντρέχουσες δραστηριότητες, απαιτώντας έτσι μια μεθόδευση για διαχείριση του ταυτοχρονισμού αυτού που επιλαμβάνεται ενδεχόμενων σφαλμάτων (bugs) όπως οι συνθήκες συναγωνισμού (race conditions).

- Περιορισμένοι πόροι: Οι μονάδες αυτές (motes) έχουν πολύ περιορισμένους φυσικούς πόρους, λόγω των ιδιαίτερων αναγκών για μικρό μέγεθος, χαμηλό κόστος και μικρή κατανάλωση ενέργειας. Οι περιορισμοί αυτοί δεν αναμένεται να εκλείψουν, καθώς τα οφέλη από την προσδοκία του νόμου του Moore θα οδηγεί συνεχώς σε μείωση του μεγέθους και του κόστους παρά σε αύξηση δυνατοτήτων-ικανοτήτων στο ίδιο μέγεθος.

- Αξιοπιστία: Αν και είναι αναμενόμενο οι μονάδες αυτές να παθαίνουν βλάβη λόγω σφαλμάτων υλικού, είναι έντονη η ανάγκη για εφαρμογές οι οποίες μπορεί να τρέχουν για μεγάλο χρονικό διάστημα. Για παράδειγμα, οι εφαρμογές παρακολούθησης περιβάλλοντος πρέπει να είναι ικανές να συλλέγουν δεδομένα χωρίς την ανθρώπινη παρέμβαση για μήνες κάθε φορά. Ένας σημαντικός στόχος είναι η μείωση των σφαλμάτων κατά τη διάρκεια της εκτέλεσης (run-time errors), καθώς δεν υπάρχει ουσιαστικός μηχανισμός ανάκαμψης σφαλμάτων εκτός από την αυτόματη επανεκκίνηση του συστήματος.

- Μικρές απαιτήσεις για λειτουργίες πραγματικού χρόνου: Παρόλο που υπάρχουν κάποιες εργασίες που είναι χρονικά κρίσιμες, όπως η διαχείριση της ασύρματης επικοινωνίας ή η σταθμοσκόπηση των αισθητήρων, σε γενικές γραμμές δεν υπάρχουν μεγάλες απαιτήσεις για πραγματικού χρόνου λειτουργίες. Μάλιστα η εμπειρία έχει δείξει ότι οι όποιοι χρονικοί περιορισμοί μπορούν να ικανοποιηθούν έχοντας απόλυτο έλεγχο της εφαρμογής και του λειτουργικού συστήματος και παράλληλα μειώνοντας την χρήση (utilization). Μια από τις λίγες κρίσιμες από πλευράς χρόνου λειτουργίες στα δίκτυα αισθητήρων είναι η ασύρματη επικοινωνία. Δεδομένου όμως της βασικής αναξιοπιστίας της ραδιοζεύξης γενικότερα δε θα χρειαστεί απαραίτητα να ικανοποιήσουμε δύσκολες απαιτήσεις στον τομέα αυτό.

Η γλώσσα NesC είναι προέκταση της C και αναμενόμενα, έχει όμοια σύνταξη, εντούτοις παρέχει τρία σημαντικά στοιχεία που τη διαφοροποιούν σημαντικά:

- Η γλώσσα NesC ορίζει ένα μοντέλο συστατικών που υποστηρίζει συστήματα οδηγούμενα από συμβάντα. Το μοντέλο αυτό παρέχει αμφίδρομες διεπαφές (interfaces) προς απλοποίηση της ροής των συμβάντων και επιτρέπει αποδοτική και ελαφριά υλοποίηση χωρίς τη δημιουργία εικονικών συναρτήσεων και δυναμικών στοιχείων.

- Παράλληλα ορίζει ένα απλό αλλά συγκεκριμένο μοντέλο ταυτοχρονισμού σε συνδυασμό με εκτεταμένη ανάλυση κατά τη μεταγλώττιση: ο μεταγλωττιστής (compiler) της NesC εντοπίζει τις πλειονότητα των περιπτώσεων ανταγωνισμού δεδομένων (data race) κατά τη διάρκεια της μεταγλώττισης. Αυτός ο συνδυασμός επιτρέπει τη δημιουργία σύγχρονων εφαρμογών που απαιτούν περιορισμένους πόρους.

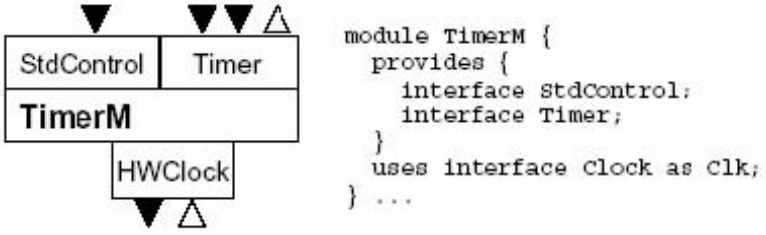
- Τέλος, η γλώσσα NesC παρέχει μια μοναδική ισορροπία μεταξύ της ανάλυσης προγράμματος, για τη βελτίωση της αξιοπιστίας και τη μείωση του κώδικα, και της δυνατότητας για δημιουργία ολοκληρωμένων εφαρμογών.

Επειδή η γλώσσα NesC έχει αποδειχθεί ότι είναι αποτελεσματική στην περίπτωση ανάπτυξης εφαρμογών για ασύρματα δίκτυα αισθητήρων, χρησιμοποιείται σαν την προγραμματιστική γλώσσα για το λειτουργικό σύστημα TinyOS, ένα μικρό λειτουργικό σύστημα για ασύρματα δίκτυα αισθητήρων που έχει υιοθετηθεί από ένα μεγάλο πλήθος ερευνητικών ομάδων σε όλο τον κόσμο. Σε εξέλιξη βρίσκονται έρευνες προς ανάπτυξη στο πρότυπο του TinyOS και άλλων γλωσσών προγραμματισμού αλλά μέχρι στιγμής η NesC είναι η μόνη γλώσσα που μπορεί να χρησιμοποιηθεί για την ανάπτυξη προγραμμάτων στο TinyOS[39].

4.2.2 Συστατικά και διεπαφές(Components and interfaces)

Η nesC είναι μια γλώσσα βασισμένη σε συστατικά (components). Τα συστατικά της nesC χρησιμοποιούν ένα καθαρά τοπικό namespace. Αυτό σημαίνει ότι εκτός από τη δήλωση των λειτουργιών που υλοποιεί, ένα συστατικό πρέπει επίσης να δηλώσει τις λειτουργίες που καλεί από άλλο συστατικό. Τα ονόματα που ένα συστατικό χρησιμοποιεί για να καλέσει αυτές τις λειτουργίες έχουν αυστηρά τοπική εμβέλεια: το όνομα στο οποίο αναφέρεται δεν είναι απαραίτητα το ίδιο με αυτό στο οποίο υλοποιείται η λειτουργία. Όταν ένα συστατικό Α δηλώνει ότι καλεί μια λειτουργία (συνάρτηση) Β, εισάγει ουσιαστικά το όνομα Α.Β ως ένα καθολικό namespace. Ένα διαφορετικό συστατικό, C, το οποίο καλεί τη λειτουργία Β εισάγει C.B ως ένα καθολικό namespace. Ακόμα κι αν και το Α και το C αναφέρονται στη λειτουργία Β. μπορεί να αναφέρονται σε απολύτως διαφορετικές υλοποιήσεις.

Κάθε συστατικό έχει μια προδιαγραφή, ένα μπλοκ κώδικα που δηλώνει τις λειτουργίες που παρέχει (implements) και τις λειτουργίες που χρησιμοποιεί (calls). Στην πράξη, τα συστατικά πολύ σπάνια δηλώνουν τις μεμονωμένες λειτουργίες στην προδιαγραφή τους. Αντί αυτού, η nesC έχει διεπαφές, οι οποίες είναι συλλογές των σχετικών λειτουργιών. Οι συστατικές προδιαγραφές είναι σχεδόν πάντα σε αναλογία προς τις διεπαφές. Αυτές οι διεπαφές είναι τα μοναδικά σημεία πρόσβασης στο συστατικό και είναι αμφίδρομες. Μια διεπαφή, γενικά, απεικονίζει μια υπηρεσία (όπως η αποστολή ενός μηνύματος) και προσδιορίζεται από ένα τύπο διεπαφής (interface type). Η εικόνα 4.2 δείχνει το συστατικό TimerM, τμήμα της υπηρεσίας χρονομετρητή (timer) του TinyOS, το οποίο παρέχει τις διεπαφές StdControl και Timer και χρησιμοποιεί τη διεπαφή Clock (όλες οι διεπαφές φαίνονται στην εικόνα 4.3). Το συσταατικό TimerM παρέχει τη λογική που δημιουργεί την αντιστοιχία μεταξύ ενός ρολογιού υλικού (hardware clock) και της έννοιας του χρονομετρητή (timer) στο TinyOS.



Εικόνα 4.2 και γραφική απεικόνιση του συστατικού TimerM

```
interface StdControl {
  command result_t init();
}

interface Timer {
  command result_t start(char type, uint32_t interval);
  command result_t stop();
  event result_t fired();
}

interface Clock {
  command result_t setRate(char interval, char scale);
  event result_t fire();
}

interface Send {
  command result_t send(TOS_Msg *msg, uint16_t length);
  event result_t sendDone(TOS_Msg *msg, result_t success);
}

interface ADC {
  command result_t getData();
  event result_t dataReady(uint16_t data);
}
```

Εικόνα 4.3 Μερικοί τύποι διεπαφών

Η σύνδεση των παροχών και των χρηστών από κοινού καλείται καλωδίωση (wiring). γιατί ο κώδικας χωρίζεται στα συστατικά, ιδιαίτερες μονάδες λειτουργίας τα οποία «καλωδιώνονται για να λειτουργήσει μια εφαρμογή.

Ένα συστατικό μπορεί μόνο να παραπέμψει μεταβλητές από το δικό του τοπικό namespace και δε μπορεί να ονομάζει μεταβλητές σε ένα άλλο συστατικό. Εντούτοις, μπορεί να δηλώσει ότι χρησιμοποιεί μια λειτουργία που καθορίζεται από άλλο συστατικό ή ότι παρέχει μια λειτουργία που ένα άλλο συστατικό μπορεί να καλέσει.

Οι διεπαφές στη NesC είναι, όπως είπαμε, αμφίδρομες. Μια διεπαφή δηλώνει ένα σύνολο από συναρτήσεις, οι οποίες καλούνται εντολές (commands), τις οποίες ο πάροχος μιας διεπαφής πρέπει να υλοποιήσει, καθώς και ένα σύνολο συναρτήσεων, ονομαζόμενων συμβάντα (events), τις οποίες ο χρήστης μιας διεπαφής πρέπει να υλοποιήσει. Για να καλέσει κάποιο συστατικό τις εντολές σε ένα interface, πρέπει να υλοποιήσει τα συμβάντα αυτής της διεπαφής. Ένα συστατικό από μόνο του μπορεί να χρησιμοποιεί και να παρέχει περισσότερες από μια διεπαφές καθώς και πολλαπλά στιγμιότυπα της ίδιας διεπαφής. Για παράδειγμα, η διεπαφή Timer (Εικόνα 4.3) ορίζει τις εντολές start και stop και το συμβάν fired. Στην Εικόνα 4.2, οι παρεχόμενες διεπαφές είναι αυτές που φαίνονται πάνω από το συστατικό TimerM ενώ, αυτές που χρησιμοποιούνται παριστάνονται κάτω από αυτό. Τα ‘βέλη’ που δείχνουν προς τα κάτω απεικονίζουν τις εντολές, ενώ αυτά που δείχνουν προς τα πάνω απεικονίζουν τα συμβάντα. Παρόλο που η ίδια αλληλεπίδραση μεταξύ του timer και του πελάτη (client) θα μπορούσε να πραγματοποιηθεί μέσω δυο ξεχωριστών διεπαφών (μια για τις εντολές start και stop και μια για το συμβάν fired), η ομαδοποίηση αυτών των εντολών και συμβάντων στην ίδια διεπαφή κάνει τον προσδιορισμό πιο ξεκάθαρο και βοηθάει στην αποφυγή λαθών κατά τη διασύνδεση των συστατικών μεταξύ τους. Οι εκτελούμενες σε δυο φάσεις λειτουργίες μπορούν εύκολα να μοντελοποιηθούν τοποθετώντας τις αιτούμενες εντολές και τα συμβάντα ανταπόκρισης στην ίδια διεπαφή. Στην Εικόνα 4.3 βλέπουμε δυο τέτοιες περιπτώσεις. Η διεπαφή send έχει την εντολή send και το συμβάν sendDone μέσα στο διαχωρισμένο σε φάσεις πακέτο send.

Ο διαχωρισμός αυτός στους ορισμούς των τύπων των διεπαφών από τη χρήση τους στα συστατικά προωθεί τον ορισμό πρότυπων διεπαφών, κάνοντας τα συστατικά (components) πιο ευέλικτα και επαναχρησιμοποιήσιμα.. Αξίζει να σημειώσουμε ότι ένα συστατικό μπορεί να παρέχει και να χρησιμοποιεί τον ίδιο τύπο διεπαφής ή να παρέχει την ίδια διεπαφή περισσότερες από μια φορές. Σε αυτές τις περιπτώσεις το συστατικό πρέπει να δώσει σε κάθε στιγμιότυπο της διεπαφής (interface instance) μια ξεχωριστή ονομασία χρησιμοποιώντας την χαρακτηριστική λέξη 'as', όπως φαίνεται και στο παράδειγμα της Εικόνας 4.2 για το Clk. Τέλος, ένα σημαντικό αλλά λεπτό σημείο είναι ότι οι αμφίδρομες διεπαφές μπορούν να υποστηρίξουν πολύ εύκολα τις διακοπές υλικού (hardware interrupts). Αντιθέτως, οι μονοσήμαντες διεπαφές που βασίζονται σε κλήσεις διαδικασιών υπαγορεύουν τη σταθμοσκόπηση υλικού (hardware polling) ή τη χρήση δυο ξεχωριστών διεπαφών για τις λειτουργίες υλικού και τις αντίστοιχες διακοπές (interrupts).

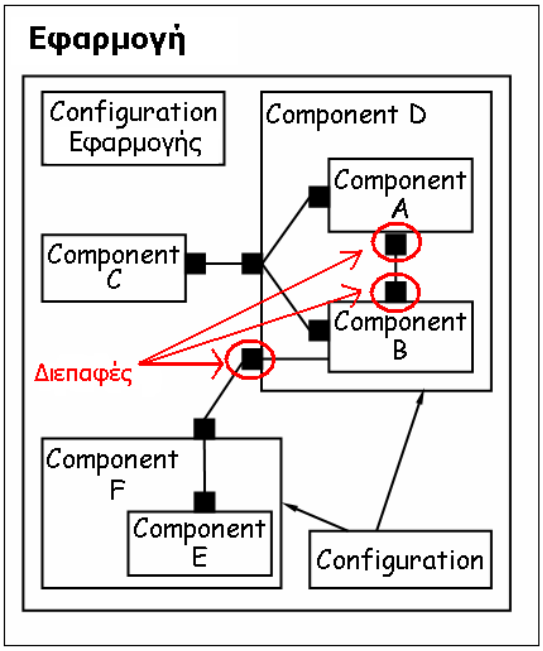
4.2.2.1 Παραμετροθετημένες διεπαφές (Parameterized interfaces)

Στη NesC γίνεται ευρεία χρήση των παραμετροθετημένων διεπαφών. Μια παραμετροθετημένη διεπαφή (parameterized interface) επιτρέπει σε ένα συστατικό του συστήματος να παρέχει πολλαπλές υποστάσεις μιας διεπαφής, οι οποίες παίρνουν μια συγκεκριμένη τιμή παραμέτρου κατά τη διάρκεια της μεταγλώττισης. Ένα συστατικό ορίζει μια λίστα παραμέτρων η οποία δημιουργεί και μια ξεχωριστή διεπαφή για κάθε πλειάδα τιμών παραμέτρων. Οι παραμετροθετημένες διεπαφές χρησιμοποιούνται για τη μοντελοποίηση των Active Messages AM (Ενεργών Μηνυμάτων) του TinyOS. Στα Active Messages, τα πακέτα περιέχουν έναν αναγνωριστικό αριθμό ο οποίος προσδιορίζει ποιος χειριστής συμβάντων υλικού πρέπει να εκτελεστεί. Η διασύνδεση μιας parameterized διεπαφής πρέπει να ορίζει μια συγκεκριμένη διεπαφή μέσω μιας σταθεράς (τιμής).

Σε ένα module, οι υλοποιήσιμες εντολές και τα συμβάντα μιας parameterized διεπαφής λαμβάνουν επιπλέον παραμέτρους που προσδιορίζουν την επιλεγμένη διεπαφή. Έτσι, για παράδειγμα, μπορούμε σε μια εφαρμογή να χρησιμοποιήσουμε πολλαπλούς χρονομετρητές (timers), καθένας από τους οποίους μπορεί να διαχειριστεί ανεξάρτητα. Μπορεί δηλαδή σε μια εφαρμογή ένα συστατικό να χρειάζεται έναν timer που θα παίρνει τιμές από έναν αισθητήρα κάθε δευτερόλεπτο, ενώ παράλληλα ένα άλλο συστατικό θα θέλει έναν timer ο οποίος θα χρονομετρεί με διαφορετικό ρυθμό ώστε να διαχειρίζεται την ασύρματη μετάδοση. Συνδέοντας (wiring) τη διεπαφή Timer καθένας από αυτά τα συστατικά σε διαφορετική υπόσταση της διεπαφής Timer, που παρέχεται από τον TimerC, μπορούμε να δώσουμε σε κάθε συστατικό το δικό του 'προσωπικό' Timer.

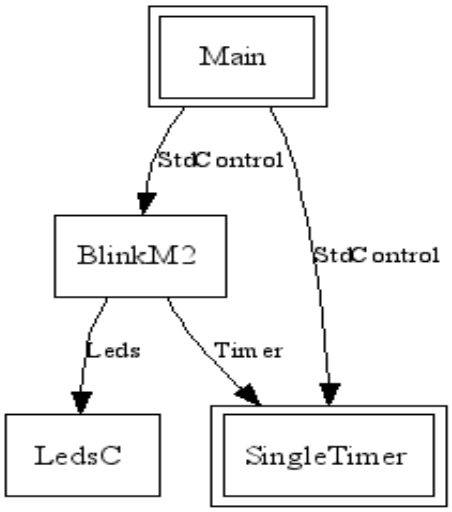
4.2.3 Υλοποίηση συστατικών-τμημάτων

Υπάρχουν δύο είδη συστατικών στη γλώσσα NesC: Τα modules και τα configurations. Τα modules παρέχουν τον κώδικα της εφαρμογής, υλοποιώντας μια ή περισσότερες διεπαφές. Τα configurations χρησιμοποιούνται για να συγκεντρώσουν τα υπόλοιπα συστατικά μαζί, συνδέοντας διεπαφές που χρησιμοποιούνται από συστατικά με διεπαφές που παρέχονται από άλλα συστατικά. Αυτή η διαδικασία ονομάζεται διασύνδεση ή wiring. Κάθε εφαρμογή περιγράφεται από μια διάρθρωση πάνω επιπέδου (top-level configuration) η οποία διασυνδέει τα συστατικά εσωτερικά (Εικόνα 4.4).



Εικόνα 4.4 Δομή μιας εφαρμογής

Το σώμα ενός module είναι γραμμένο σε κώδικα NesC. Μια εντολή (command) ή συμβάν (event) f μέσα σε μια διεπαφή i ορίζεται ως i.f . Η κλήση (call) μιας εντολής γίνεται όπως και η κλήση μιας απλής συνάρτησης προτάσσοντας όμως την κωδική λέξη 'call' . Η χρήση της call γίνεται πρακτικά όταν ένα συστατικό-πελάτης θέλει να εκτελέσει μια λειτουργία που παρέχεται από ένα συστατικό-εξυπηρετητή. Όμοια, η σηματοδοσία ενός συμβάντος (event signal) γίνεται όπως η κλήση μιας συνάρτησης βάζοντας το πρόθεμα 'signal'. Στην περίπτωση αυτή το 'signal' χρησιμοποιείται όταν το συστατικό-εξυπηρετητής θέλει να καλέσει μια λειτουργία μέσα στο συστατικό-πελάτη. Το συστατικό-πελάτης είναι υπεύθυνο στο να υλοποιήσει μια λειτουργία χειρισμού συμβάντος (event handling function) ώστε το συστατικό–εξυπηρετητής να μπορέσει να την καλέσει. Με αυτό τον τρόπο ο πελάτης είναι στατικά συνδεδεμένος με τον εξυπηρετητή, όπως και το αντίστροφο, ικανοποιώντας έτσι και την απαίτηση σχεδιασμού της NesC: όλα είναι στατικά. Ο προσδιορισμός μιας εντολής ή ενός συμβάντος με την ονομασία i.f γίνεται με το πρόθεμα command ή event αντίστοιχα. Αυτός ο υπομνηματισμός γίνεται για τη βελτίωση της σαφήνειας του κώδικα.



Εικόνα 4.5 Διάγραμμα Διασύνδεσης

Η διεπαφή ενός συστατικού μπορεί να διασυνδεθεί (wiring) περισσότερες από μια φορές. Σαν αποτέλεσμα, ένα αυθαίρετο πλήθος κλήσεων συγκεκριμένων εντολών μπορεί να διασυνδεθεί σε μια μόνο υλοποίηση της εντολής (“fan-in”), και επίσης μια απλή κλήση εντολής μπορεί να είναι συνδεδεμένη με ένα αυθαίρετο πλήθος υλοποιήσεων της εντολής (“fan-out”). Στη δεύτερη περίπτωση, οι πολλαπλές απαντήσεις από όλες τις κλήσεις πρέπει να συνδυαστούν σε μια. Αυτό γίνεται με το συνδυασμό των τιμών result_t με τη λογική πρόσθεση τους (logical AND). Ο τύπος απάντησης result_t επιστρέφει τις τιμές SUCCESS (1) και FAIL (0). Οπότε, με τη λογική πρόσθεσή τους, αν κάποια τιμή προκύψει FAIL, τότε αυτός που κάνει την κλήση βλέπει FAIL σαν τελική απάντηση.

4.2.4 Η λειτουργία split-face

Επειδή οι κόμβοι αισθητήρων έχουν ένα μεγάλο εύρος ικανοτήτων υλικού, ένας από τους στόχους του TinyOS είναι να υπάρξει ένα εύκαμπτο όριο υλικού/λογισμικού. Μια εφαρμογή που κρυπτογραφεί τα πακέτα πρέπει να είναι σε θέση να χρησιμοποιεί εναλλακτικές εφαρμογές υλικού ή λογισμικού χρήσης. Το υλικό, εντούτοις, είναι σχεδόν πάντα split-face παρά blocking. Είναι split-face στο ότι η ολοκλήρωση ενός αιτήματος είναι μια επανάκληση (callback). Παραδείγματος χάριν, για να γίνει μία ανάγνωση του αισθητήρα με τον αναλογικό σε ψηφιακό μετατροπέα (ADC), το λογισμικό γράφει σε μερικούς καταλόγους διαμόρφωσης να αρχίσει μία λήψη δείγματος από τον αισθητήρα. Όταν το δείγμα του ADC ολοκληρωθεί, τα ζητήματα υλικού διακόπτουν, και το λογισμικό διαβάζει την τιμή από έναν κατάλογο στοιχείων.

Το TinyOS επομένως υιοθετεί την εξής μέθοδο. Παρά να τα καταστήσει όλα σύγχρονα μέσω νημάτων(threads), διαδικασίες που είναι split-face στο υλικό είναι split-face και στο λογισμικό επίσης. Αυτό σημαίνει ότι πολλές κοινές διαδικασίες, όπως η δειγματοληψία των αισθητήρων και η αποστολή των πακέτων, είναι split-face. Ένα σημαντικό χαρακτηριστικό των διεπαφών split-face είναι ότι είναι αμφίδρομες: υπάρχει ένα downcall για να αρχίσει η λειτουργία, και ένα upcall που δηλώνει ότι η λειτουργία ολοκληρώθηκε. Σε nesC, downcalls είναι γενικά εντολές,

ενώ upcalls είναι γεγονότα(events). Μια διεπαφή διευκρινίζει και τις δύο πλευρές αυτής της σχέσης. Παραδείγματος χάριν, αυτή είναι η βασική διεπαφή TinyOS αποστολής πακέτων, Send:

```
interface Send {
  command error_t send(message_t* msg, uint8_t len);
  event void sendDone(message_t* msg, error_t error);

  command error_t cancel(message_t* msg);
  command void* getPayload(message_t* msg);
  command uint8_t maxPayloadLength(message_t* msg);
}
```

Εικόνα 4.6 Η split-fase διεπαφή Send

Εάν ένα συστατικό παρέχει ή χρησιμοποιεί τη Send διεπαφή καθορίζει ποια πλευρά της λειτουργίας split-face αντιπροσωπεύει. Ένας προμηθευτής Send καθορίζει την αποστολή και ακυρώνει τις λειτουργίες και μπορεί να επισημάνει το γεγονός sendDone.

Αντιθέτως, ένας χρήστης Send πρέπει να καθορίσει το γεγονός sendDone και μπορεί να καλέσει τις εντολές send και cancel. Όταν μια κλήση αποστολής Send επιστρέφει Success, η παράμετρος msg έχει περάσει στον προμηθευτή, ο οποίος θα προσπαθήσει να στείλει το πακέτο. Όταν η αποστολή ολοκληρωθεί, τα σήματα sendDone των προμηθευτών, περνούν το δείκτη πίσω στο χρήστη.

4.2.5 Tasks

Ένα task είναι μια αναβληθήσα κλήση διαδικασίας. Μια ενότητα μπορεί να ταχυδρομήσει ένα task στο TinyOS χρονοπρογραμματιστή. Σε κάποιο σημείο αργότερα, ο χρονοπρογραμματιστής θα εκτελέσει το task.Επειδή το task δεν καλείται αμέσως, δεν υπάρχει καμία επιστροφή τιμής . Επίσης, επειδή ένα task εκτελεί στο ονομαζόμενο πλαίσιο ενός συστατικού, δεν παίρνει οποιεσδήποτε παραμέτρους: οποιαδήποτε παράμετρο θέλετε να περάσετε μπορείτε να την αποθηκεύετε στο συστατικό. Tasks, όπως οι λειτουργίες, μπορούν να δηλωθούν εκ των προτέρων με μια δήλωση που μοιάζει με:

```
task void readDoneTask();
```

Εικόνα 4.7 Δήλωση Task

Ένας καθορισμός είναι όπως τη δήλωση, αλλά περιλαμβάνει και ένα σώμα λειτουργίας. Ένα συστατικό στέλνει έναν task στον TinyOS χρονοπρογραμματιστή με τη λέξη κλειδί post :

```
post readDoneTask();
```

Εικόνα 4.8 Αποστολή ενός Task στον χρονοπρογραμματιστή

Τα tasks είναι non-preemptive. Αυτό σημαίνει ότι μόνο ένα task τρέχει οποιαδήποτε στιγμή, και το TinyOS δεν διακόπτει ένα task για να τρέξει άλλο. Μόλις αρχίσει ένα task, κανένα άλλο task δεν τρέχει έως ότου ολοκληρωθεί. Αυτό σημαίνει ότι τα tasks εκτελούνται ατομικά όσον αφορά το ένα το άλλο. Εάν ένα συστατικό εκτελεί έναν υπολογισμό που διαρκεί πολύ, πρέπει να τον σπάσει σε πολλαπλά tasks. Ένα task μπορεί να ποστάρει τον εαυτό του. Παραδείγματος χάριν, στο βασικός βρόχος εκτέλεσης του Mate bytecode ο διερμηνέας είναι ένα task που εκτελεί μερικές οδηγίες ενός νήματος και ποστάρει τον εαυτό του.

4.2.6 Ταυτοχρονισμός στην NesC

Ο ταυτοχρονισμός (concurrency) παίζει βασικό ρόλο στα συστατικά της NesC. Τα συμβάντα (ή οι εντολές) μπορούν να σηματοδοτηθούν άμεσα ή έμμεσα από μια διακοπή, κάτι που τα κατατάσσει στον τομέα των ασύγχρονων κωδίκων. Για να χειριστεί αυτόν τον ταυτοχρονισμό, η γλώσσα NesC παρέχει δυο εργαλεία τα atomic sections και τις εργασίες (tasks).

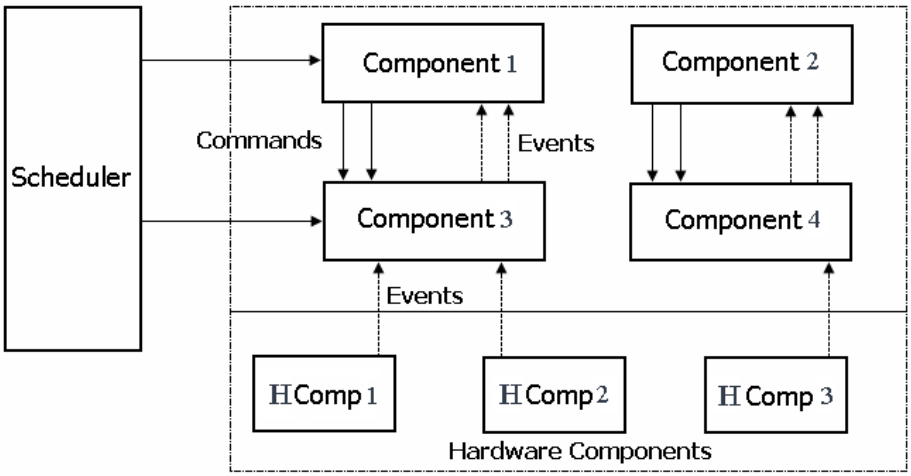
Στην περίπτωση που ένας ασύγχρονος κώδικας αποκτά πρόσβαση σε μια μεταβλητή x, τότε κάθε πρόσβαση στη μεταβλητή αυτή έξω από μια atomic δήλωση είναι λάθος που χτυπάει κατά τη μεταγλώττιση. Για αυτό, ο προγραμματιστής πρέπει να δηλώσει το τμήμα αυτό ως ‘atomic’ ή να προωθήσει το ανάλογο τμήμα του κώδικα σε μια εργασία.

Η χρήση ασύγχρονου (async) κώδικα για την απόκριση σε μια διακοπή υλικού πρέπει να γίνεται πολύ προσεκτικά.. Και αυτό γιατί, εκτελώντας ο ασύγχρονος κώδικας μια σχετικά χρονοβόρα διαδικασία επεξεργασίας, αναγκαστικά δεν αφήνει περιθώρια στον επεξεργαστή να χειριστεί αξιόπιστα άλλες διακοπές υλικού εκείνη τη στιγμή, με αποτέλεσμα το όλο σύστημα να χάνει σε ανταπόκριση και αξιοπιστία. Για αυτό το λόγο το μέγεθος της επεξεργασίας που εκτελεί ο ασύγχρονος κώδικας πρέπει να είναι μικρό και τα atomic sections μέσα στον κώδικα πρέπει να αποφεύγουν την απευθείας κλήση εντολών ή σηματοδοσία συμβάντων όσο το δυνατόν περισσότερο. Στην περίπτωση που ένας ασύγχρονος κώδικας έχει να εκτελέσει ένα μεγάλο επεξεργαστικό φόρτο μπορεί να θέσει μια εργασία η οποία θα αναλαμβάνει την επεξεργασία αυτή. Με αυτό τον τρόπο μεταφέρεται ο έλεγχος από ένα ασύγχρονο πλαίσιο εφαρμογής σε ένα σύγχρονο πλαίσιο. Η συγκεκριμένη εργασία θα εκτελεστεί σύγχρονα και, ενώ μπορεί να μην έχει ήδη ολοκληρωθεί, υπάρχει περίπτωση μια άλλη διακοπή υλικού να εμφανιστεί η οποία θα διαχειριστεί άμεσα (καθώς υπερτερεί της εργασίας) και με τον ίδιο τρόπο θα δημιουργηθεί μια νέα εργασία η οποία θα εκτελεστεί μόλις τελειώσει η προηγούμενη (καθώς η μια εργασία δεν μπορεί να προηγηθεί της άλλης).

4.3 Δομή του TinyOS

Προκειμένου να ικανοποιηθούν τα απαραίτητα επίπεδα παραλληλισμού (concurrency levels), το TinyOS χρησιμοποιεί ένα δομικό μοντέλο που βασίζεται σε καταστάσεις αντί σε νήματα. Μετατρέποντας τα επιμέρους τμήματα διεργασιών (components) σε «μηχανές καταστάσεων», δημιουργείται η δυνατότητα αποδοτικής χρήσης της CPU και της μνήμης. Έτσι, αντί να αφιερώνονται πολλαπλοί σωροί για κάθε τρέχουσα διεργασία, μπορεί να μοιρασθεί ένα ενιαίο πλαίσιο εκτέλεσης μεταξύ των διαφόρων μηχανών καταστάσεων.

Το κάθε συστατικό, τώρα, χρησιμοποιεί γεγονότα και εντολές για να μεταβαίνει γρήγορα από κατάσταση σε κατάσταση. Συνήθως, αυτές οι μεταβάσεις θεωρούνται στιγμιαίες, απαιτώντας πολύ μικρή υπολογιστική ισχύ. Έτσι, κάθε συστατικό προσωρινά δεσμεύει τον επεξεργαστή για όσο χρόνο διαρκούν αυτές οι αλλαγές καταστάσεων. Σε περίπτωση, βέβαια, που απαιτείται περισσότερη υπολογιστική ισχύ χρησιμοποιούνται οι διεργασίες, οι οποίες χρονοπρογραμματίζονται στον scheduler και εκτελούνται χωρίς να διακόπτονται από άλλες διεργασίες (μόνο από τα μεγαλύτερης προτεραιότητας events) μέχρι την ολοκλήρωσή τους. Η διαδικασία δε, που ακολουθείται κατά τον χρονοπρογραμματισμό των διεργασιών είναι FIFO (First In First Out). Η δομή του TinyOS φαίνεται στη Εικόνα 4.9.



Εικόνα 4.9 Δομή του TinyOS

Τα συστατικά ικανοποιούν την απαίτηση για μια αρκετά δομημένη αρχιτεκτονική λογισμικού. Κάθε συστατικό, όπως φαίνεται και στην Εικόνα 4.10, αποτελείται από τέσσερα τμήματα:

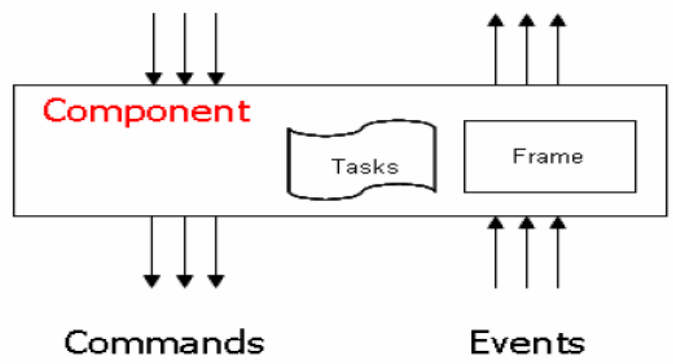
— ένα χειριστή εντολών (command handler)

— ένα χειριστή γεγονότων (event handler)

— μια ποσότητα μνήμης (frame)

— ένα σύνολο από δυνατές διεργασίες (tasks)

Η μνήμη χρησιμοποιείται για αποθήκευση της εσωτερικής κατάστασης του συστατικού, ενώ οι διεργασίες και οι χειριστές εντολών και γεγονότων εκτελούνται με βάση το περιεχόμενο αυτού του πλαισίου μνήμης. Επιπλέον δε, το κάθε συστατικό δηλώνει τις εντολές που χρησιμοποιεί και τα γεγονότα που δημιουργεί. Με αυτόν τον τρόπο, δημιουργούνται επίπεδα-στρώματα από συστατικά, με αυτά των υψηλότερων επιπέδων κάθε φορά να χρησιμοποιούν εντολές για κλήση λειτουργιών που παρέχονται από κατώτερων επιπέδων συστατικά και τα τελευταία να δημιουργούν γεγονότα προς τα πρώτα. Έτσι, εισάγεται η έννοια της αμφίδρομης διεπαφής (bi-directional interface) η οποία ορίζει τις εντολές και τα γεγονότα που χρησιμοποιούνται κατά την αλληλεπίδραση των συστατικά μεταξύ τους.



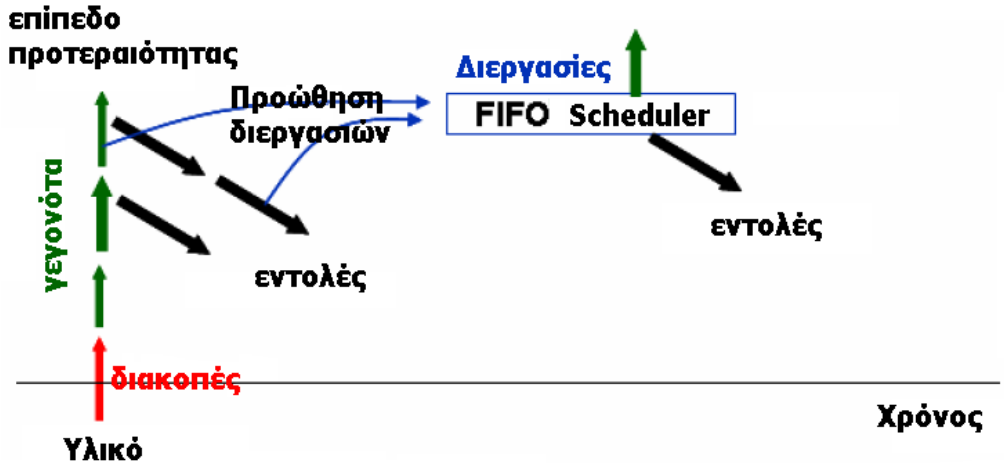
Εικόνα 4.10 Δομή ενός component

Οι εντολές είναι αιτήματα που υποβάλλονται στα χαμηλότερου επιπέδου συστατικά. Κάθε μια από αυτές επιστρέφει στο συστατικό που την κάλεσε μια πληροφορία γύρω από την κατάστασή της. Την πληροφορία αυτή λαμβάνει ο χειριστής εντολών και την τοποθετεί στο πλαίσιο μνήμης του συστατικού και έπειτα προωθεί μια διεργασία στο σωρό των διεργασιών για εκτέλεση. Η επιβεβαίωση ότι η λειτουργία που συνεπάγεται η συγκεκριμένη κάθε φορά εντολή ολοκληρώθηκε, γίνεται με τη δημιουργία ενός γεγονότος από το χαμηλότερου επιπέδου συστατικού.

Οι χειριστές γεγονότων αντιδρούν σε γεγονότα που παράγονται από κατώτερων επιπέδων συστατικά ή όταν είναι άμεσα συνδεδεμένοι με το υλικό, σε διακοπές (interrupts). Επίσης, όπως και στις εντολές, το περιεχόμενο του πλαισίου μνήμης τροποποιείται και δημιουργούνται διεργασίες. Έτσι, τα γεγονότα μπορούν να καλέσουν εντολές, να δημιουργήσουν νέα γεγονότα και διεργασίες και να διακόψουν διεργασίες, ενώ δεν μπορούν να δημιουργηθούν από εντολές και να διακοπούν από διεργασίες. Τα κατώτερα επιπέδου γεγονότα, βέβαια, δημιουργούνται από διακοπές του υλικού.

Οι διεργασίες επιτελούν τις βασικές λειτουργίες και τους εντατικούς υπολογισμούς. Εκτελούνται μέχρι την ολοκλήρωσή τους και μπορούν να διακοπούν μόνο από γεγονότα. Επίσης, χρονοπρογραμματίζονται στον αντίστοιχο scheduler με FIFO τρόπο, δηλαδή σειριακά, γι' αυτό και πρέπει να είναι σύντομες. Ωστόσο, μπορεί να χρησιμοποιηθεί και βασισμένος σε προτεραιότητα scheduler αντί του FIFO, ώστε να ελαττωθεί η συνολική καθυστέρηση εκτέλεσης των διεργασιών.

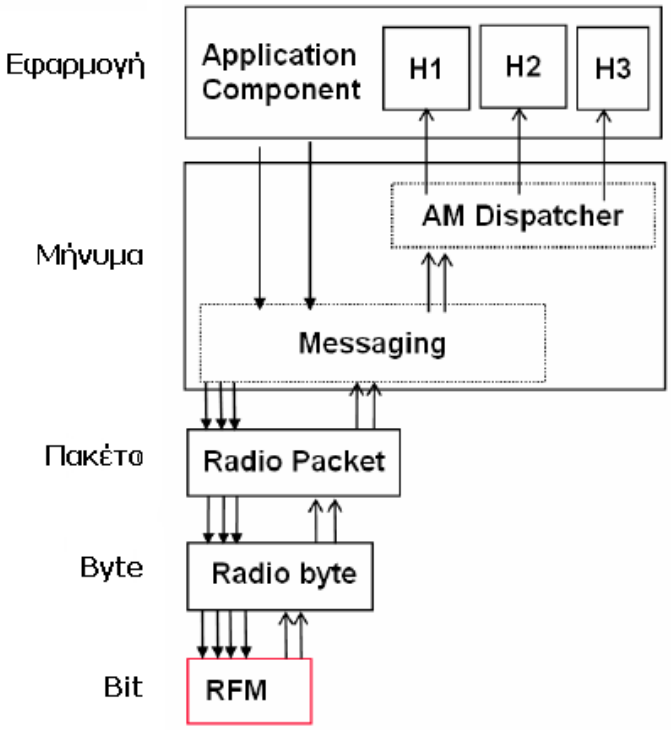
Παρακάτω, στην Εικόνα 4.11 φαίνεται η όλη δραστηριότητα μεταξύ των τεσσάρων στοιχείων από τα οποία αποτελείται ένα συστατικό.



Εικόνα 4.11 Δραστηριότητα των στοιχείων ενός component

4.4 Επικοινωνία στο TinyOS

Η στοίβα επικοινωνίας στο TinyOS φαίνεται στην Εικόνα 4.12.



Εικόνα 4.12 Στοίβα επικοινωνίας του TinyOS

Κάθε ένα από τα επίπεδα επικοινωνίας περιγράφεται παρακάτω:

- Επίπεδο bit (RFM - RF module – μονάδα επικοινωνίας): Σε αυτό το επίπεδο μπορεί να καθορισθεί ο τρόπος λειτουργίας (εκπομπή ή λήψη) και ο ρυθμός δειγματοληψίας, να ληφθεί ή να αποσταλεί ένα bit και να υπάρξει κατάλληλη ειδοποίηση για το αν η εκπομπή ή λήψη ολοκληρώθηκε, έτσι ώστε να μεταβεί το RFM σε ανενεργή κατάσταση ή γενικά σε μικρή κατανάλωση ενέργειας κατάσταση.

- Επίπεδο byte (Radio byte) : Αυτό το επίπεδο είναι υπεύθυνο για την κωδικοποίηση των bit μετάδοσης (κωδικοποίηση Manchester) σε επίπεδο byte, την ανίχνευση και διόρθωση λαθών, και την ανίχνευση του επιπέδου ισχύος του σήματος έτσι ώστε να μπορεί να εντοπιστεί αν το κανάλι είναι ελεύθερο για μετάδοση δεδομένων, διαφορετικά εισέρχεται η μονάδα επικοινωνίας σε αναμονή για ένα τυχαίο χρονικό διάστημα.

- Επίπεδο πακέτου (Radio packet) : Σε αυτό το επίπεδο γίνεται 16-bit crc έλεγχος και δημιουργία πλεονάζουσας εκπομπής δεδομένων.

- Επίπεδο μηνύματος (Messaging) : Εδώ, οι πληροφορίες ενσωματώνονται σε πακέτα (διαίρεση, συνδυασμός) και γίνεται δρομολόγησή τους σύμφωνα με κατάλληλες διευθύνσεις προς την μονάδα επικοινωνίας (broadcast) ή την σειριακή θύρα (UART). Μέρος δε αυτού, αποτελεί και ο AM dispatcher ο οποίος κατευθύνει τα πακέτα-ενεργά μηνύματα προς τους αντίστοιχους χειριστές πακέτων (packet handlers)

- Επίπεδο εφαρμογής (Application) : Σε αυτό το επίπεδο αναπτύσσονται τα πρωτόκολλα δρομολόγησης, οι εφαρμογές και γενικά η επεξεργασία δεδομένων που παρέχονται από τα πακέτα μετάδοσης.

4.4.1 Η δομή error_t

Το TinyOS ορίζει ένα ειδικό τύπο δεδομένων ο οποίος καλύπτει πολλούς διαφορετικούς κωδικούς σφάλματος, το error_t. Ο συγκεκριμένος τύπος είναι η τιμή που επιστρέφουν πολλές εντολές και χειριστές συμβάντων. Οι πιο συνηθισμένες τιμές που μπορεί να πάρει φαίνονται παρακάτω [43]:

- SUCCES : Μήνυμα επιτυχίας
- FAIL : Γενικό μήνυμα λάθους (παρωχημένο)
- EBUSY: Το υποσύστημα είναι απασχολημένο με ένα άλλο αίτημα, δοκιμάστε ξανά αργότερα
- ERETRY: Υπήρξε κάποιο πρόβλημα, δοκιμάστε ξανά αργότερα.

Άλλες τιμές ορίζονται στη διεύθυνση $TOSROOT/types/TinyError.h

Παραδείγματος χάριν, στην περίπτωση που μια εφαρμογή καλεί τη Send, ακολουθούν τα εξής: Αφού ολοκληρωθεί η εκτέλεση της Send, γίνεται η σηματοδότηση του γεγονότος sendDone(message_t* msg, error_t err), το οποίο με τη σειρά του αποθηκεύει στη μεταβλητή error την κατάλληλη τιμή σφάλματος, ανάλογα με την έκβαση που είχε η εκτέλεση της Send. Όπως είναι λογικό ότι η σημασία του τύπου δεδομένων error_t είναι μεγάλη, αφού δίνει τη δυνατότητα στο σύστημα να γνωρίζει αν κάποια αναμενόμενη ενέργεια έχει ολοκληρωθεί σωστά ή όχι και να προχωρήσει στην ανάλογη ενέργεια.

4.4.2 Χρήση της δομής του Ενεργού Μηνύματος

Στο λειτουργικό σύστημα TinyOS, πρωτόκολλα, όπως το TCP/IP, δε μπορούν να χρησιμοποιηθούν για την υλοποίηση της επικοινωνίας. Οι λόγοι είναι ότι τα πρωτόκολλα αυτά βασίζονται στο μεγάλο εύρος ζώνης των αντίστοιχων επικοινωνιών, χρησιμοποιούν σήματα έναρξης, διακοπής και αναμονής και τέλος απαιτούν πολλή μνήμη για αποθήκευση δεδομένων στο σωρό του δικτύου (network stack) όπου και παραμένουν μέχρι τα διάφορα νήματα (threads), που αποτελούν τις εφαρμογές, τα «διαβάσουν». Το τελευταίο δε, σημαίνει ότι τα νήματα μπλοκάρονται μέχρι τα δεδομένα να είναι διαθέσιμα στο σωρό. Στα παραπάνω έρχεται να προστεθεί και το γεγονός ότι στα WSN υπάρχουν οι περιορισμοί πραγματικού χρόνου και χαμηλής επεξεργαστικής ισχύος.

Το ενεργό μήνυμα (Active Message – AM) είναι μια δομή η οποία χρησιμοποιείται για τα μεταδιδόμενα πακέτα δεδομένων σε ένα WSN και έχει τα εξής χαρακτηριστικά:

- Ενσωμάτωση επικοινωνίας και υπολογισμού
- Ταύτιση στοιχείων της επικοινωνίας με δυνατότητες του υλικού
- Παροχή ενός μοντέλου οδηγούμενων συμβάντων όσον αφορά την επικοινωνία μεταξύ των κόμβων, εννοώντας ότι και στο επίπεδο επικοινωνίας, όπως και τα συστατικά (αναφέρθηκε προηγουμένως), οι κόμβοι στέλνουν ο ένας γεγονότα-events στον άλλο.
- Ταίριασμα με το αντίστοιχο οδηγούμενων συμβάντων μοντέλο του TinyOS.

Η δομή αυτή ονομάζεται message_t και ορίζεται στη διεύθυνση $TORROOT/tos/types/message.h. Πιο συγκεκριμένα η δομή φαίνεται στην Εικόνα 4.13. Κάθε πλατφόρμα έχει την ελευθερία να ορίσει το περιεχόμενο των πεδίων κεφαλίδα (header), υποσέλιδο (footer), και μεταδεδομένα (metadata). Το πεδίο φορτίο (payload) χρησιμοποιείται από την εκάστοτε εφαρμογή για την αποθήκευση δεδομένων. Το μέγιστο μέγεθος αποθηκευμένων δεδομένων είναι ίσο με TOSH_DATA_LENGTH και μπορεί να οριστεί από το χρήστη, ενώ η προεπιλογή είναι 28[43].

Κάθε μήνυμα – πακέτο AM – περιέχει στο πεδίο του φορτίου (payload) ένα πεδίο που ορίζει το χειριστή γεγονότων (event-handler) που θα ενεργοποιηθεί και θα λάβει τα δεδομένα που περιέχονται σε αυτό (στο μήνυμα). Για το λόγο αυτό, οι χειριστές των μηνυμάτων πρέπει να εκτελούνται γρήγορα ώστε να μη δημιουργείται συμφόρηση στο δίκτυο και να επιτυγχάνεται ικανοποιητική λειτουργία του όλου συστήματος. Στο τελευταίο συμβάλλει και η χρήση των αισθητήρων, οι οποίοι δημιουργώντας αντίστοιχα γεγονότα μπορούν να συντονίσουν την όλη

επικοινωνία σε ένα δίκτυο. Η διαδικασία της επικοινωνίας που εκτελείται είναι η εξής: Κάθε ενεργό μήνυμα περιέχει το όνομα-αναγνωριστικό ενός χειριστή γεγονότων. Ένας κόμβος, λοιπόν, που στέλνει ένα τέτοιο μήνυμα, ενσωματώνει τα δεδομένα, ονομάζει έναν χειριστή γεγονότων και ζητά αποστολή του μηνύματος η οποία επιβεβαιώνεται με ένα αντίστοιχο σήμα όταν ολοκληρωθεί η διαδικασία αποστολής. Από την άλλη μεριά, στον κόμβο που λαμβάνει το μήνυμα ενεργοποιείται αυτόματα ο χειριστής γεγονότων που καθορίζεται από το μήνυμα αυτό. Έτσι, δεν υπάρχουν εμποδιζόμενες ή καθυστερημένες διεργασίες, ενώ επικρατεί απλή διαδικασία αποθήκευσης στο σωρό του δικτύου (network stack).

```
typedef nx_struct message_t {
  nx_uint8_t header[sizeof(message_header_t)];
  nx_uint8_t data[TOSH_DATA_LENGTH];
  nx_uint8_t footer[sizeof(message_footer_t)];
  nx_uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

Εικόνα 4.13 Δομή ενεργού μηνύματος

Η παραπάνω δομή χρησιμοποιείται για την ανταλλαγή δεδομένων από τις διεπαφές του TinyOS που σχετίζονται με επικοινωνία και αυτό την καθιστά μία εξαιρετικά σημαντική, αλλά και εξαιρετικά χρήσιμη δομή. Τέτοιες διεπαφές είναι οι AMSend, Receive, Packet, AMPacket κ.α.

4.5 Το πρωτόκολλο ΤΥΜΟ

4.5.1 Επισκόπηση πρωτοκόλλων δρομολόγησης Multi-Hop [52]

- Flooding

Πρόκειται για την πιο απλή μέθοδο για την αποστολή πακέτων χωρίς τη χρήση πίνακα δρομολόγησης διπλής κατεύθυνσης. Το πακέτο στέλνεται από τον κόμβο σε όλους τους γείτονές του, κίνηση που ονομάζεται εκπομπή. Όλοι οι ενεργοί κόμβοι προωθούν τα πακέτα στους γείτονές τους. Το πακέτο διαδίδεται κατά συνέπεια ως μια πλημμύρα (Flood, από όπου προκύπτει και ο όρος Flooding). Το μεγάλο μειονέκτημα του συγκεκριμένου πρωτοκόλλου είναι η υψηλή κατανάλωση ενέργειας.

- DD

Η δρομολόγηση Direct Diffusion είναι ένα αντιδραστικό πρωτόκολλο που βασίζεται σε πίνακα. Η λειτουργία αυτού του πρωτοκόλλου δρομολόγησης είναι περιγράφεται στο [45].

- MCFA

Το πρωτόκολλο Minimum Cost Forwarding Algorithm είναι μια αντιδραστική μέθοδος για την αποστολή δεδομένων κατά μήκος ενός προ-αρχικοποιημένου μονοπατιού. Περισσότερες πληροφορίες δίνονται στο [46].

- AODV

Το Ad-hoc On-demand Distance Vector είναι ένα αντιδραστικό πρωτόκολλο δρομολόγησης. Μια εισαγωγή για να AODV δίνεται στο [47] και υλοποίηση δίνεται στο [48]. Σε ένα τέτοιο πρωτόκολλο το μονοπάτι από την πηγή στον προορισμό επεκτείνεται on-demand.

- DYMO / TYMO

Ο στόχος της εφαρμογής Tymo είναι να παρέχει ένα συστατικό σε μια εφαρμογή, προκειμένου να γίνεται διαφανής αποστολή και λήψη δεδομένων σε ένα δίκτυο multi-hop. Είναι η υλοποίηση στο TinyOS του DYMO πρωτοκόλλου, ενός point-to-point πρωτοκόλλου δρομολόγησης για Mobile Ad-hoc δίκτυα (MANETs). Αρχικά είχε σχεδιαστεί από την IETF για να βρίσκει διαδρομές δυναμικά στην κορυφή της IP stack. Αναλυτικές πληροφορίες δίνονται στο [49], [50] και [51]. Στο TYMO, προσαρμόστηκε η μορφή πακέτου του και υλοποιήθηκε πάνω στη στοίβα του Ενεργού Μηνύματος του TinyOS.

Μία σύντομη αξιολόγηση των προαναφερθέντων πρωτοκόλλων φαίνεται στον πίνακα 4-1[52].

Πίνακας 4-1 Αξιολόγηση πρωτοκόλλων δρομολόγησης

	Flooding	MCFA	DD	AODV	DYMO
energy efficiency	--	0	-	+	+
scalability	++	+	+	++	++
application area	++	0	+	0	0
robustness	++	0	+	+	+

++ very good; + good; o neutral; - bad; -- poor

4.5.2 DYMO: Dynamic MANET On-demand

Ως reactive πρωτόκολλο, το DYMO δεν αποθηκεύει ρητά την τοπολογία του δικτύου. Αντ'αυτού, οι κόμβοι υπολογίζουν μια διαδρομή προς τον επιθυμητό προορισμό μόνο όταν χρειάζεται. Ως αποτέλεσμα, λίγες πληροφορίες δρομολόγησης ανταλλάσσονται, το οποίο μειώνει την επιβάρυνση του δικτύου και κατά συνέπεια εξοικονομεί εύρος ζώνης και ενέργεια. Επίσης, δεδομένου ότι λίγες πληροφορίες δρομολόγησης αποθηκεύονται, το DYMO

μπορεί να χρησιμοποιηθεί από συσκευές με περιορισμένους πόρους μνήμης, όπως οι πλατφόρμες αισθητήρων.

Όταν ένας κόμβος χρειάζεται μια διαδρομή, διαδίδει μια Αίτηση Διαδρομής (RREQ), η οποία είναι ένα πακέτο, ζητώντας για μια διαδρομή μεταξύ του εντολέα και ενός κόμβου-στόχου. Το πακέτο στέλνεται στο σύνολο του δικτύου ή σε έναν αριθμό hops από τον εντολέα. Όταν το πακέτο φθάσει στο στόχο του (ή σε ένα κόμβο που έχει μια νέα διαδρομή προς τον στόχο), ο κόμβος απαντά με μια Απάντηση Διαδρομής (RREP). Ένα τέτοιο πακέτο είναι παρόμοιο με ένα Αίτημα Διαδρομής, αλλά ακολουθεί μια διαδρομή μοναδικής διανομής και δεν ενεργοποιείται καμία απόκριση όταν ο στόχος έχει επιτευχθεί.

Όταν οι κόμβοι λάβουν ένα RREQ ή RREP, αποθηκεύουν προσωρινά πληροφορίες σχετικά με τον αποστολέα σε επίπεδο συνδέσμου (τον τελευταίο αποστολέα του RREQ ή RREP) και την πηγή προέλευσης, ώστε να γνωρίζουν μια διαδρομή προς την πηγή προέλευσης που μπορούν να χρησιμοποιησούν αργότερα (αν είναι αρκετά πρόσφατη) χωρίς την αποστολή ενός RREQ. Οι κόμβοι έχουν τη δυνατότητα να συγκεντρώνουν την διαδρομή που ακολούθησε το πακέτο στο ίδιο το πακέτο. Έτσι, όταν οι κόμβοι διαδίδουν μια RREQ ή RREP, πολλές πληροφορίες μπορούν να ληφθούν από το πακέτο, πολύ περισσότερο από μια διαδρομή μεταξύ δύο κόμβων.

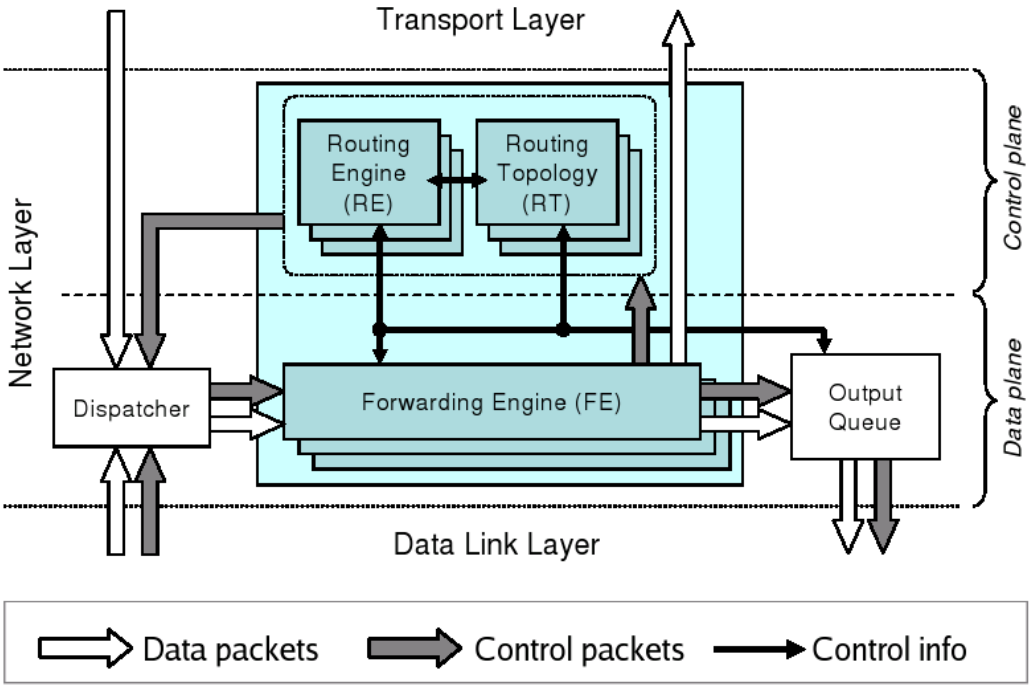
Όταν διαδρομές δεν έχουν χρησιμοποιηθεί για μεγάλο χρονικό διάστημα, διαγράφονται. Αν ένας κόμβος καλείται να διαβιβάσει ένα πακέτο μέσα από μια διαδρομή που έχει διαγραφεί, παράγει ένα μήνυμα σφάλματος διαδρομής (RERR) για να προειδοποιήσει το αρχικό κόμβο (και άλλους κόμβους) ότι αυτή η διαδρομή δεν είναι πλέον διαθέσιμη. Ως ένα άλλο μηχανισμό συντήρησης διαδρομής, το DYMO χρησιμοποιεί αριθμούς ακολουθίας και μετράει hop για τον προσδιορισμό της χρησιμότητας και της ποιότητας της διαδρομής.

4.5.3 Υλοποίηση Tymo

Ο σκοπός της DYMO είναι να βρεθούν διαδρομές on demand. Για την μεταφορά των δεδομένων κατά μήκος αυτών των διαδρομών, σχεδιάστηκε και εφαρμόστηκε ένα πολύ απλό πρωτόκολλο μεταφοράς που απλά ονομάζεται MH για multi-hop και χρησιμοποιεί διεύθυνση 16-bit, δηλαδή της ίδιας μορφής με το DYMO. Δεν κάνει τίποτα παραπάνω από την προώθηση πακέτων, εκτός εάν ο κόμβος-παραλήπτης είναι ο στόχος του πακέτου. Η αποστολή και λήψη μηνυμάτων MH γίνεται μέσα από τις ίδιες τεχνικές διεπαφές με τη στοίβα Ενεργού Μηνύματος του TinyOS.

4.5.3.1 Αρχική διάταξη

Η διάταξη του Tymo διαιρείται σε δύο μέρη: το επίπεδο των δεδομένων και το επίπεδο ελέγχου. Η λειτουργία αυτής της διάταξης φαίνεται στο την Εικόνα 4.14.



Εικόνα 4.14 Η αποσύνθεση σε επίπεδο δικτύου

Ο Δρομολογητής (Dispatcher) εξετάζει την επικεφαλίδα των πακέτων που προέρχονται από το κατώτερο ή ανώτερο στρώμα προκειμένου να καθοριστεί το πρωτόκολλο, στο οποίο ανήκει το πακέτο, και περνάει το πακέτο στην αρμόδια υπηρεσία πρωτοκόλλου, το οποίο είναι ένα σύνολο που αποτελείται από τη Μηχανή Προώθησης (Forwarding Engine), τη Μηχανή Δρομολόγησης (Routing Engine) και την Τοπολογία Δρομολόγησης (Routing Topology).

Αν και η Μηχανή Προώθησης είναι μέρος μιας υπηρεσίας πρωτοκόλλου, δεν γνωρίζει τη μορφή του πρωτοκόλλου και των αλγορίθμων. Ζητά απλώς από τη Μηχανή Δρομολόγησης την πλήρωση της επικεφαλίδας δρομολόγησης του πακέτου πριν την προώθησή του, ή να παραδώσει το πακέτο στο ανώτερο στρώμα όταν το πακέτο έχει φτάσει στον προορισμό της. Ο λόγος για τον οποίο η Μηχανή Προώθησης ανήκει στην υπηρεσία πρωτοκόλλου είναι ότι μπορεί να εκτελέσει συνάθροιση πακέτων ή προγραμματισμό και αυτά τα καθήκοντα εξαρτώνται από το πρωτόκολλο.

Η Μηχανή Δρομολόγησης και η Τοπολογία Δρομολόγησης είναι οι βασικές συνιστώσες ενός πρωτοκόλλου: ενώ η Μηχανή Δρομολόγησης παράγει και επεξεργάζεται πακέτα ελέγχου, η Τοπολογία Δρομολόγησης υπολογίζει και αποθηκεύει τις απαραίτητες πληροφορίες σχετικά με την τοπολογία του δικτύου, σύμφωνα με τα στοιχεία που αναφέρθηκαν από τη Μηχανή Δρομολόγησης.

Τέλος, η ουρά εξόδου χειρίζεται τα πακέτα που αποστέλλονται από όλα τα πρωτόκολλα που εκτελούνται στον κόμβο. Δεδομένου ότι όλα τα πακέτα πρέπει να περάσουν από αυτό το συστατικό για να σταλούν, η ουρά εξόδου μπορεί να τα προγραμματίζει σύμφωνα με την πολιτική του κόμβου.

4.5.3.2 Διασυνδέσεις, συστατικά και διασύνδεση

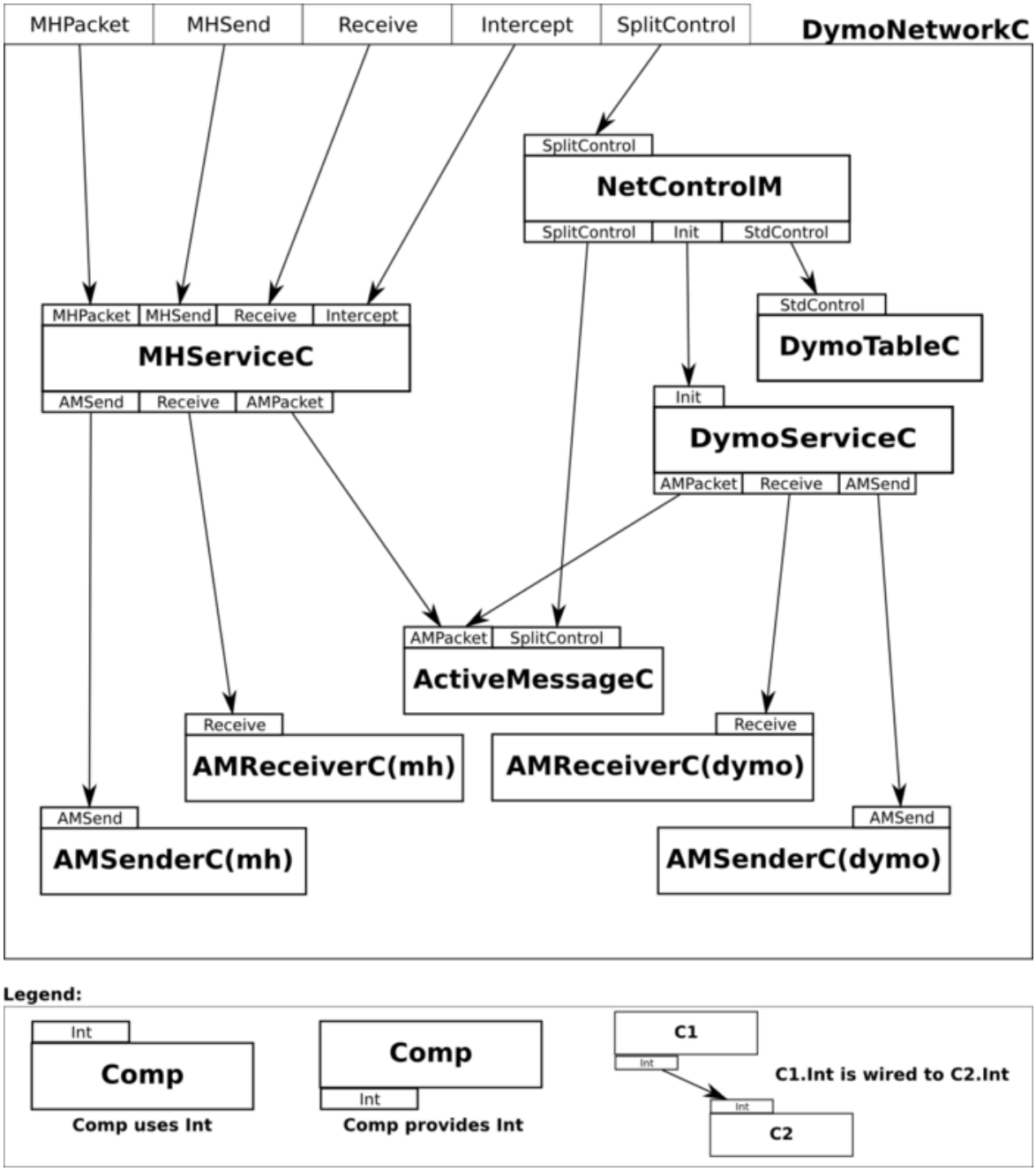
Για τη αποστολή και λήψη μηνυμάτων MH, θα πρέπει να δηλωθεί η χρήση των διεπαφών:

```
module TestM {
       uses {
         interface SplitControl;
         interface AMPacket as MHPacket;
         interface Packet;
         interface Receive;
         interface AMSend as MHSend;
       }
     }
```

Για διευκόλυνση του χρήστη, το συστατικό που ονομάζεται DymoNetworkC παρέχει μια σειρά διεπαφών Receive και Send, έτσι ώστε διάφορες εφαρμογές να μπορούν να χρησιμοποιήσουν το MH ανεξάρτητα. Ως εκ τούτου, αρκεί μια δήλωση παρόμοια με:

```
implementation {
       components TestM, DymoNetworkC;

       TestM.SplitControl -> DymoNetworkC;
       TestM.Packet       -> DymoNetworkC;
       TestM.MHPacket     -> DymoNetworkC;
       TestM.Receive      -> DymoNetworkC.Receive[1];
       TestM.MHSend       -> DymoNetworkC.MHSend[1];
     }
```

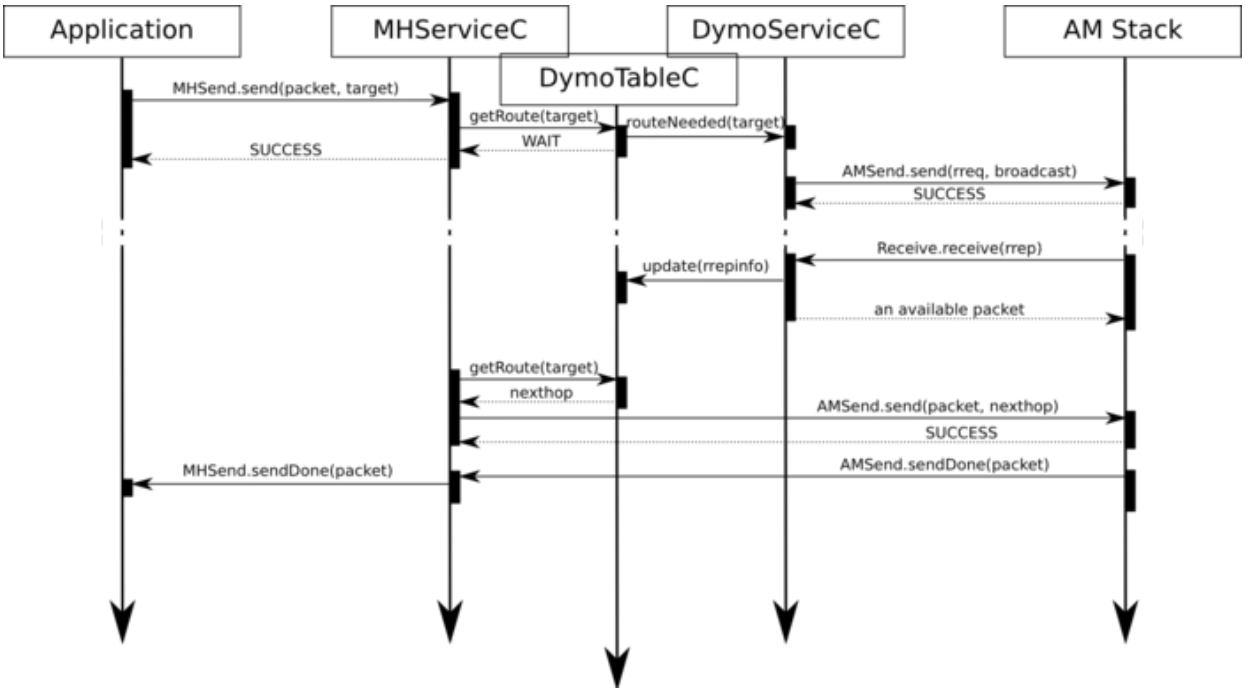
Οι διασυνδέσεις που χρησιμοποιεί το DymoNetworkC φαίνονται στη Εικόνα 4.14. Για να μην περιπλακεί περαιτέρω το διάγραμμα, δεν αναπαριστούνται οι διεπαφές Packet.



Εικόνα 4.15 Γενικό σχεδιάγραμμα των συστατικών που χρησιμοποιεί το DymoNetworkC

Για καλύτερη κατανόηση, το διάγραμμα ακολουθίας στην Εικόνα 4.16 δείχνει τις αλληλεπιδράσεις μεταξύ των συστατικών όταν η εφαρμογή θέλει να στείλει ένα πακέτο σε ένα "άγνωστο" κόμβο. Η εφαρμογή αγνοεί τις διαδικασίες δρομολόγησης, έτσι στέλνει το πακέτο ως εάν ήταν ένα single-hop πακέτο. Το πακέτο δίνεται στο MHServiceC το οποίο δεν ξέρει πώς το πρωτόκολλο δρομολόγησης λειτουργεί, αλλά έχει επίγνωση του γεγονότος ότι η λήψη μιας διαδρομής μπορεί να μην είναι άμεση. Ως εκ τούτου, όταν ο πίνακας δρομολόγησης (ο οποίος μοιράζεται μεταξύ των δύο υπηρεσιών πρωτοκόλλου) σηματοδοτεί ότι η διαδρομή δεν είναι ακόμη διαθέσιμη, η εντολή send επιστρέφει και η υπηρεσία MH θα επαναλαμβάνει τακτικά την

αποστολή του πακέτου. Εν τω μεταξύ, ο πίνακας δρομολόγησης σηματοδοτεί την υπηρεσία DYMO ότι μια διαδρομή είναι αναγκαία, και ένα Αίτημα Διαδρομής εκδίδεται. Όταν η Απάντηση Διαδρομής φτάσει, ο πίνακας δρομολόγησης ενημερώνεται, έτσι ώστε η επόμενη προσπάθεια από την υπηρεσία MH να είναι επιτυχής. Το πακέτο δεδομένων τελικά αποστέλλεται στο επόμενο hop στη διαδρομή, και το sendDone γεγονός σηματοδοτείται στην εφαρμογή, ώστε να μπορεί να χρησιμοποιηθεί εκ νέου ο buffer πακέτων.



Εικόνα 4.16 διάγραμμα ακολουθίας για την αποστολή ενός πακέτου προκαλώντας ένα RREQ

4.5.3.3 Περιορισμοί

Είναι απαραίτητο να σημειωθεί ότι το έργο αυτό είναι υπο ανάπτυξη και έτσι δεν μπορεί να θεωρηθεί ως σταθερό ακόμη γιατί υπάρχουν κάποια σφάλματα κρυμμένα στον κώδικα και υπάρχουν στόχοι που πρέπει να επιτευχθούν για τη βελτίωση του.

4.6 Λήψη μετρήσεων από τους αισθητήρες

Για τη λήψη μετρήσεων από τους αισθητήρες που διαθέτει κάθε πλατφόρμα, το TinyOS διαθέτει τα συστατικά που παρέχουν μία ή περισσότερες Split-phase διεπαφές Read (όπως έχει αναφερθεί στο 4.2.4) της μορφής που φαίνεται στην Εικόνα 4.14.

```
interface Read<val_t> {
  command error_t read();
  event void readDone(error_t result, val_t val);
}
```

Εικόνα 4.17 Η διεπαφή Read

Για να ληφθεί μία μέτρηση, πρέπει να κληθεί η εντολή read. Με την ολοκλήρωσή της, σηματοδοτείται το γεγονός readDone. Δεδομένου ότι η επιστρεφόμενη τιμή της μεταβλητής result είναι SUCCESS, η μεταβλητή val θα περιέχει την ζητούμενη μέτρηση[43].

Κάθε εφαρμογή που έχει σκοπό τη λήψη μετρήσεων, πρέπει να χρησιμοποιήσει τις κατάλληλες διεπαφές για το συγκεκριμένο σκοπό. Ποιες θα είναι αυτές εξαρτάται τόσο από τον τύπο της πλατφόρμας και των τύπων αισθητήρων που αυτή διαθέτει, όσο και από τον τύπο των μετρήσεων, καθώς κάθε πλατφόρμα υποστηρίζει παραπάνω από ένα τύπο μετρήσεων και βρίσκονται διαθέσιμες στη διεύθυνση $TOSROOT/tos/sensorboard/[sensorboard].

Συγκεκριμένα, για την πλατφόρμα Tmote Sky οι κατάλληλες διεπαφές είναι οι εξής:

- SensirionSht11C() για μέτρηση θερμοκρασίας και υγρασίας
- HamamatsuS1087ParC για μέτρηση Ενεργής Φωτοσυνθετικά Ακτινοβολίας (PAR)
- HamamatsuS10871TsrC() για μέτρηση Ολικής Ηλιακής Ακτινοβολίας (TSR)
- Msp430InternalVoltageC() για μέτρηση εσωτερικής τάσης
- Msp430InternalTemperatureC() για μέτρηση εσωτερικής θερμοκρασίας

5. ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ

5.1 Επίπεδο σχεδιασμού: Μονάδες Tmote Sky

Το σημαντικότερο μέρος της παρούσας εργασίας στηρίζεται πάνω στην σωστή λειτουργία της multi-hop επικοινωνίας των μονάδων tmote sky μεταξύ τους. Για το σκοπό αυτό, είναι απαραίτητο η ανάπτυξη της εν λόγω εφαρμογής να γίνει με προσοχή και λεπτομέρεια. Για την αποφυγή λαθών, πάρθηκε η απόφαση της ανάπτυξη της σταδιακά. Με αυτό τον τρόπο αυξάνεται σταδιακά η πολυπλοκότητα της εφαρμογής και κάθε φορά που επιχειρείται να προστεθεί μια επιπλέον λειτουργία, έχει ήδη διαπιστωθεί ότι η προηγούμενη εκδοχή τρέχει σωστά και χωρίς προβλήματα. Έτσι διασφαλίζεται ότι στην περίπτωση που προκύψει κάποιο πρόβλημα, η αιτία βρίσκεται στις τροποποιήσεις που έχουν γίνει πρόσφατα και όχι στα αρχικά στάδια ανάπτυξη, δηλαδή γίνεται πιο εύκολη η αποσφαλμάτωση του κώδικα της εφαρμογής. Τα βήματα που ακολουθήθηκαν ήταν τα παρακάτω:

- Λήψη μετρήσεων από τον αισθητήρα της μονάδας
- Αποστολή μετρήσεων σειριακά στο PC μέσω USB
- Αποστολή μετρήσεων σε άλλο αισθητήρα (single-hop)
- Αποστολή μετρήσεων με προορισμό τον κόμβο-βάση (multi-hop)
- Τροποποίηση εφαρμογής κόμβου βάσης για να επεξεργάζεται τον τύπο μηνύματος multi-hop

Η εφαρμογή ονομάζεται SampleToRadio και βρίσκεται. Κάποια ενδιάμεσα στάδια του κώδικα είναι περιττό να παρουσιαστούν, εφόσον έχουν εξαλειφθεί ή έχουν τροποποιηθεί στην τελική εφαρμογή. Συνεπώς αναλυτικά παρακάτω θα αναφερθούν μόνο τα απαραίτητα στοιχεία του κώδικα.

5.1.1 Εγκατάσταση λογισμικού για τη χρήση μονάδων Tmote Sky

5.1.1.1 Μονάδες που απαρτίζουν το ασύρματο δίκτυο

Το πρώτο βήμα για την διαχείριση των αισθητήρων είναι η εγκατάσταση του λειτουργικού TinyOS. Η ανάπτυξη της εφαρμογής έγινε σε υπολογιστή με εγκατεστημένο το περιβάλλον Windows XP.

Με την εφαρμογή VMware player 7 [53] έγινε δυνατή η χρήση Linux λειτουργικού μέσω Windows (χωρίς την ανάγκη αλλαγής λειτουργικού) και συγκεκριμένα του Ubuntu Hardy Heron 8.04[54].

Αρχικά είναι απαραίτητη η εγκατάσταση του subversion και του Gnu C Compiler εκτελώντας τις παρακάτω εντολές σε ένα τερματικό:

```
sudo apt-get install subversion
```

```
sudo apt-get install build-essential
```

Έπειτα γίνεται ενεργοποίηση του TinyOS repository με τις εντολές:

```
sudo echo "deb http://TinyOS.stanford.edu/TinyOS/dists/ubuntu Hardy Heron" >>
/etc/apt/sources.list

sudo apt-get update
```

και εγκατάσταση των παρακάτω εργαλείων:

[1] NesC Compiler: `sudo apt-get install nesC`

[2] Debian MSP430: `sudo apt-get install msp430-binutils-TinyOS msp430-gcc-TinyOS msp430-libc-TinyOS`

[3] Debian AVR: `sudo apt-get install avr-binutils-TinyOS msp430-gcc-TinyOS msp430-libc-TinyOS`

Έχοντας πλοηγηθεί στον επιθυμητό φάκελο, η εγκατάσταση της τελευταίας διαθέσιμης έκδοσης του λειτουργικού TinyOS, 2.1.1[55][58] γίνεται με svn με την εντολή

```
svn checkout http://TinyOS-main.googlecode.com/svn/trunk/ TinyOS-2.x
```

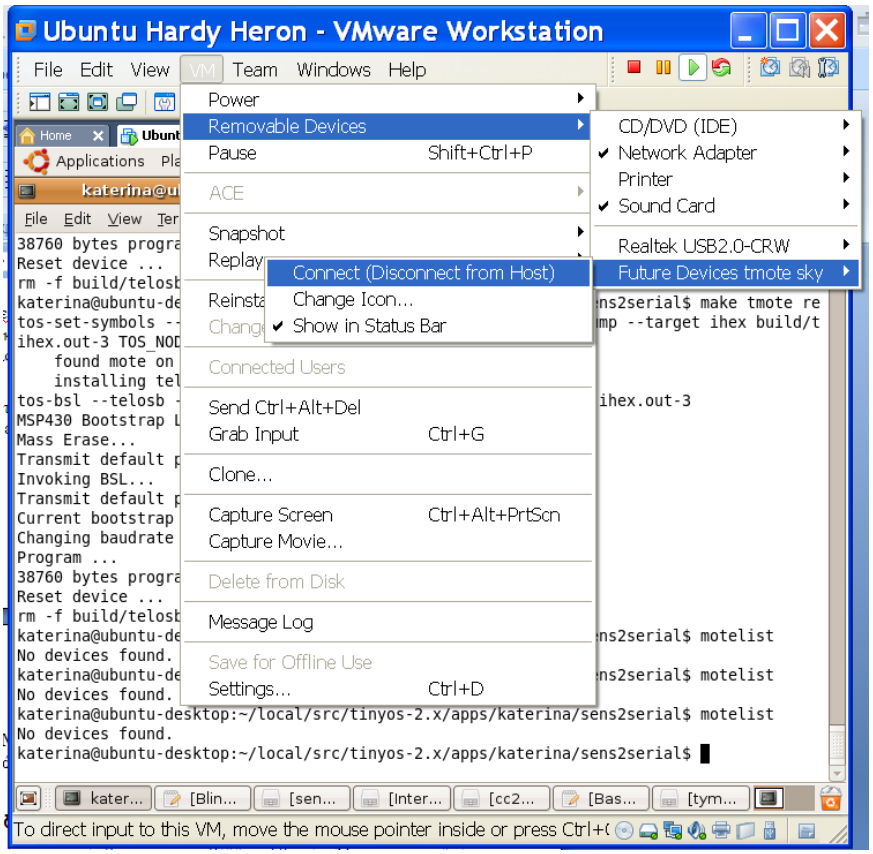
Για την ολοκλήρωση της εγκατάστασης χρειάζεται η μεταγλώττιση των TinyOS tools:

```
sudo apt-get install automake

cd TinyOS-2.x/tools

./Bootstrap

./configure --prefix=$HOME/local

make all

make install
```

και ο ορισμός των μεταβλητών περιβάλλοντος, προσθέτοντας τα παρακάτω στο αρχείο .bashrc:

```
export PATH=$HOME/local/bin:$PATH

export TOSROOT=$HOME/local/src/TinyOS-2.x

export TOSDIR=$TOSROOT/tos

export MAKERULES=$TOSROOT/support/make/Makerules

export CLASSPATH=$TOSROOT/support/sdk/java/TinyOS.jar:.

export PYTHONPATH=.:$TOSROOT/support/sdk/python:$PYTHONPATH

export PATH=$TOSROOT/support/sdk/c:$PATH
```

Μετά την εγκατάσταση όλων των απαραίτητων προγραμμάτων και αφού όλα τρέχουν σωστά, για τη διαχείριση του αισθητήρα χρειάζεται απλά η σύνδεση του σε μία θύρα USB και ενεργοποίηση της σύνδεσης της θύρας στο φιλοξενούμενο σύστημα, δηλαδή στο Ubuntu όπως φαίνεται στην εικόνα 5.1.



Εικόνα 5.1 Ενεργοποίηση της σύνδεσης της θύρας USB στο φιλοξενούμενο σύστημα Ubuntu

Έπειτα, για να εγκατασταθεί και να τρέξει μια εφαρμογή στον κόμβο-αισθητήρα ακολουθείται η εξής διαδικασία:

Αφού βρούμε που αντιστοιχούν αυτές οι συσκευές σε επίπεδο λειτουργικού (δίνοντας την εντολή 'motelist' ή την καθαρά unix-οειδή 'dmesg | tail'), μπαίνουμε διαδοχικά στους φακέλους που βρίσκονται οι εφαρμογές, τις μεταγλωττίζουμε και τις εγκαθιστούμε στους αισθητήρες. Για παράδειγμα:

Βάζουμε το πρώτο Tmote Sky σε μια θύρα. Δίνουμε σε ένα τερματικό:

```
Motelist
```

Βλέπουμε ως έξοδο:

```
Reference  Device           Description
---------- ---------------- ----------------------------------------------
M49WC0LI   /dev/ttyUSB0     Moteiv tmote sky
```

Άρα ξέρουμε ότι το Tmote Sky μας έχει αναγνωριστεί ως /dev/ttyUSB0.Ενδύκνειται να αλλάξουμε τα permissions της συσκευής:

```
sudo chmod 666 /dev/ttyUSB0
```

Πηγαίνουμε στον φάκελο της εφαρμογής που μας ενδιαφέρει:

```
cd /opt/TinyOS-2.1.1/apps/sampletoradio
```

Για να μεταγλωττίσουμε την εφαρμογή δίνουμε την εντολή:

```
make tmote
```

Επίσης μπορεί να χρησιμοποιηθεί:

```
make tmote reinstall, 1 bsl,/dev/ttyUSB0
```

Η παραπάνω εντολή μεταγλωττίζει την εφαρμογή sampletoradio και την εγκαθιστά στην συσκευή /dev/ttyUSB0 προσδίδοντάς της και την ονομασία "κόμβος 1" (reinstall, ***1***), ένα αναγνωριστικό κόμβου δηλαδή για λόγους διάκρισης. Στην περίπτωση που δεν δοθεί η επιλογή "bsl,/dev/ttyUSB0", το σύστημα θα χρησιμοποιήσει το bsl για να βρει την πρώτη μονάδα Tmote Sky που είναι συνδεδεμένη στο USB bus με την εντολή motelist (Εικόνα 5.2).

Εικόνα 5.2 Εκτέλεση της εντολής make tmote reisntall,2

Αν η εφαρμογή έχει ήδη μεταγλωττιστεί, χρησιμοποιώντας την εντολή:

```
make tmote install, 1 bsl,/dev/ttyUSB0
```

γίνεται εγκατάσταση της στην συσκευή /dev/ttyUSB0 προσδίδοντάς της και την ονομασία "κόμβος 1". Όμοια με παραπάνω το "bsl,/dev/ttyUSB0" μπορεί να παραληφθεί.

Πρέπει να τονίσουμε το γεγονός ότι αυτή η διαδικασία αρκεί να γίνει μια φορά, καθώς μετά το επιτυχές πέρας της ο κάθε αισθητήρας έχει εγκαταστημένη την ανάλογη εφαρμογή "για πάντα", η οποία τίθεται σε λειτουργία κάθε φορά που παρέχουμε ρεύμα στον αισθητήρα, είτε μέσω USB θύρας είτε μέσω μπαταριών. Μπορούμε φυσικά να κάνουμε εγκατάσταση οποιασδήποτε εφαρμογής "σβήνοντας" την παλιά.

5.1.1.2 Μονάδα-βάση συνδεδεμένη στον υπολογιστή

Η παραπάνω διαδικασία γίνεται, όπως αναφέρθηκε, σε περιβάλλον Ubuntu. Όμως το εργαλείο Listen.java, που είναι υπεύθυνο για την ανάγνωση των πακέτων που προέρχονται από τη μονάδα-βάση, χρησιμοποιείται σε περιβάλλον Windows, Συνεπώς είναι απαραίτητο να μπορεί να αναγνωριστεί η μονάδα βάση από τα Windows με την εγκατάσταση του απαραίτητου λογισμικού[57]. Μετά τις παρακάτω ρυθμίσεις θα είναι δυνατή η διαχείριση του Tmote Sky και μέσω των Windows αν το επιθυμούμε.

Αρχικά απαιτείται η εγκατάσταση των Java SDK[56], USB Serial Driver[59] και Cygwin[60]. Έπειτα με την εντολή rpm 'rpm –ivh <rpm>' στο Cygwin γίνεται η εγκατάσταση των εξής:

[1] TI MSP430 Tools

- msp430tools-base-0.1-20050607.cygwin.i386.rpm[61]
- msp430tools-python-tools-1.0-1.cygwin.noarch.rpm[62]
- msp430tools-binutils-2.16-20050607.cygwin.i386.rpm[63]
- msp430tools-gcc-3.2.3-20050607.cygwin.i386.rpm[64]
- msp430tools-libc-20080808-1.cygwin.i386.rpm[65]

[2] TinyOS-specific Tools

- nesC-1.3.1-1.cygwin.i386.rpm[66] (rpm –Uvh)
- TinyOS-deputy-1.1-1.cygwin.i386.rpm[67] (rpm –i)
- TinyOS-tools-1.4.0-3.cygwin.i386.rpm[68] (rpm -Uvh)

[3] TinyOS 2.x source tree

[4] TinyOS-2.1.1-3.cygwin.noarch.rpm[69]

Τέλος πρέπει να τεθούν οι μεταβλητές περιβάλλοντος:

TOSROOT :/opt/TinyOS-2.x

TOSDIR:$TOSROOT/tos

CLASSPATH:C:\cygwin\opt\TinyOS-2.x\support\sdk\java\TinyOS.jar;

MAKERULES:$TOSROOT/support/make/Makerules

PATH:/opt/msp430/bin:/opt/jflashmm:$PATH

MOTECOM: serial@COM16:tmote

Αυτό μπορεί να γίνει με την εντολή export <όνομα μεταβλητής>=`cygpath -w <τιμή μεταβλητής>` μέσα από το Cygwin.

Η μεταβλητή MOTECOM λέει στο εργαλείο Listen.java ποια πακέτα πρέπει να «ακούει».

5.1.2 Ανάπτυξη εφαρμογής

Έχοντας ολοκληρώσει την εγκατάσταση των απαραίτητων στοιχείων, το επόμενο βήμα είναι η ανάπτυξη της εφαρμογής. Όπως έχει αναφερθεί στα προηγούμενα, πρέπει να δημιουργηθούν δύο αρχεία, ένα module και ένα configuration, τα SampleToRadioAppC.nc και SampleToRadioC.nc αντίστοιχα. Συμπληρωματικά, χρειάζεται το αρχείο SampleToRadio.h για τον ορισμό δομών και σταθερών που θα χρησιμοποιηθούν στην εφαρμογή.

Το αρχείο SampleToRadioApp.nc περιέχει την απαραίτητη «καλωδίωση» της εφαρμογής, δηλαδή τις διεπαφές που χρησιμοποιούνται:

- LedsC: άναμμα και σβήσιμο των Leds
- TimerMilliC(): χρονόμετρο
- ActiveMessageC: ασύρματη επικοινωνία
- DymoNetworkC: πρωτόκολλο Tymo για ασύρματη επικοινωνία
- SerialActiveMessageC, SerialAMSenderC(AM_SAMPLETORADIO), PoolC(message_t, 10), QueueC(message_t*, 10) : Σειριακή επικοινωνία
- SensirionSht11C(): μέτρηση θερμοκρασίας και υγρασίας
- HamamatsuS1087ParC: μέτρηση PAR
- HamamatsuS10871TsrC: μέτρηση TSR

- Msp430InternalVoltageC: μέτρηση εσωτερικής τάσης
- Msp430InternalTemperatureC: μέτρηση εσωτερικής θερμοκρασίας
- CC2420ActiveMessageC: μέτρηση rssi

Σύμφωνα με το 4.4.2, κάθε εφαρμογή που αναπτύσσεται στο TinyOS έχει τη δυνατότητα χρήσης του πεδίου payload της δομής του ενεργού μηνύματος message_t για αποθήκευση δεδομένων. Το μέγιστο μέγεθος αποθηκευμένων δεδομένων είναι ίσο με TOSH_DATA_LENGTH με προεπιλογή είναι 28 (14 byte). Συνεπώς η εφαρμογή SampleToRadio θα χρησιμοποιήσει το συγκεκριμένο πεδίο για την αποστολή των επιθυμητών δεδομένων τα οποία είναι (εικόνα 5.3 και 5.4):

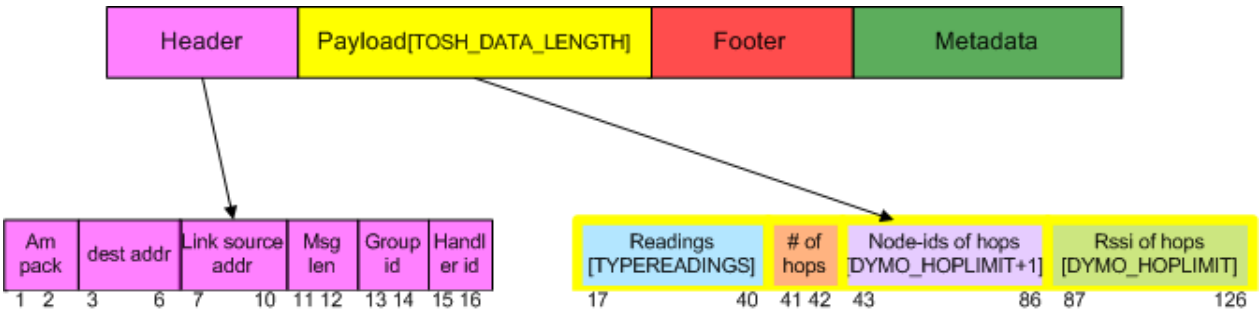
[1] Περιβαλλοντικά δεδομένα(6*2=12 bytes)

[2] Αριθμός βημάτων(hops) (1 byte)

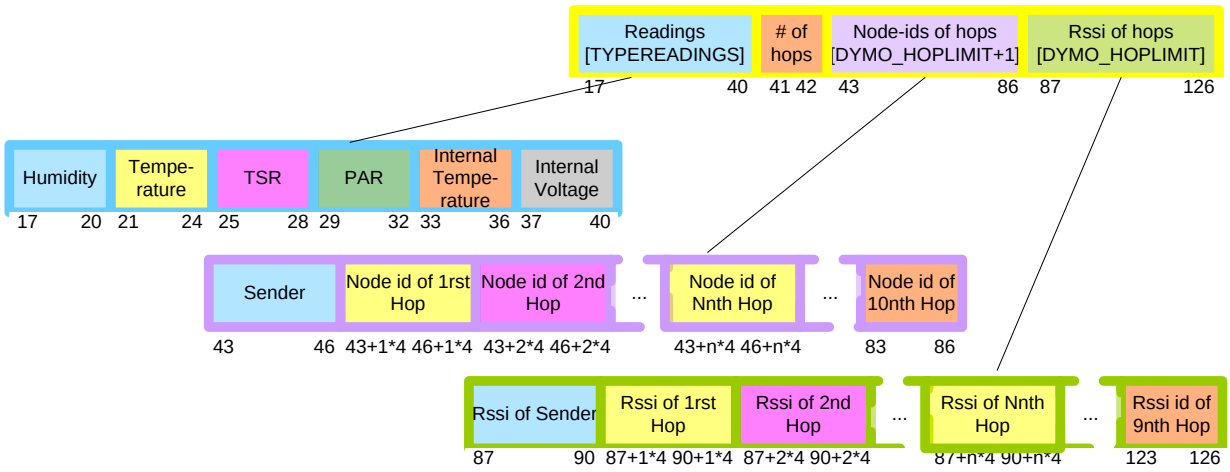
[3] Id μονάδας-αισθητήρα κάθε βήματος (το πρώτο θα είναι ο αποστολέας και το τελευταίο η μονάδα-βάση)

[4] Rssi που καταγράφει η μονάδα-αισθητήρας κάθε βήματος εκτός από τον αποστολέα

Επιλέγοντας ο μέγιστος αριθμός βημάτων να είναι 10, πρέπει το πεδίο [3] να μπορεί να αποθηκεύσει 11 καταχωρήσεις id (10 hops+ο αποστολέας) και το πεδίο [4] 10. Γνωρίζοντας ότι το id μιας μονάδας και οι μετρήσεις του rssi είναι 2 κάθε μία, το [3] θα είναι 11*2=22 bytes και το [4] 10*2=20. Συνεπώς είναι αναγκαίο να γίνει το TOSH_DATA_LENGTH ίσο με τουλάχιστον 12+1+22+20=55*2=110, όσο και το μέγεθος του μηνύματος που θα χρησιμοποιηθεί. Τα προηγούμενα φαίνονται αναλυτικά στις εικόνες 5.3 και 5.4.



Εικόνα 5.3 Η δομή message_t

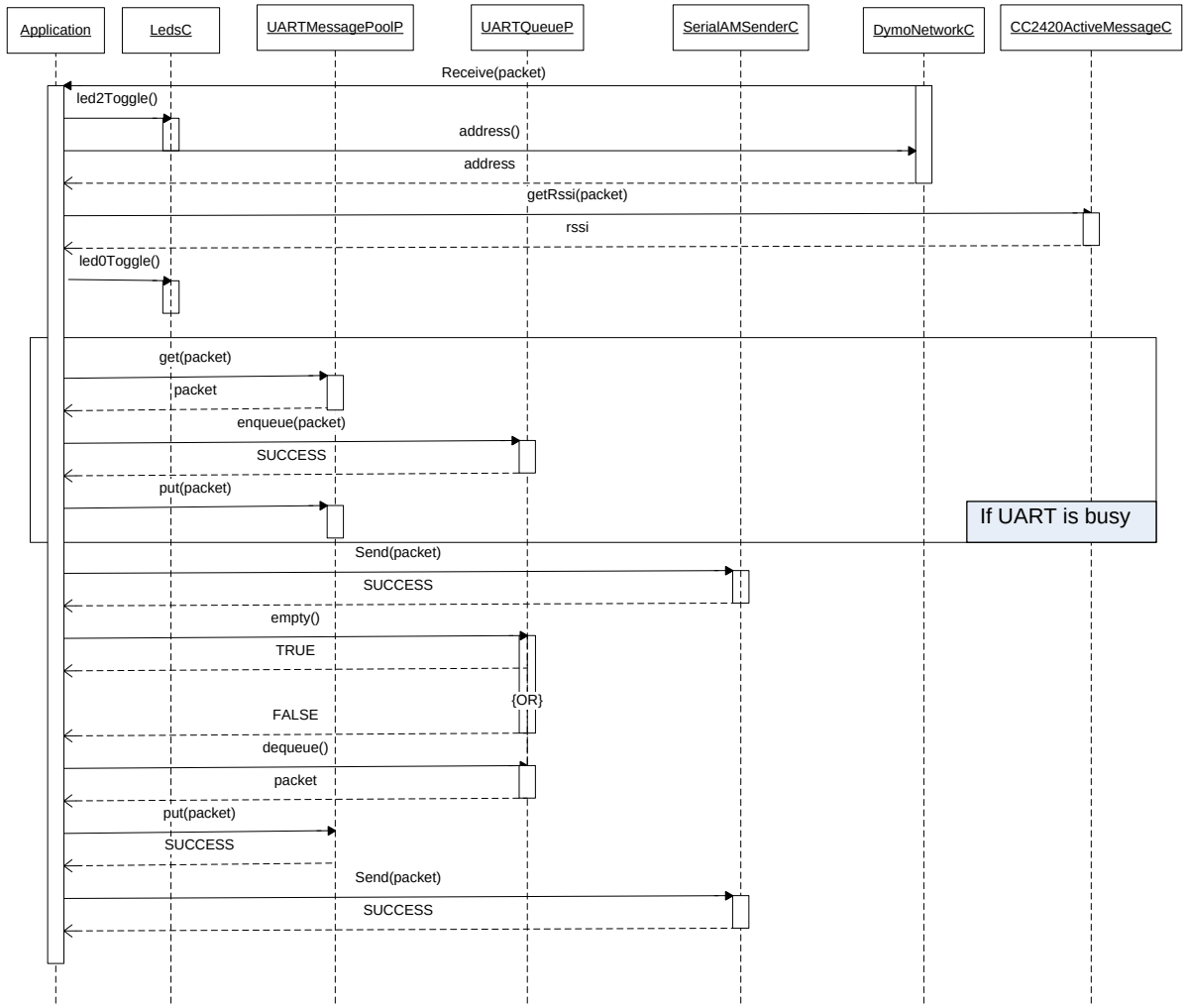


Εικόνα 5.4 Ανάλυση του πεδίου Payload του message_t

Η σταθερά TOSH_DATA_LENGTH βρίσκεται στο αρχείο CC2420.h (διαδρομή αρχείου $TOSROOT/tos/chips/CC2420.h). Επιλέγεται η τιμή 126 για να υπάρχει περιθώριο αύξησης του μηνύματος αν χρειαστεί. Επίσης χρειάζεται να δηλωθεί ο μέγιστος αριθμός βημάτων και γι'αυτό το σκοπό η σταθερά DYMO_HOPLIMIT του αρχείου routing.h ($TOSROOT/tos/lib/net/tymo/routing.h) γίνεται ίση με 1 0 Η δήλωση του περιγραφέντος μηνύματος γίνεται στο αρχείο SampleToRadio.h και είναι η παρακάτω:

```
typedef nx_struct SampleToRadioMsg {

    nx_uint16_t readings[TYPEREADINGS]; /*Pinakas me metriseis diastasis ari8mos metrisewn ana paketo pou stelnetai epi ari8mos typwn metrisewn(temp,hum etc)*/

    nx_uint8_t hops;

    nx_uint16_t nodeid[DYMO_HOPLIMIT+1]; //max 10 hops-11 nodes

    nx_uint16_t rssi[DYMO_HOPLIMIT];  //msg sent max 10 times

} SampleToRadioMsg;
```

Το SampleToRadioC.nc περιέχει τον κώδικα τον οποίο εκτελεί η εφαρμογή. Αρχικά πρέπει να αναφερθεί ότι χρησιμοποιείται η σύμβαση ότι ο κόμβος-βάση έχει id ίσο με 500, ενώ δεν υπάρχει περιορισμός για το id των υπολοίπων κόμβων, εκτός του ότι πρέπει να είναι το πολύ 2 bytes. Εφόσον όλη η εφαρμογή έχει κοινό κώδικα, δηλαδή δεν εκτελείται διαφορετικός κώδικας στη μονάδα-βάση από ότι στη μονάδα-κόμβο, ο χρήστης της εφαρμογής έχει τη δυνατότητα να ορίσει αυτός ποια θα είναι η βάση, απλά δίνοντας το id 500 στην επιθυμητή μονάδα κατά την εγκατάσταση της εφαρμογής σε αυτό.



Εικόνα 5.5 Ακολουθιακό UML διάγραμμα για τη μονάδα-βάση

Παραπάνω φαίνεται το ακολουθιακό διαγράμμα για την εφαρμογή όπως εκτελείται στη μονάδα-βάση. Η περιγραφή της είναι η εξής:

Το πρώτο βήμα για την υλοποίηση είναι με την εκκίνηση της εφαρμογής (γεγονός Boot.booted()), να γίνεται η αρχικοποίηση της ασύρματης και της σειριακής επικοινωνίας και των μεταβλητών κατάστασης. Όταν ολοκληρωθεί η αρχικοποίηση της σειριακής επικοινωνίας, γίνεται έλεγχος αν πρόκειται για τη μονάδα-βάση (id=500), οπότε και γίνεται TRUE το flag rootme. Στην αντίθετη περίπτωση ξεκινάει ένα χρονόμετρο που θα σηματοδοτεί τη τη δειγματοληψία των αισθητήρων τις μονάδας.

Έγινε η επιλογή η μονάδα βάση να μην παίρνει μετρήσεις, αλλά μόνο να λαμβάνει πακέτα και να τα προωθεί στη σειριακή θύρα, για να μην έχει παραπάνω φόρτο εργασίας. Έτσι όταν γίνεται η λήψη ενός πακέτου από τη βάση γίνονται τα εξής:

→ Τροποποίηση του ληφθέντος πακέτου:

- αύξηση του αριθμού των βημάτων
- υπολογισμός του rssi του τελευταίου βήματος
- καταχώρηση του id της βάσης ως τελευταίο βήμα

→ Γίνεται εναλλαγή της λειτουργίας του led 2 (Αν είναι σβηστό ανάβει και αν είναι αναμμένο σβήνει).

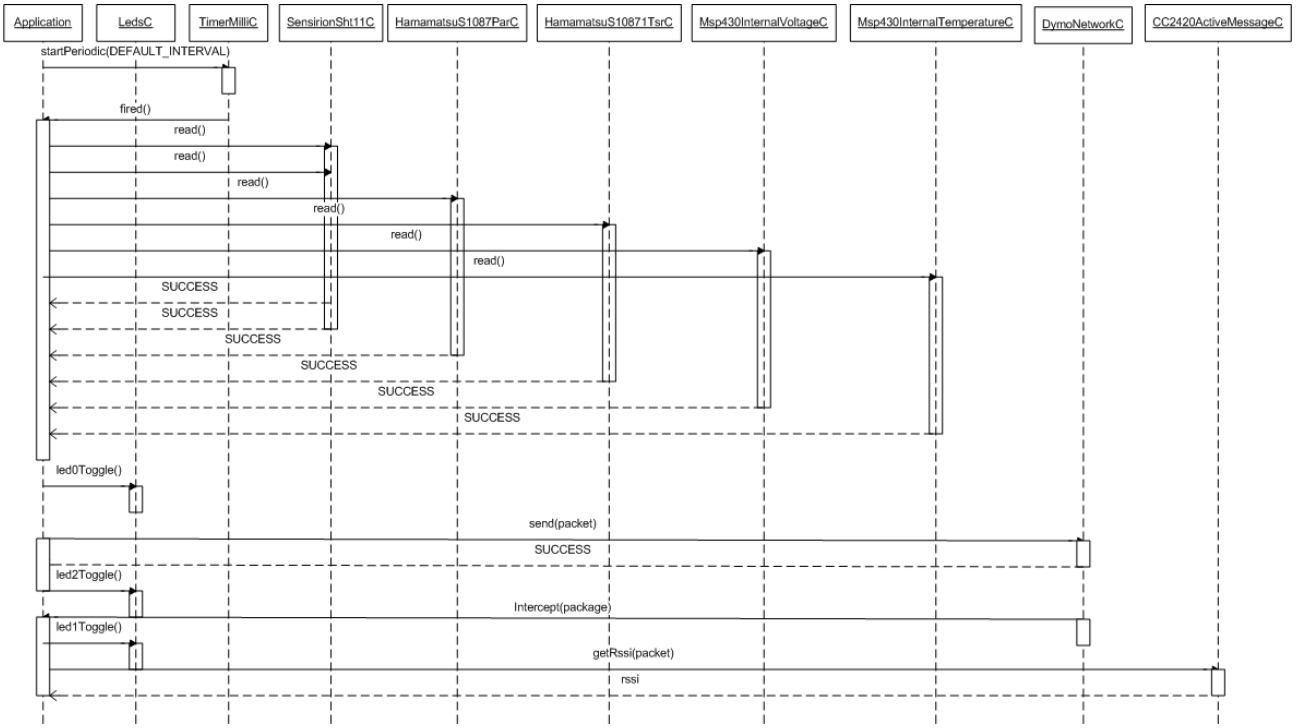
→ Αν δεν γίνεται ήδη κάποια αποστολή, το πακέτο προωθείται στη σειριακή θύρα

→ Γίνεται εναλλαγή της λειτουργίας του led 0.

→ Αν η σειριακή θύρα είναι απασχολημένη με άλλη αποστολή, το πακέτο μπαίνει σε μια ουρά για να γίνει η προώθηση του αργότερα.

→ Με την ολοκλήρωση μιας αποστολής προς τη σειριακή θύρα (γεγονός SerialSend.sendDone), ελέγχεται αν υπάρχει πακέτο προς αποστολή στην ουρά, οπότε ξεκινάει μία νέα αποστολή, αλλιώς αναμένεται η άφιξη ενός νέου πακέτου.

Στην περίπτωση που υπάρξει κάποιο σφάλμα κατά την παραπάνω διαδικασία, τα led ανάβουν για ειδοποίηση του χρήστη.



Εικόνα 5.6 Ακολουθιακό UML διάγραμμα για τη μονάδα-κόμβο.

Παραπάνω φαίνεται το ακολουθιακό διαγράμμα για την εφαρμογή όπως εκτελείται στις μονάδες-κόμβους. Η περιγραφή της είναι η εξής:

→ Το χρονόμετρο προκαλεί το γεγονός Timer0.fired() κάθε DEFAULT_INTERVAL, που είναι και η περίοδος δειγματοληψίας και έχει οριστεί ίσο με 1024, δηλαδή 1 sec.

→ Γίνεται μέτρηση των ζητούμενων μεγεθών

→ Με την ολοκλήρωση και της τελευταίας μέτρησης (το πέρας καθεμιάς δίνεται από ένα αντίστοιχο γεγονός) αυξάνεται μετρητής reading.

→ Η παραπάνω διαδικασία ολοκληρώνεται μέχρις ότου ο reading φτάσει την τιμή NREADINGS που έχει τεθεί ίσο με 10, γεγονός που σημαίνει ότι έχουν ληφθεί 10 μετρήσεις για κάθε μέγεθος.

→ Γίνεται εναλλαγή της λειτουργίας του led 0.

→ Υπολογίζεται ο μέσος όρος των μετρήσεων, αποθηκεύεται στο πακέτο προς αποστολή και μηδενίζεται ο παραπάνω μετρητής.

→ Προωθείται το πακέτο ασύρματα με προορισμό τη μονάδα-βάση.

→ Σε περίπτωση αποτυχίας ανάβει το led 0 και επιχειρείται νέα αποστολή μέχρι και RETRY_NO που έχει οριστεί ίσο με 3.

→ Σε περίπτωση επιτυχίας γίνεται εναλλαγή της λειτουργίας του led 2.

→ Όταν ένας κόμβος λάβει ένα πακέτο που δεν προορίζεται για αυτόν, είναι δηλαδή ένας ενδιάμεσος κόμβος:

- γίνεται εναλλαγή της λειτουργίας του αύξηση του αριθμού των βημάτων
- υπολογισμός του rssi του βήματος
- καταχώρηση του id του κόμβου για αυτό βήμα
- προώθηση του πακέτου στο επόμενο βήμα

Για την υλοποίηση της ασύρματης επικοινωνίας χρησιμοποιήθηκε η εφαρμογή Tymo που υλοποιεί τη συνάρτηση MHSend για multi-hop επικοινωνία στο δίκτυο.

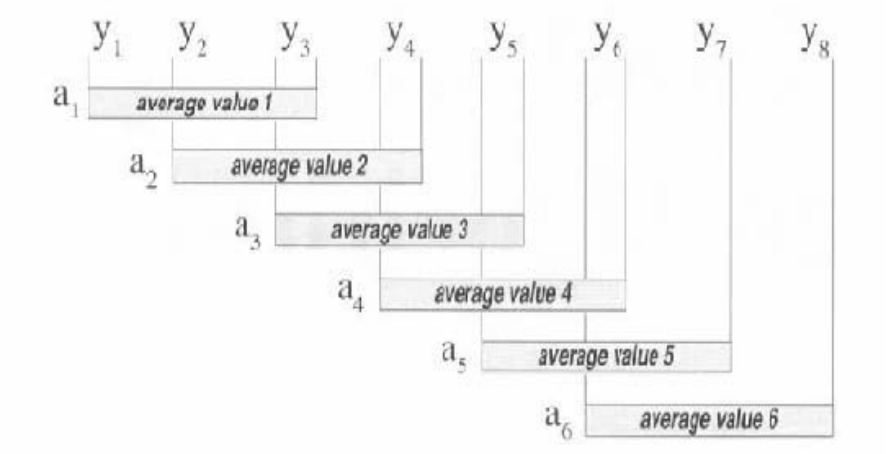
Για να μπορέσει να δουλέψει η εφαρμογή και να γίνει σωστά η «καλωδίωση» των διεπαφών, πρέπει στο φάκελο της εφαρμογής να δημιουργηθεί ένα αρχείο με το όνομα Makefile στο οποίο δηλώνεται πιο αρχείο είναι component καθώς και τις τοποθεσίες των διεπαφών που χρησιμοποιούνται. Όλα τα παραπάνω αρχεία υπάρχουν στα παραρτήματα.

5.2 Επίπεδο σχεδιασμού: Βάση Δεδομένων

5.2.1 Το φίλτρο 'Κινούμενου Μέσου Όρου' (Moving Average Filter)

Ο κινούμενος μέσος όρος είναι μια απλή μαθηματική τεχνική η οποία χρησιμοποιείται κατά κύριο λόγο για τη μείωση της απόκλισης και την ανάδειξη της τάσης σε μια συλλογή από σημεία δεδομένων. Ο κινούμενος μέσος όρος αποτελεί παράλληλα πρωτότυπο ενός FIR φίλτρου, του πιο κοινού φίλτρου που χρησιμοποιείται ευρέως σε υπολογιστικά όργανα μέτρησης. Σε περιπτώσεις θορυβώδους σήματος, όταν υπάρχει η επιθυμία να βγάλουμε τη μέση τιμή από ένα περιοδικό σήμα ή ακόμα όταν μια αργή μετατόπιση βασικής γραμμής πρέπει να εξαλειφθεί, τότε το φίλτρο κινούμενου μέσου όρου (moving average filter) μπορεί να εφαρμοστεί για να φέρει τα επιθυμητά αποτελέσματα.

Η διαδικασία που χρησιμοποιεί ο αλγόριθμος για να καθοριστεί το μέγεθος του φιλτραρίσματος, περιλαμβάνει τη χρήση ενός παράγοντα εξομάλυνσης (παράθυρο s σημείων). Αυτός ο παράγοντας μπορεί να αυξηθεί ή να ελαττωθεί ανάλογα με τον αριθμό των πραγματικών τιμών της κυματομορφής ή των δειγμάτων που θα χρησιμοποιήσει ο αλγόριθμος. Κάθε περιοδική κυματομορφή μπορεί να θεωρηθεί σαν μια σειρά ή συλλογή από τιμές δεδομένων. Ο αλγόριθμος υπολογίζει τον κινούμενο μέσο όρο παίρνοντας 2 ή περισσότερες τιμές της κυματομορφής, προσθέτοντας τις, διαιρώντας το άθροισμά τους με το συνολικό αριθμό των προστιθέμενων τιμών, αντικαθιστώντας την πρώτη τιμή με το μέσο όρο που μόλις υπολογίστηκε και επαναλαμβάνοντας τα βήματα με τη δεύτερη, τρίτη κ.ο.κ. τιμή, μέχρι την τελευταία τιμή της κυματομορφής. Το αποτέλεσμα είναι μια παραγόμενη κυματομορφή που συνίσταται από τις μέσες τιμές των δεδομένων και έχει τον ίδιο αριθμό σημείων με την αρχική κυματομορφή.



Εικόνα 5.7 Παράδειγμα εφαρμογής moving average με παράθυρο 3.

Η Εικόνα 5.5 παρουσιάζει πώς εφαρμόζεται ο αλγόριθμος του κινούμενου μέσου όρου στα σημεία y μιας κυματομορφής. Με y σημειώνονται τα συνεχόμενα σημεία της κυματομορφής ώστε να φανεί πώς υπολογίζεται ο κινούμενος μέσος όρος. Στην περίπτωση αυτή, εφαρμόστηκε ο παράγοντας εξομάλυνσης τριών σημείων, που σημαίνει ότι 3 συνεχόμενα σημεία της αρχικής κυματομορφής προστέθηκαν, το άθροισμά τους διαιρέθηκε δια του 3 και το πηλίκο αυτό αποτέλεσε το πρώτο σημείο της παραγόμενης κυματομορφής. Η διαδικασία επαναλαμβάνεται για το δεύτερο, το τρίτο

κ.ο.κ. σημείο της αρχικής κυματομορφής, μέχρι το τελευταίο σημείο της. Είναι αξιοσημείωτη η επικάλυψη που συμβαίνει κατά τον υπολογισμό του κινούμενου μέσου όρου. Αυτή η τεχνική της επικάλυψης, μαζί με μια ειδική μεταχείριση του αρχικού και τελικού σημείου, δίνει τον ίδιο αριθμό σημείων στην παραγόμενη κυματομορφή, με αυτόν της αρχικής.

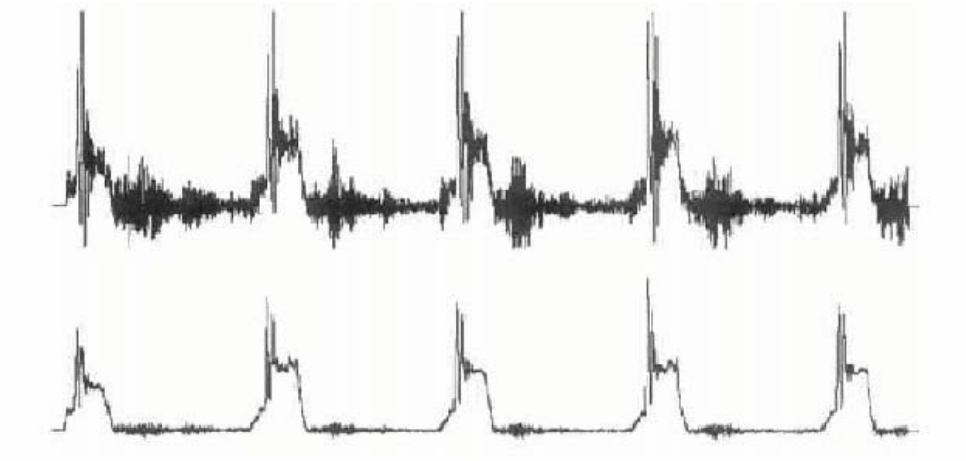
Ο κινούμενος μέσος όρος μιας κυματομορφής μπορεί να υπολογιστεί από τον τύπο:

$$a(n) = \frac{1}{s} \cdot \sum_{i=n}^{n+(s-1)} y(i)$$

όπου α = η μέση τιμή
n = η θέση του σημείου
s = ο παράγοντας εξομάλυνσης
y = η πραγματική τιμή του σημείου

Η ευελιξία του αλγορίθμου βασίζεται στο μεγάλος εύρος επιλογών που υπάρχουν για τον παράγοντα εξομάλυνσης. Ο παράγοντας αυτός καθορίζει πόσα πραγματικά σημεία θα ληφθούν υπόψη για την εξαγωγή του μέσου όρου. Κάθε θετικός παράγοντας εξομάλυνσης έχει ως αποτέλεσμα ένα βαθυπερατό φίλτρο ενώ κάθε αρνητικός, προσομοιώνει ένα υψιπερατό φίλτρο. Δεδομένης της απόλυτης τιμής του παράγοντα εξομάλυνσης, οι υψηλές τιμές δίνουν μεγαλύτερη εξομάλυνση στην παραγόμενη κυματομορφή ενώ οι χαμηλές τιμές παρέχουν μικρότερη εξομάλυνση. Εφαρμόζοντας τον κατάλληλο παράγοντα εξομάλυνσης, ο αλγόριθμος μπορεί να χρησιμοποιηθεί για την εξαγωγή της μέσης τιμής μιας περιοδικής κυματομορφής. Μεγαλύτεροι θετικοί παράγοντες εξομάλυνσης συνήθως εφαρμόζονται για την παραγωγή μεσαίων τιμών

Ένα παράδειγμα εφαρμογής του φίλτρου κινούμενου μέσου όρου, για τη μείωση του θορύβου σε ένα σήμα, φαίνεται στην επόμενη εικόνα.



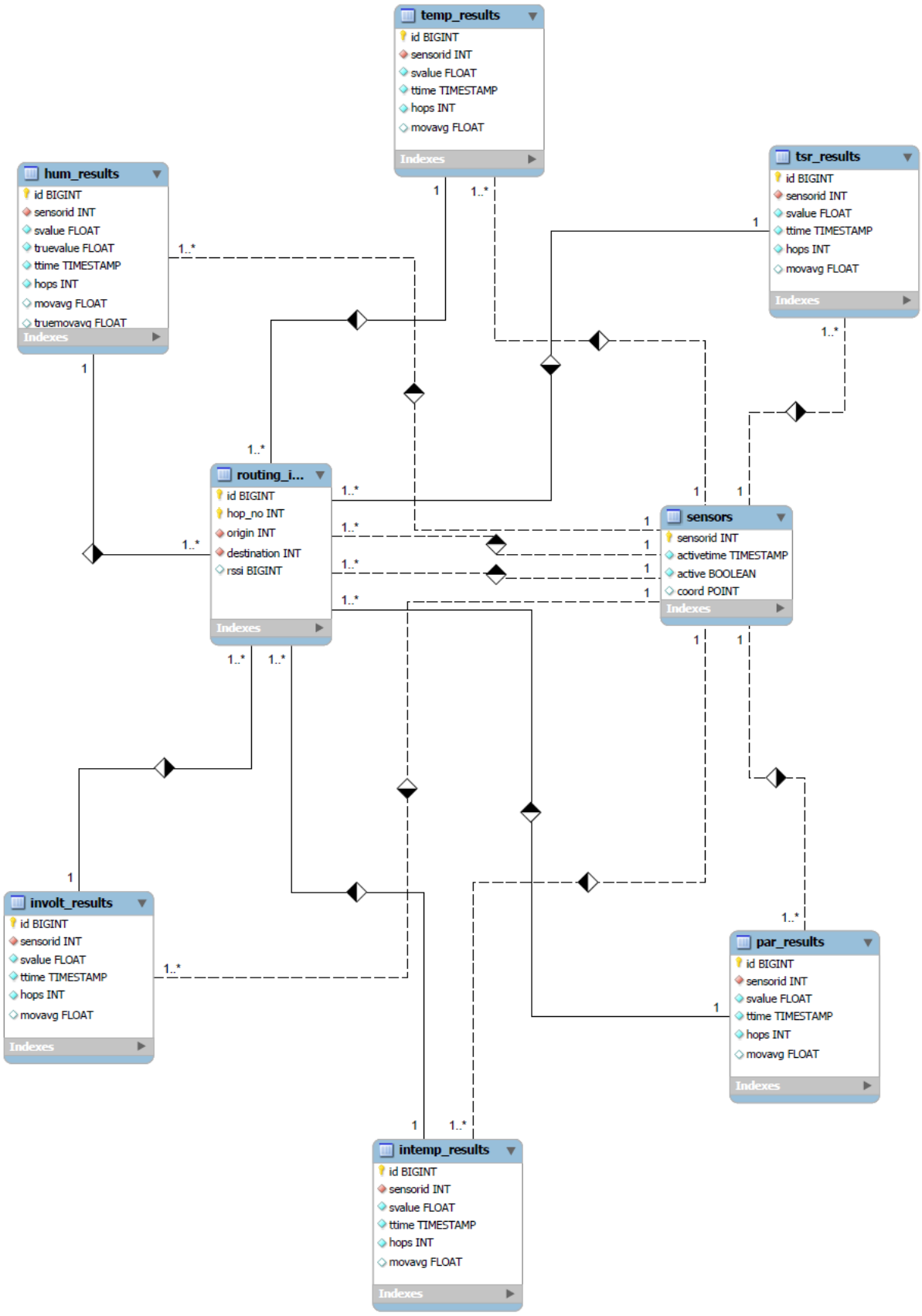
Εικόνα 5.8 Παράδειγμα εφαρμογής φίλτρου κινούμενου μέσου όρου για την ελαχιστοποίηση του θορύβου του σήματος.

Η πρώτη κυματομορφή στη παραπάνω εικόνα είναι η έξοδος ενός μετρητή πίεσης που χρησιμοποιείται για την παρακολούθηση της συμπίεσης προϊόντων σε μια διαδικασία πακεταρίσματος. Ο θόρυβος που εμφανίζεται οφείλεται στις έντονες δονήσεις που δημιουργούνται από την πρέσα κατά τη διαδικασία. Η δεύτερη κυματομορφή είναι φιλτραρισμένη χρησιμοποιώντας τον αλγόριθμο κινούμενου μέσου όρου 11 σημείων. Το αποτέλεσμα είναι μια πολύ πιο καθαρή εικόνα.

Η υλοποίηση του φίλτρου σε κώδικα έγινε σε Java και MySQL στο εργαλείο Listen.java και περιγράφεται στο 5.3.3. Η τιμή του αποθηκεύεται στη βάση όπως περιγράφεται στο 5.3.2. Επιλέχθηκε παράγοντας εξομάλυνσης ίσος με s=10.

5.2.2 Βάση Δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων που στέλνουν οι αισθητήρες δημιουργήθηκε με MySQL στο server 95.211.25.40. Παρακάτω παρατίθεται το σχεσιακό μοντέλο της βάσης καθώς και μια περιγραφή των πινάκων και των πεδίων τους. Στην εικόνα με κίτρινο θαυμαστικό σημειώνεται το πρωτεύων κλειδί (έντονη γραφή στην περιγραφή) κάθε πίνακα και με κόκκινο ρόμβο το ξένο κλειδί(πλάγια γραφή στην περιγραφή).



Εικόνα 5.9 Σχεσιακό μοντέλο της Βάσης Δεδομένων

- sensors: Περιλαμβάνει όλους τους αισθητήρες που υπάρχουν στο σύστημα και όσους υπήρξαν στο παρελθόν
    - **sensorid**: Το αναγνωριστικό της μονάδας-αισθητήρα.
    - activetime: Η ημερομηνία και ώρα που ήταν τελευταία φορά ενεργός (έστειλε πακέτο με μετρήσεις) η μονάδα-αισθητήρας.
    - active: TRUE αν ο αισθητήρας ήταν ενεργός μέσα στις προηγούμενες 2 ώρες, FALSE στην αντίθετη περίπτωση.
    - coord: Οι συντεταγμένες της θέσης του αισθητήρα
- routing_info: Περιλαμβάνει πληροφορίες για τα βήματα(hops) που ακολούθησε κάθε πακέτο μέχρι να φτάσει στη μονάδα-βάση.
    - ***id*: Το αναγνωριστικό του πακέτου.**
    - **hop_no: Το νούμερο του βήματος**
    - *origin: Η μονάδα-αισθητήρας που προώθησε το πακέτο*
    - *destination: Η μονάδα-αισθητήρας που έλαβε το πακέτο*
    - rssi: Το rssi που μέτρησε η μονάδα-αισθητήρας που έλαβε το πακέτο.
- temp_results: Περιλαμβάνει στοιχεία για τις μετρήσεις θερμοκρασίας.
    - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
    - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*
    - svalue: Η τιμή του μεγέθους.
    - ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
    - hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
    - movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.
- hum_results: Περιλαμβάνει στοιχεία για τις μετρήσεις υγρασίας.
    - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
    - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*
    - svalue: Η τιμή του μεγέθους.

- truevalue: Η πραγματική τιμή της υγρασίας, αν συνυπολογιστεί και η θερμοκρασία.
- ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
- hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
- movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.
- truemovavg: ο moving average των πραγματικών μετρήσεων, δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων true value.

- tsr_results: Περιλαμβάνει στοιχεία για τις μετρήσεις TSR.
  - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
  - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*
  - svalue: Η τιμή του μεγέθους.
  - ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
  - hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
  - movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.
- par_results: Περιλαμβάνει στοιχεία για τις μετρήσεις PAR.
  - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
  - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*
  - svalue: Η τιμή του μεγέθους.
  - ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
  - hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
  - movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.
- involt_results: Περιλαμβάνει στοιχεία για τις μετρήσεις εσωτερικής τάσης.
  - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
  - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*

- svalue: Η τιμή του μεγέθους.
- ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
- hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
- movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.

- intemp_results: Περιλαμβάνει στοιχεία για τις μετρήσεις εσωτερικής θερμοκρασίας
  - **id: Το αναγνωριστικό του πακέτου που περιείχε τη μέτρηση.**
  - *sensorid: Το αναγνωριστικό της μονάδας-αισθητήρα που πήρε τη μέτρηση.*
  - svalue: Η τιμή του μεγέθους.
  - ttime: Η ημερομηνία και ώρα που έγινε η μέτρηση.
  - hops: Ο αριθμός των βημάτων που χρειάστηκαν για να φτάσει η μέτρηση στη μονάδα βάση.
  - movavg: ο moving average δηλαδή ο μέσος όρος των τελευταίων 10 μετρήσεων.

Για να διασφαλιστεί η ακεραιότητα της βάσης έχουν αξιοποιηθεί τα εξής:

— Τύποι δεδομένων: Με την επιλογή των κατάλληλων τύπων δεδομένων για την αποθηκεύσει των μετρήσεων διασφαλίζεται ότι θα γίνει σωστή αναπαράσταση τους στη βάση.

— Πρωτεύοντα κλειδιά: Διασφαλίζεται ότι κάθε πακέτο θα έχει μοναδικό αναγνωριστικό κατά την καταχώρησή του και ότι θα είναι δυνατό να βρεθούν και οι έξι τύποι μετρήσεων που περιείχε, αφού θα έχουν το ίδιο αναγνωριστικό.

— Ξένα κλειδιά: Για να γίνει η εισαγωγή δεδομένων σε πίνακα με εξάρτηση ξένου κλειδιού πρέπει πρώτα η τιμή του πεδίου με την εξάρτηση να υπάρχει στον πίνακα από τον οποίο εξαρτάται. Επίσης κατά τη διαγραφή μιας εγγραφής ενός πίνακα γίνεται διαγραφή και των αντίστοιχων εγγραφών με αυτήν την τιμή στον εξαρτώμενο πίνακα. Έτσι δεν θα υπάρχουν πότε εγγραφές «ξεκρέμαστες».

— Triggers: Κατά την εισαγωγή νέων εγγραφών στους πίνακες temp_results, hum_results, tsr_results, par_results, involt_results και intemp_results γίνεται έλεγχος για το αν οι μονάδες αισθητήρες που βρίσκονται στη βάση ήταν ενεργοί μέσα στις προηγούμενες 2 ώρες, αλλιώς η τιμή του πεδίου sensors.active γίνεται FALSE. Από αυτό τον έλεγχο εξαιρείται η μονάδα-βάση, αφού είναι σίγουρα ενεργή, εφόσον έχει καταχωρηθεί νέα εγγραφή στη βάση.

5.2.3 Το εργαλείο Listen

Η εφαρμογή Listen.java είναι ένα βασικό κομμάτι του συστήματος που υλοποιείται γιατί πραγματοποιεί τη «γεφύρωση» ανάμεσα στο ασύρματο δίκτυο αισθητήρων και τη βάση δεδομένων. Αναλυτικά οι εργασίες που εκτελεί είναι οι εξής:

→ Ανάγνωση Πακέτων την μονάδας-βάση από τη θύρα USB(το σύστημα τη βλέπει σα σειριακή θύρα).

→ Μετατροπή της αλληλουχίας byte που ανακτά σε String από δεκαεξαδικά ψηφία.

→ Διαχωρισμός του κάθε πεδίου του πακέτου και μετατροπή του σε δεκαδική μορφή.

→ Μετατροπή των περιβαλλοντικών μετρήσεων από raw values σε SI σύμφωνα με τον πίνακα 3-4.

→ Εκτύπωση στο τερματικό τόσο του αρχικού πακέτου όσο και των δεδομένων αφού έχουν μετατραπεί.

→ Εγκατάσταση σύνδεσης με τη Βάση Δεδομένων.

→ Υπολογισμός moving average για κάθε μέγεθος.

→ Καταχώρηση στη Β.Δ. του id του αισθητήρα αν δεν υπάρχει ήδη ή ενημέρωση του πεδίου του activetime με την παρούσα ημερομηνία και ώρα στην αντίθετη περίπτωση.

→ Καταχώρηση στη βάση όλων των δεδομένων που έχουν συλλεχθεί μαζί με ημερομηνία, ώρα και moving average.

Σε κάθε πακέτο που καταφθάνει από τον αισθητήρα-βάση ανατίθεται ένα μοναδικό id. Αυτό το id είναι κοινό για κάθε τύπο μέτρησης που περιέχει αυτό το πακέτο. Συνεπώς αν ένα πακέτο έχει id 34, στη βάση θα υπάρχουν καταχωρήσεις στους πίνακες temp_results, hum_results, tsr_results, par_results, involt_results και intemp_results με id 34, οι οποίες προέρχονται από αυτό το πακέτο. Όμως αυτό το id θα είναι μοναδικό για κάθε πίνακα, αφού είναι και το πρωτεύον κλειδί.

5.3 Επίπεδο σχεδιασμού: Server

Στο site θα αποθηκευτούν τα απαραίτητα αρχεία για την δημιουργία του ιστότοπου όπου θα γίνει η παρουσίαση της εφαρμογής και των αποτελεσμάτων. Επιπλέον η βάση δεδομένων που χρησιμοποιείται είναι αποθηκευμένη εκεί. Ο server είναι ο 95.211.25.40.

5.3.1 Ανάκτηση δεδομένων από τη Βάση Δεδομένων

Πριν την γραφική αναπαράσταση, πρέπει να ανακτηθούν τα αποθηκευμένα δεδομένα από τη βάση. Για αυτή την εργασία επιλέχθηκε η γλώσσα PHP η οποία συνεργάζεται καλά με τη MySQL. Οι λειτουργίες που εκτελεί είναι απλές: σύνδεση στη βάση, ανάκτηση δεδομένων σύμφωνα με το ερώτημα που έχει οριστεί και αποθήκευσή τους σε μεταβλητές, έτσι ώστε να είναι έτοιμα για χρήση.

5.3.2 Γραφική αναπαράσταση δεδομένων

Αφού τα δεδομένα βρίσκονται έτοιμα, αποθηκευμένα σε μεταβλητές, το επόμενο βήμα είναι η παρουσίαση τους στο χρήστη, η οποία γίνεται με τη χρήση Google Charts Tools[70]. Το Chart API παρέχει έναν απλό τρόπο για διαγραμμάτων-εικόνων διαφόρων τύπων, στέλνοντας μια μορφοποιημένη διεύθυνση URL που περιλαμβάνει τόσο τα δεδομένα και τις επιλογές διαμόρφωσης του γραφήματος σε ένα διακομιστή της Google. Περιλαμβάνει ένα κλειστό σύνολο των charts με διάφορες επιλογές. Ουσιαστικά πρόκειται για ένα απλό και εύκολο εργαλείο για την γραφική αναπαράσταση δεδομένων. Το script που χρησιμοποιεί είναι JavaScript. Περνώντας τα δεδομένα που έχουν αποθηκευτεί σε PHP μεταβλητές στα Google Charts, και επιλέγοντας τον τύπο του γραφήματος και κάποιες ακόμα επιλογές αναπαράστασης, δημιουργήθηκαν τα ζητούμενα γραφήματα. Ουσιαστικά, η βάση του κώδικα που χρησιμοποιήθηκε για κάθε διάγραμμα είναι η ίδια, όπως αναφέρθηκε παραπάνω, με αλλαγή του ερωτήματος που γίνεται στη βάση, ανάλογα με τα δεδομένα που αναπαριστούνται και του τύπου του διαγράμματος. Παρακάτω παρατίθενται τα αρχεία που δημιουργήθηκαν και το είδος του γραφήματος που υλοποιείται:

- dbcontemp.php: Εμφανίζει τις τελευταίες 20 μετρήσεις θερμοκρασίας και moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbconhum.php: Εμφανίζει τις τελευταίες 20 μετρήσεις υγρασίας, πραγματικής υγρασίας, moving average και true moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbcontsr.php: Εμφανίζει τις τελευταίες 20 μετρήσεις tsr και moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbconpar.php: Εμφανίζει τις τελευταίες 20 μετρήσεις par και moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbconinvolt.php: Εμφανίζει τις τελευταίες 20 μετρήσεις εσωτερικής τάσης και moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbconintemp.php: Εμφανίζει τις τελευταίες 20 μετρήσεις εσωτερικής θερμοκρασίας και moving average συναρτήσει του χρόνου για τον αισθητήρα του οποίου το id περνάει ο χρήστης μέσω του browser.

- dbconrssi.php: Εμφανίζει τις τελευταίες 20 μετρήσεις rssi συναρτήσει του χρόνου που έχουν πάρει για τον αισθητήρα, του οποίου το id περνάει ο χρήστης μέσω του browser, οι γείτονες του.

- dbcounts.php: Εμφανίζει τα ποσοστά ανά μονάδα-αισθητήρα-αποστολέα των πακέτων που έχει λάβει η μονάδα-βάση.

- dbconrssibase.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Δείκτη Ισχύος Ληφθέντος Σήματος (RSSI) ανά αισθητήρα.

- dbminmax.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Θερμοκρασίας ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax2.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Υγρασίας ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax3.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Πραγματικής Υγρασίας ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax4.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή TSR ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax5.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή PAR ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax6.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Εσωτερικής Τάσης ανά μονάδα-αισθητήρα αποστολέα.

- dbminmax7.php: Εμφανίζει την ελάχιστη, τη μέγιστη και τη μέση τιμή Εσωτερικής Θερμοκρασίας ανά μονάδα-αισθητήρα αποστολέα.

Επίσης για την αποτύπωση των θέσεων των αισθητήρων, χρησιμοποιήθηκε το Google Maps JavaScript Api[72] που δίνει τη δυνατότητα ενσωμάτωσης Google Maps σε ιστότοπο επιλέγοντας το σημείο, ζουμ και τύπο χάρτη που είναι επιθυμητό, μαζί με άλλες επιλογές. Ανακτώντας τις συντεταγμένες των αισθητήρων με PHP με τον ίδιο τρόπο που αναφέρθηκε, καταχωρούνται στο χάρτη με markers. Το εν λόγω αρχείο είναι το maps2.php

5.3.3 Ιστότοπος παρουσίασης αποτελεσμάτων

Η διαχείριση του ιστότοπου http://dpapanikolaou.gr/plevraki/ όπου γίνεται σύντομη παρουσίαση της εφαρμογής και εκτενής παρουσίαση των αποτελεσμάτων έγινε με την εφαρμογή Joomla[71], μια εφαρμογή ανοικτού κώδικα. Με αυτήν έγινε επιλογή της εμφάνισης του ιστοτόπου, του περιεχομένου, δηλαδή του κειμένου, εικόνων και γραφημάτων που δημοσιεύτηκαν και το ακριβές σημείο, μενού και σελίδα που εμφανίζεται το κάθε ένα από αυτά.

Έτσι δημιουργήθηκε μια αρχική σελίδα με γενικές πληροφορίες και τη θέση των αισθητήρων, ένα drop-menu με τις τρεις μονάδες-αισθητήρες και τη μονάδα-βάση και τα αντίστοιχα διαγράμματα σε κάθε σελίδα, όπως έχουν περιγραφεί παραπάνω, μία σελίδα με πληροφορίες για

την πλατφόρμα Tmote Sky και τέλος μία σελίδα με συγκεντρωτικά αποτελέσματα μετρήσεων ανά μονάδα αισθητήρα. Οι σελίδες που περιέχουν τα διαγράμματα ανανεώνονται αυτόματα κάθε 1 λεπτό έτσι ώστε να απεικονίζουν τα νεώτερα δεδομένα.. Εικόνες όπου φαίνονται όλα τα παραπάνω παρατίθενται στην επόμενη ενότητα.

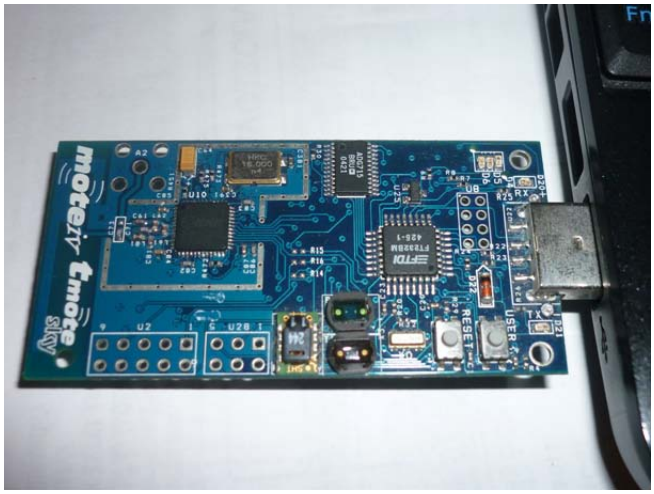
6. ΤΟ ΣΥΣΤΗΜΑ ΟΠΩΣ ΠΡΟΚΥΠΤΕΙ ΣΕ ΛΕΙΤΟΥΡΓΙΑ

Το ασύρματο δίκτυο, που είναι ένα από τα μέλη που απαρτίζουν το σύστημα που τέθηκε σε λειτουργία, αποτελείται από 4 αισθητήρες: 3 μονάδες-κόμβους (Εικόνες 6.1, 6.2, 6.3) και τη μονάδα-βάση (Εικόνα 6.4).

Εικόνα 6.1 Μονάδα-αισθητήρας 1

Εικόνα 6.2 Μονάδα-αισθητήρας 2

Εικόνα 6.3 Μονάδα-αισθητήρας 3



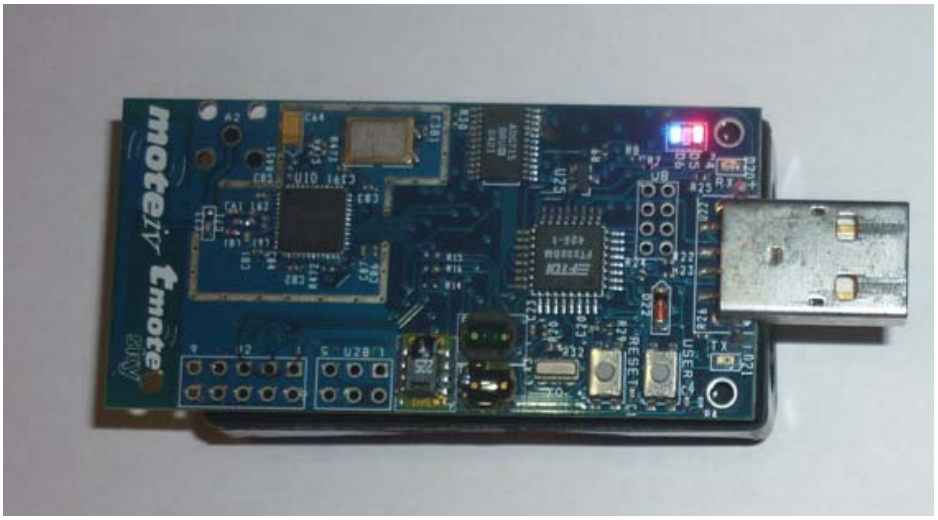
Εικόνα 6.4 Μονάδα-βάση

Όλες οι μονάδες έχουν εγκατεστημένη την εφαρμογή SampleToRadio. Από τη στιγμή που θα τους παρέχουμε την απαραίτητη τάση, θα αρχίσουν να εκτελούν τον κώδικα που έχει περιγραφεί στο προηγούμενο κεφάλαιο. Η μονάδα βάση συνδέεται στη θύρα USB ενός φορητού υπολογιστή, όπως φαίνεται στην εικόνα 6.4, από την οποία και τροφοδοτείται. Οι άλλες μονάδες λειτουργούν με 2 μπαταρίες ΑΑ η καθεμία.

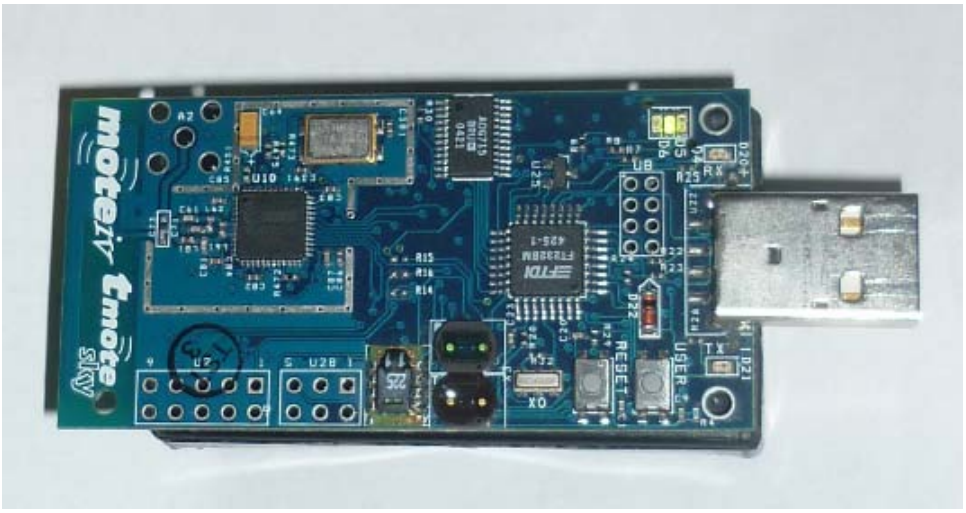
Εικόνα 6.5 Η μονάδα-βάση συνδεδεμένη στο φορητό υπολογιστή.

Από τη στιγμή που η μονάδα-βάση είναι συνδεδεμένη στον υπολογιστή, είναι έτοιμη να δεχτεί πακέτα από τους άλλους αισθητήρες. Κάθε μονάδα-κόμβος παίρνει μετρήσεις κάθε δευτερόλεπτο και όταν συμπληρωθούν 10 (συνεπώς κάθε 10 δευτερόλεπτα) προωθεί το πακέτο

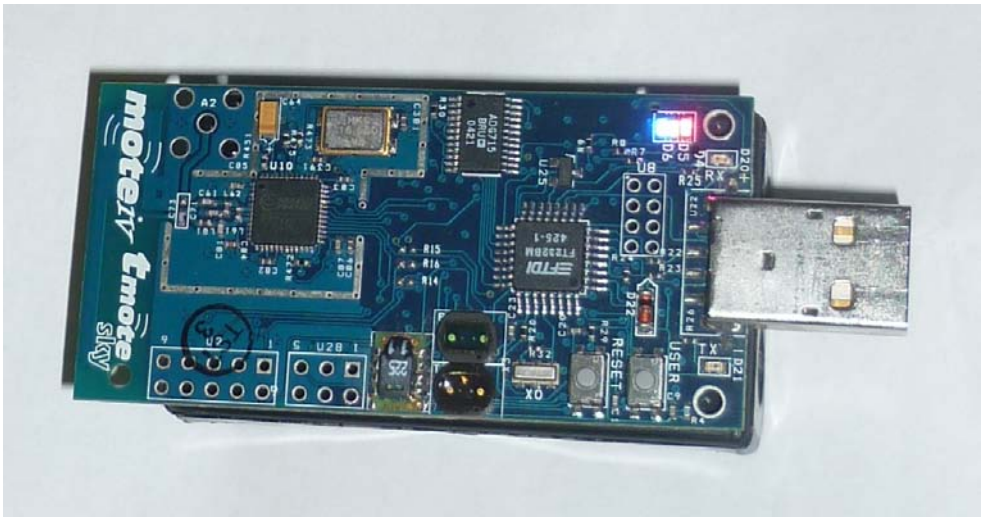
με το μέσο όρο τους προς τη βάση. Όταν συμπληρωθούν οι 10 μετρήσεις, εναλλάσσεται η λειτουργία του led 0, δηλαδή όταν είναι αναμμένο σβήνει κ αντίστροφα (εικόνα 6.6). Το ίδιο γίνεται και με το led 2 όταν έχει γίνει επιτυχώς η ασύρματη αποστολή του πακέτου (εικόνα 6.6) Επίσης, όταν ο κόμβος παραλάβει ένα πακέτο που πρέπει να προωθηθεί στη βάση, γίνεται εναλλαγή της λειτουργίας του led 1 (εικόνα 6.7).



Εικόνα 6.6 Ένδειξη ότι συμπληρώθηκαν 10 μετρήσεις (led 0) και επιτυχής ασυρμάτου αποστολής (led 2).

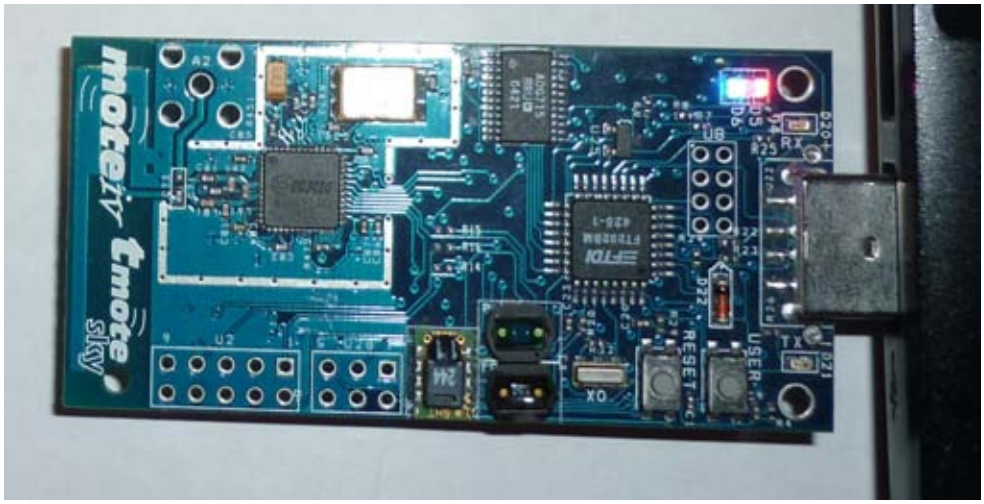


Εικόνα 6.7 Ένδειξη λήψης πακέτου προς προώθηση στη μονάδα-βάση (led 1).

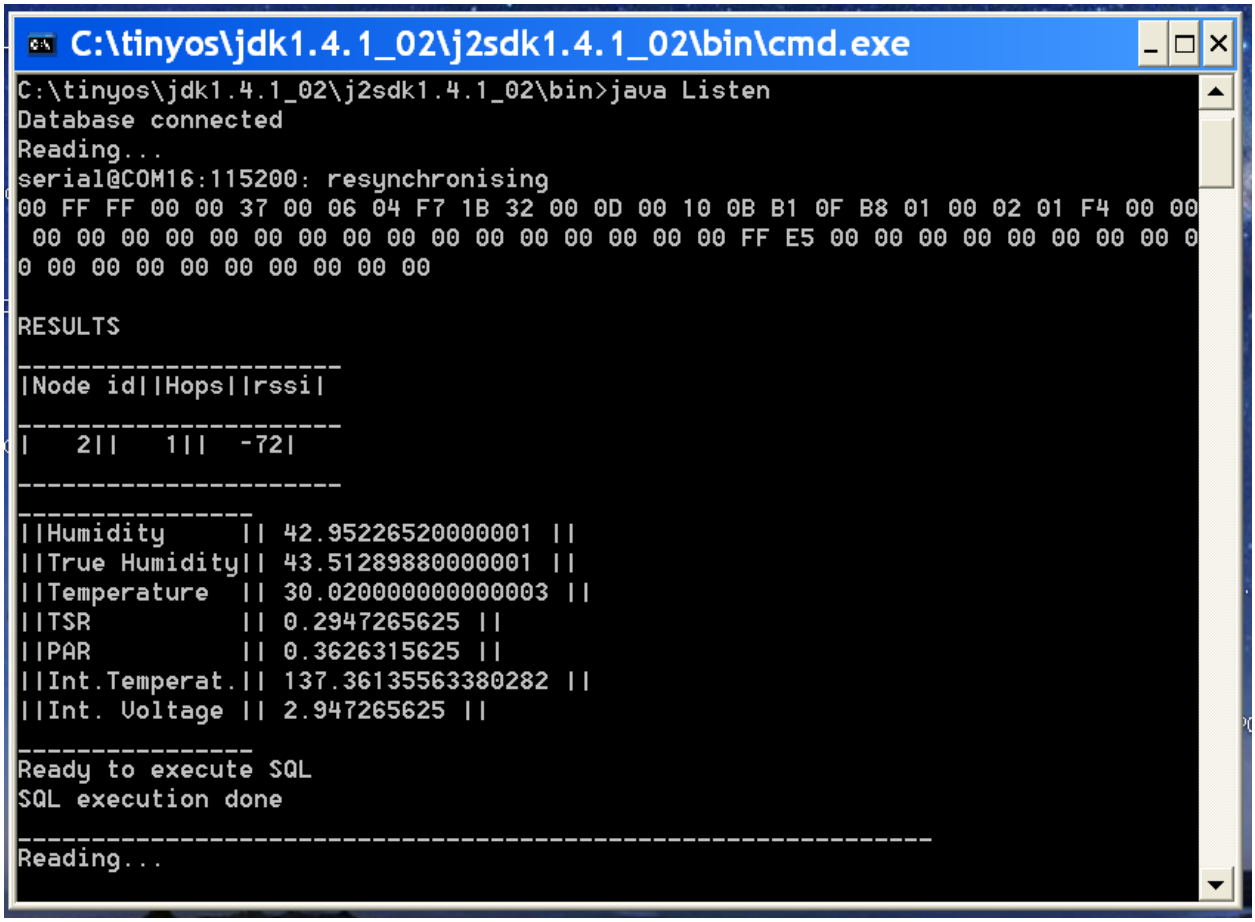


Εικόνα 6.8 Χρήση των led για ειδοποίηση του χρήστη

Όσο για τη μονάδα-βάση, όταν λάβει ένα πακέτο εναλλάσσει τη λειτουργία του led 0 και όταν αυτό το μήνυμα σταλθεί επιτυχώς προς τη σειριακή θύρα, εναλλάσσει το led 2 (εικόνα 6.9).



Εικόνα 6.9 Ένδειξη λήψης πακέτου (led 0) και επιτυχής σειριακής αποστολής (led 1).

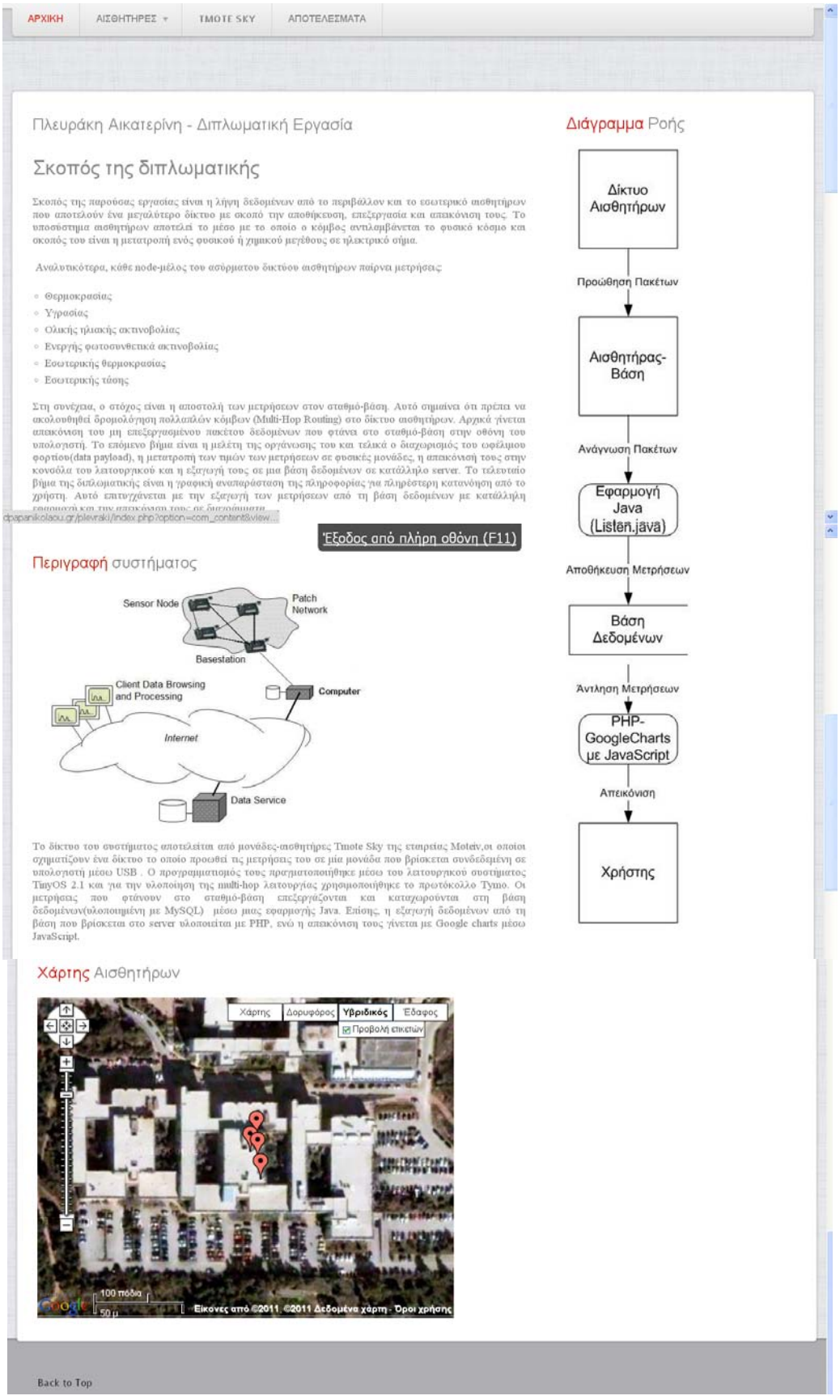


Εικόνα 6.10 Εκτέλεση της Listen.java και ακόλουθο output.

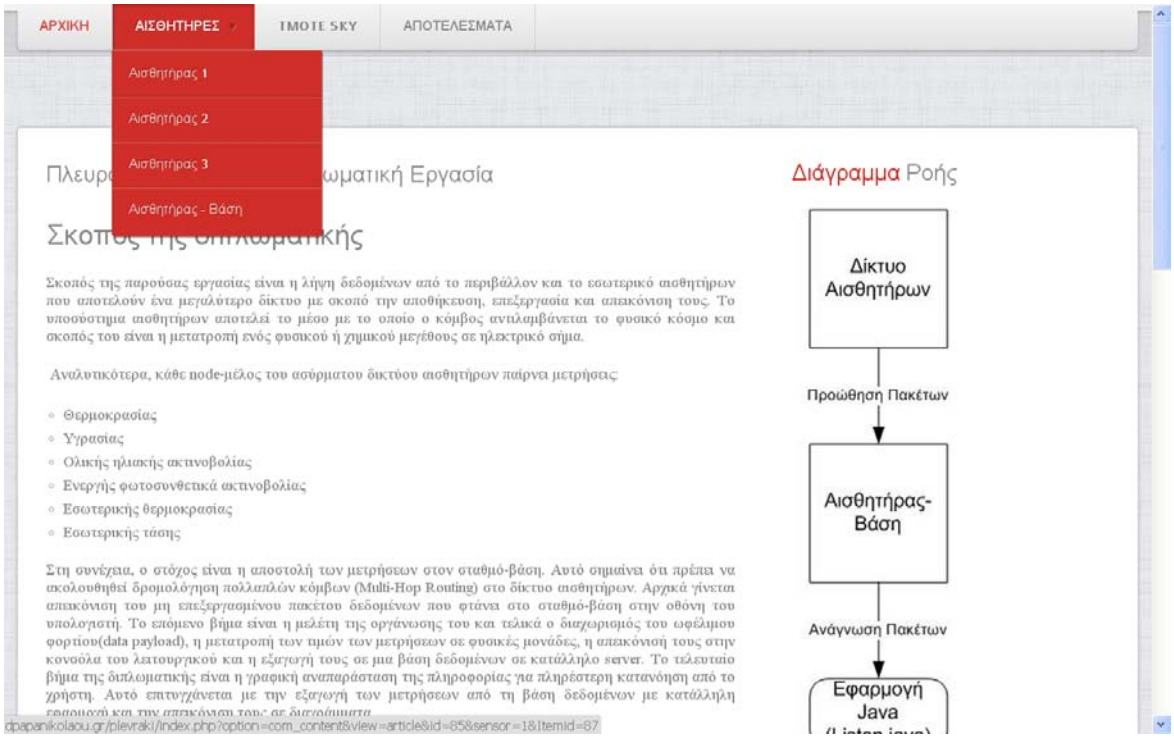
Στην οθόνη εμφανίζονται μηνύματα συστήματος και οι μετρήσεις μαζί με το id της μονάδας αισθητήρα και τα hops.

Για να δούμε τις γραφικές αναπαραστάσεις των μετρήσεων ανατρέχουμε στον ιστότοπο http://dpapanikolaou.gr/plevraki/, οποίος αποτελείται από:

- την αρχική σελίδα (εικόνα 6.11)
- πληροφορίες και αποτελέσματα ανά μονάδα-αισθητήρα (εικόνα 6.12 το drop down menu και 6.13 για τον αισθητήρα 1 και 6.14 για τη μονάδα-βάση)
- πληροφορίες για την πλατφόρμα Tmote Sky(εικόνα 6.15)
- συγκεντρωτικά αποτελέσματα (εικόνα 6.16)



Εικόνα 6.11 http://dpapanikolaou.gr/plevraki/ αρχική σελίδα.



Εικόνα 6.12 http://dpapanikolaou.gr/plevraki/ drop down menu αισθητήρες.

ΑΡΧΙΚΗ | ΑΙΣΘΗΤΗΡΕΣ | TMOTE SKY | ΑΠΟΤΕΛΕΣΜΑΤΑ

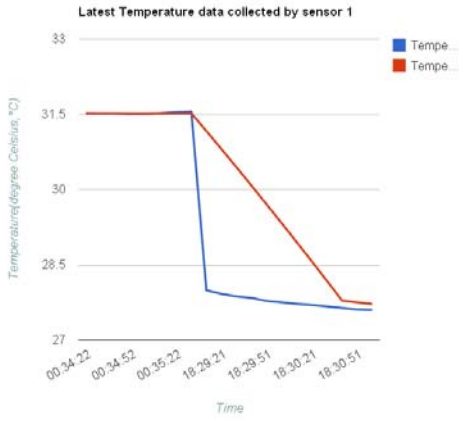
Αισθητήρας 1

Για την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκε η μονάδα - αισθητήρας **Tmote Sky** της εταιρείας moteiv. Η παρούσα μονάδα παίρνει μετρήσεις των ζητούμενων μεγεθών.Όταν συμπληρωθούν 10 μετρήσεις από κάθε μέγεθος, υπολογίζει το μέσο όρο τους και πρωθεί τα αποτελέσματα στη μονάδα-βάση.

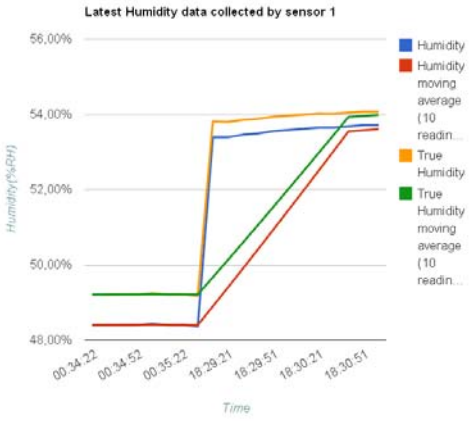
Η τοποθεσία του εν λόγω αισθητήρα δίνεται σε **χάρτη**. (Παρατίθεται και στην αρχική σελίδα.)

Παρακάτω παρουσιάζονται οι τελευταίες 20 μετρήσεις που έχει στείλει η μονάδα σην την ισχύ του σήματος του όπως έχει μετρηθεί από τους γείτονες του τη στιγμή της λήψης των πακέτων που προέρχονται από αυτόν.

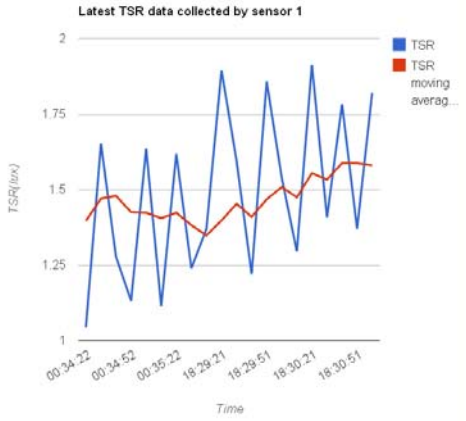
Θερμοκρασία



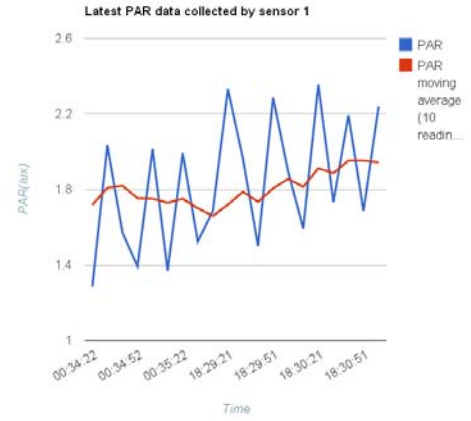
Υγρασία

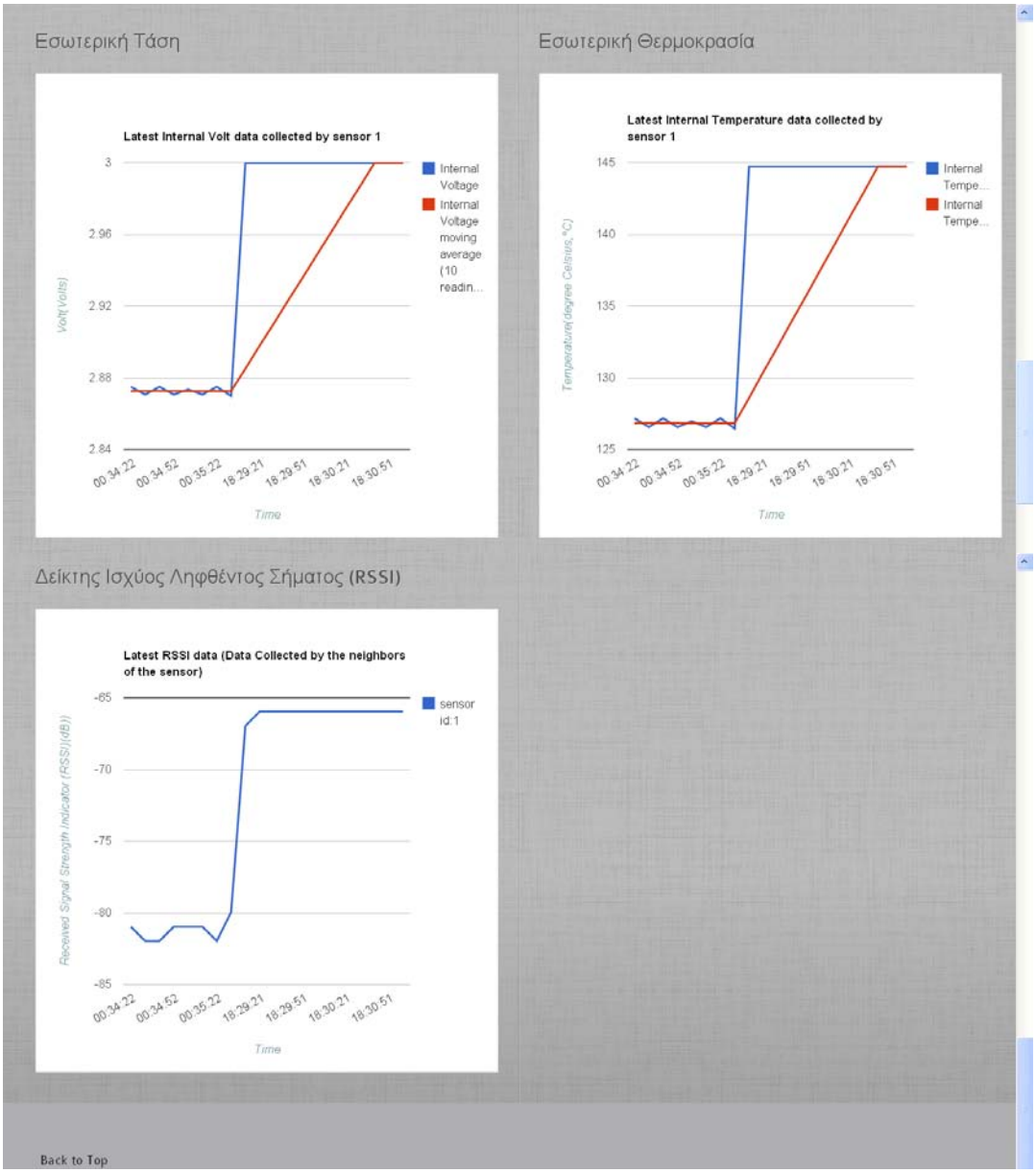


Ολική ηλιακή Ακτινοβολία

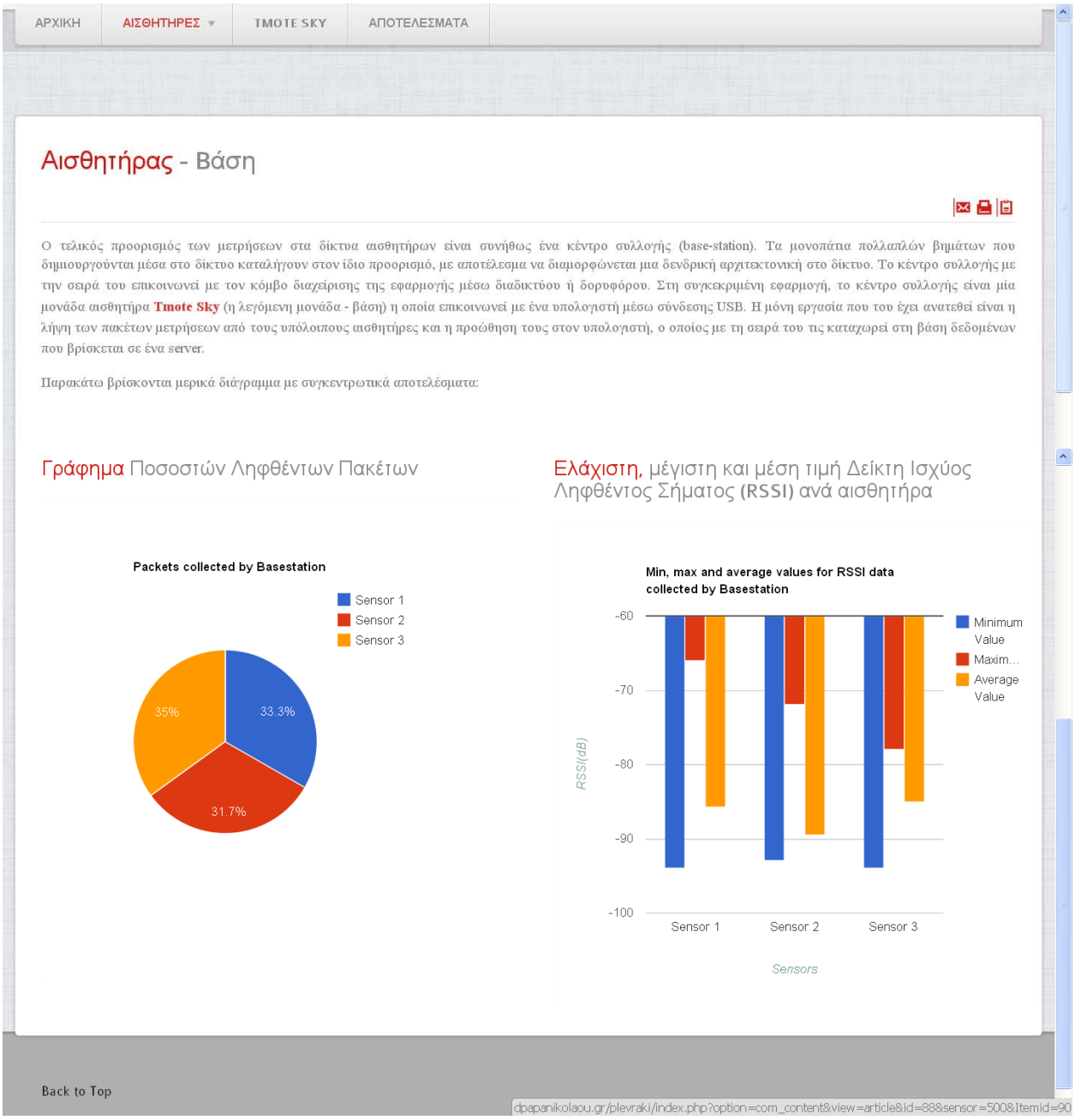


Ενεργή Φωτοσυνθετικά Ακτινοβολία





Εικόνα 6.13 http://dpapanikolaou.gr/plevraki/ αισθητήρας 1



Εικόνα 6.14 http://dpapanikolaou.gr/plevraki/ αισθητήρας βάση.

Πληροφορίες για τη μονάδα αισθητήρα Tmote Sky

Το Tmote Sky είναι μια εξαιρετικά χαμηλής ενεργειακά κατανάλωσης μονάδα ασύρματης επικοινωνίας για χρήση σε δίκτυα αισθητήρων, εφαρμογές παρακολούθησης, και την ταχεία εφαρμογή πρωτοτύπων με σκοπό τόσο την ανεκτικότητα στο θόρυβο όσο και την ευκολία περαιτέρω ανάπτυξης και αξιοποίησης. Το Tmote Sky αξιοποιεί πρότυπα όπως το USB και το IEEE 802.15.4 για να επικοινωνεί άψογα με άλλες συσκευές. Χρησιμοποιώντας βιομηχανικά πρότυπα και ενσωματώνοντας μετρητές υγρασίας, θερμοκρασίας, αισθητήρες φωτός, και παρέχοντας ευέλικτες διασυνδέσεις με περιφερειακά, το Tmote Sky βοηθά στην υλοποίηση ενός μεγάλου φάσματος εφαρμογών δικτύου πλέγματος. Το Tmote Sky είναι ο αντικαταστάτης του επιτυχημένου Telos και περιλαμβάνει αυξημένες επιδόσεις, λειτουργικότητα, και επεκτασιμότητα. Με απόλυτη υποστήριξη από το TinyOS, το Tmote Sky αξιοποιεί τα νέα πρωτόκολλα ασύρματης δικτύωσης και το κίνημα του ανοιχτού λογισμικού. Το Tmote Sky είναι μέρος μιας σειράς πλατφορμών που αναπτύχθηκαν από το Πανεπιστήμιο του Berkeley, California και τα οποία διαθέτουν on-board αισθητήρες για να αυξηθεί η ισχύς, ενώ ταυτόχρονα μειώνεται το κόστος και το μέγεθος του πακέτου.

Βασικά γνωρίσματα

Τα κυριότερα γνωρίσματα και τεχνικά χαρακτηριστικά του Tmote Sky φαίνονται περιληπτικά παρακάτω:

- Ασύρματος πομποδέκτης 250kbpsGHz IEEEChipcon,
- Διαλειτουργικότητα με άλλες συσκευές IEEE 802.15.4,
- Μικροελεγκτής 8MHz Texas Instruments MSP430 (10k RAM, 48k Flash),
- Ολοκληρωμένος ADC, DAC, Supply Voltage Supervisor και ελεγκτής DMA,
- Υλοποιημένη κεραία με εμβέλεια 50m σε εσωτερικούς χώρους/125m σε εξωτερικούς,
- Ενσωματωμένοι αισθητήρες υγρασίας, θερμοκρασίας και φωτός,
- Εξαιρετικά χαμηλή κατανάλωση ρεύματος,
- Γρήγορη αφύπνιση (<6μs),
- Κωδικοποίηση και πιστοποίηση αυθεντικότητας στο στρώμα ζεύξης υλικού,
- Προγραμματισμός και συλλογή δεδομένων μέσω USB,
- Υποστήριξη επέκτασης 16pin και προαιρετικός συνδετήρας SMA για εξωτερική κεραία,
- Υποστήριξη λειτουργικού συστήματος TinyOS, πλέγμα εφαρμογής δικτύωσης και επικοινωνίας,
- Συμμορφώνεται με τους κανονισμούς FCC Part 15,
- Φιλικό προς το περιβάλλον - είναι σύμφωνη με τους κανονισμούς RoHS.

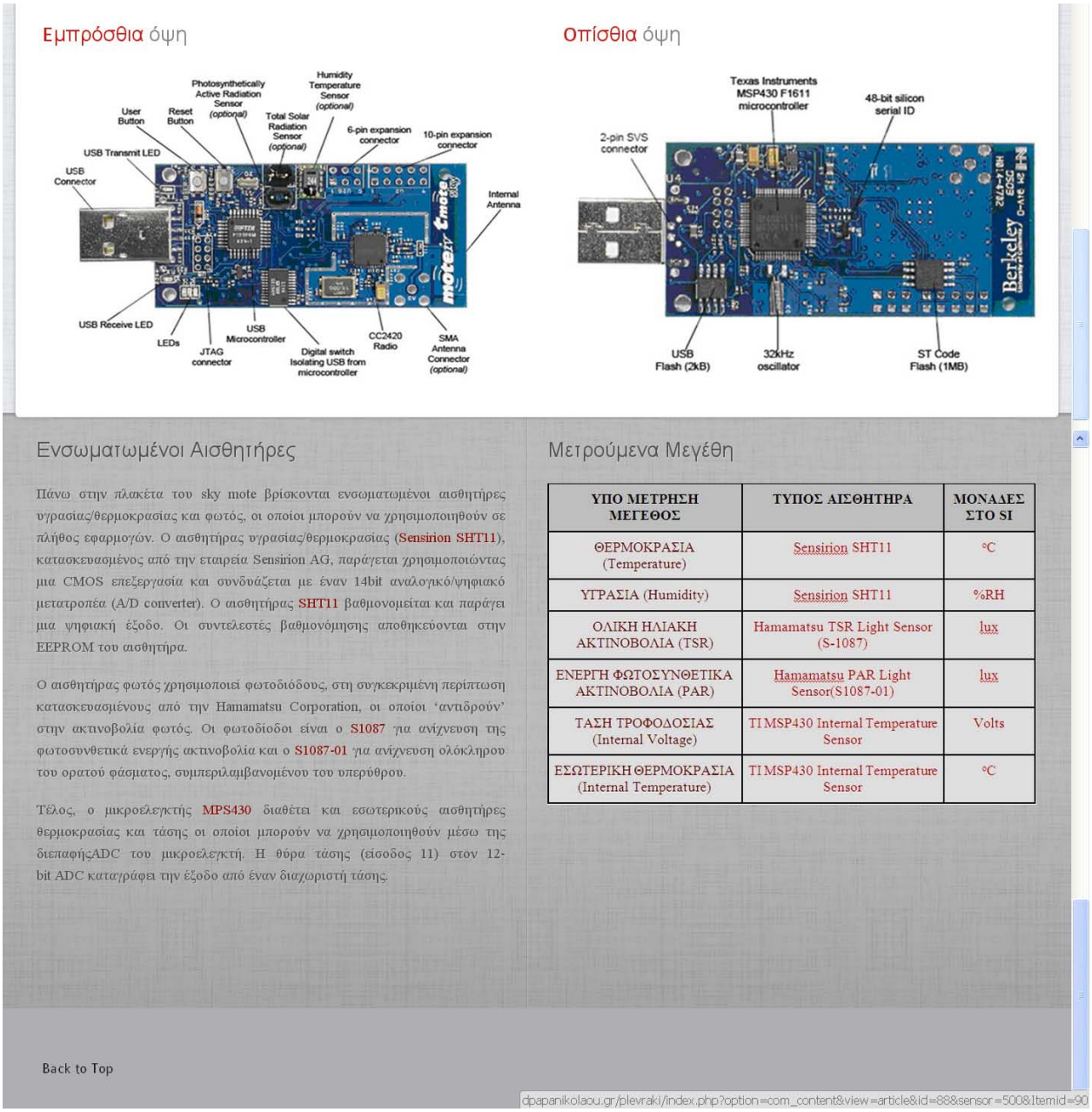
Πίνακας Τεχνικών Χαρακτηριστικών

CPU	
Bus Speed	8 MHz
RAM	10 kB
Program Space	48 kB
External Flash	1024KB
Serial Communications	DIO,SPI,I2C,UART
Current (active w/ Radio on)	19 mA
Current (sleep)	5.1 uA
Startup Time	6 us
Voltage	1.8-3.6 V
Radio	
Frequency	2400-2483 MHz
Data rate	250 kbps
Output Power Startup Time	-25 to 0 dBm 580 us
Antenna Type	Inverted-F or SMA Coax
Humidity Sensor	
Humidity Accuracy	3.5% RH
Temperature Accuracy	0.5 °C
Sampling Rate	90 Hz

Τεχνικά χαρακτηριστικά

Το Tmote Sky χρησιμοποιεί μια ενσωματωμένη κεραία στα 2.4GHz , η οποία είναι μια μικροταινία σε σχήμα ανεστραμμένου F (Planar Inverted Folded Antenna – PIFA) και η οποία βρίσκεται τυπωμένη στην άκρη της πλακέτας, όπως φαίνεται και στην παραπάνω εικόνα (εμπρός όψη). Η κεραία αυτή επιτυγχάνει εμβέλεια 50 μέτρων σε εσωτερικούς χώρους και μπορεί να φτάσει μέχρι και τα 125 μέτρα σε ανοιχτούς. Μια προαιρετική SMA coax σύνδεση μπορεί να χρησιμοποιηθεί αντί της εσωτερικής κεραίας. Η ενσωμάτωση της κεραίας χαμηλώνει το συνολικό κόστος του mote αφού δεν απαιτείται άλλο ακριβό σύστημα εξωτερική κεραίας. Ο προγραμματισμός της μονάδας γίνεται μέσω σύνδεσης με τη θύρα USB ενός υπολογιστή. Για αυτό το λόγο ενσωματώνει πάνω του το κατάλληλο βύσμα USBπου το απαλλάσσει από την ανάγκη χρήσης εξωτερικών καρτών διεπαφών.

Η μονάδα tmote χρησιμοποιεί τον μικροελεγκτή MSP430, ο οποίος έχει τη χαμηλή κατανάλωση ενέργειας σε καταστάσεις αδράνειας και ενεργής λειτουργίας και λειτουργεί με ελάχιστη τάση 1.8 V. Η απαίτηση χαμηλών τιμών τάσης είναι σημαντική για την εξαγωγή όλης της ενέργειας από μια πηγή τάσης. Για παράδειγμα, οι μπαταρίες τύπου AA έχουν τάση αποκοπής στα 0.9V. Αν χρησιμοποιηθούν 2 μπαταρίες σε σειρά, η τάση αποκοπής του συστήματος είναι 1.8V, ακριβώς η ίδια με την ελάχιστη τάση που απαιτεί ο MSP430. Ο MSP430 μεταβαίνει από την κατάσταση αναμονής (standby 1μA) στην κατάσταση λειτουργίας το πολύ σε 6 μs. Επίσης, διαθέτει έναν ελεγκτή DMA (Direct Memory Accesscontroller) προσφέροντας τη δυνατότητα μείωσης του φορτίου στον πυρήνα του μικροελεγκτή και της κατανάλωση ενέργειας, καθώς και αύξησης της απόδοσης.

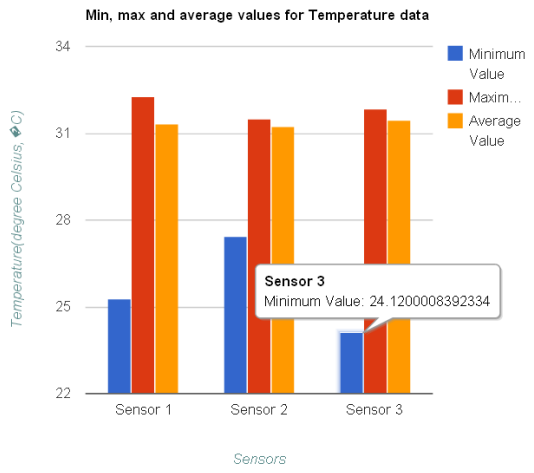


Εικόνα 6.15 http://dpapanikolaou.gr/plevraki/ Tmote Sky.

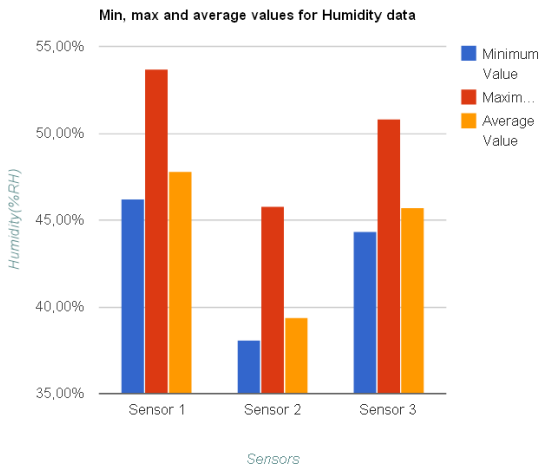
Συγκεντρωτικά Αποτελέσματα

Παρακάτω παρουσιάζονται συγκριτικά διαγράμματα για όλους τους τύπους μερήσεων ανά αισθητήρα. Σε κάθε ένα από αυτά αποτυπώνονται οι ακραίες (μέγιστη και ελάχιστη) και η μέση τιμή κάθε μεγέθους. Οι μεγάλες αποκλίσεις που είναι πιθανό να παρατηρηθούν στις ακραίες τιμές είναι αναμενόμενες εφόσον οι αισθητήρες είναι τοποθετημένοι σε διαφορετικά σημεία και σε κάποιες περιπτώσεις λόγω του περιθωρίου σφάλματος που υπάρχει.

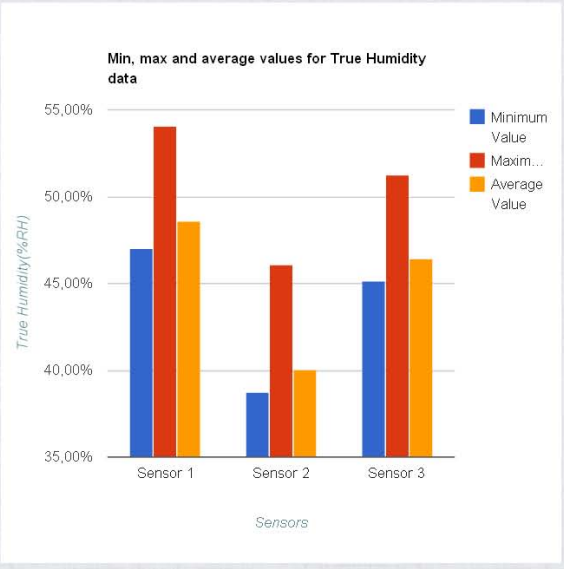
Ελάχιστη, μέγιστη και μέση τιμή Θερμοκρασίας ανά αισθητήρα



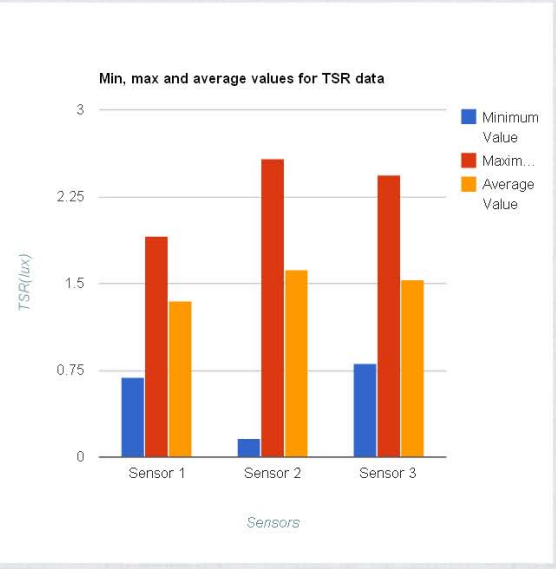
Ελάχιστη, μέγιστη και μέση τιμή Υγρασίας ανά αισθητήρα

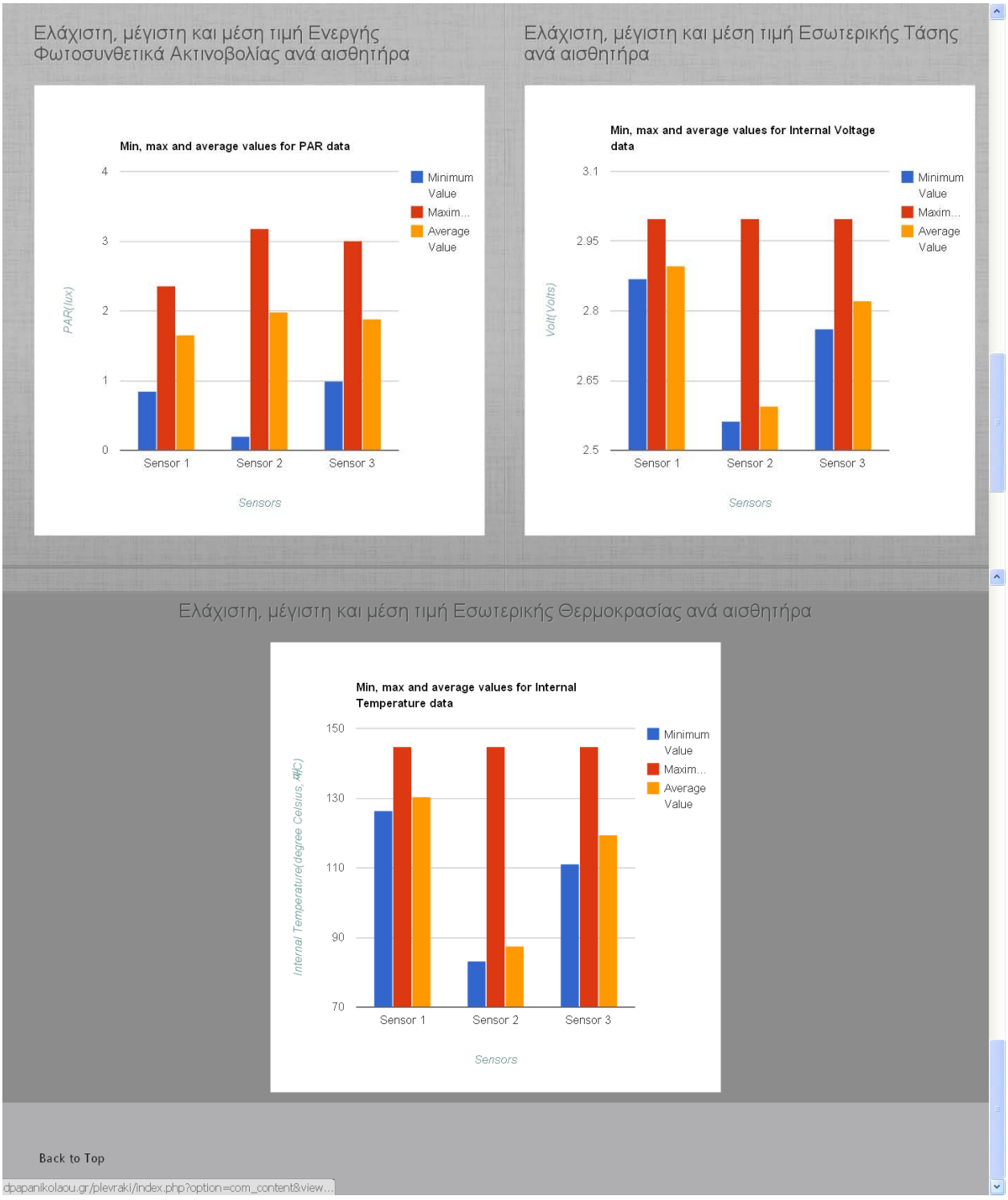


Ελάχιστη, μέγιστη και μέση τιμή Πραγματικής Υγρασίας ανά αισθητήρα



Ελάχιστη, μέγιστη και μέση τιμή Ολικής Ηλιακής Ακτινοβολίας ανά αισθητήρα





Εικόνα 6.16 http://dpapanikolaou.gr/plevraki/ αποτελέσματα.

7. ΕΠΙΛΟΓΟΣ-ΣΥΜΠΕΡΑΣΜΑΤΑ

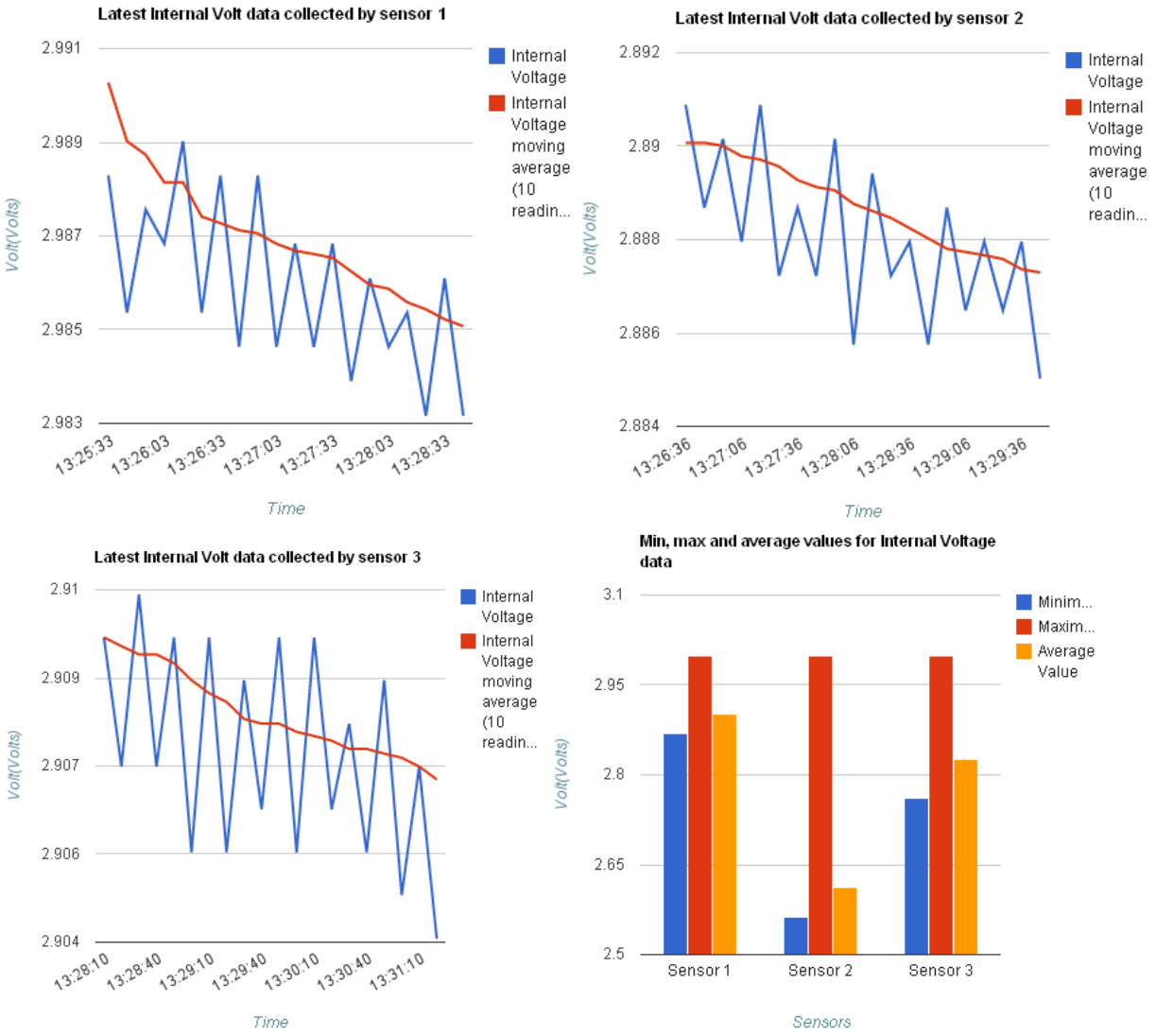
Στην παρούσα εργασία υλοποιήθηκε μια εφαρμογή λήψης, αποθήκευσης και απεικόνισης μετρήσεων με μονάδες-αισθητήρες. Αξίζει να γίνει παράθεση όλων των διαγραμμάτων για όλους τους κόμβους-αισθητήρες (Με κόκκινο σημειώνεται ο moving average κάθε μεγέθους). Σε γενικές γραμμές οι τιμές που παρατηρούνται είναι αναμενόμενες. Κάποιες ακραίες τιμές είναι πολύ πιθανό να οφείλονται σε σφάλμα. Γενικά είναι καλό να γίνεται συχνά ένα calibration του αισθητήρα, κυρίως για τη θερμοκρασία και την υγρασία. Επίσης πρέπει να σημειωθεί ότι οι τιμές της εσωτερικής θερμοκρασίας, αν και υψηλές, είναι αναμενόμενες σύμφωνα με την εικόνα 3.10. Άλλωστε ο αισθητήρας εσωτερικής θερμοκρασίας δεν είναι calibrated και έτσι μπορεί να παρουσιάσει μεγάλα σφάλματα.

Εικόνα 7.1 Γραφήματα θερμοκρασίας για όλες τις μονάδες-αισθητήρες.

Εικόνα 7.2 Γραφήματα υγρασίας και πραγματικής υγρασίας για όλες τις μονάδες-αισθητήρες.

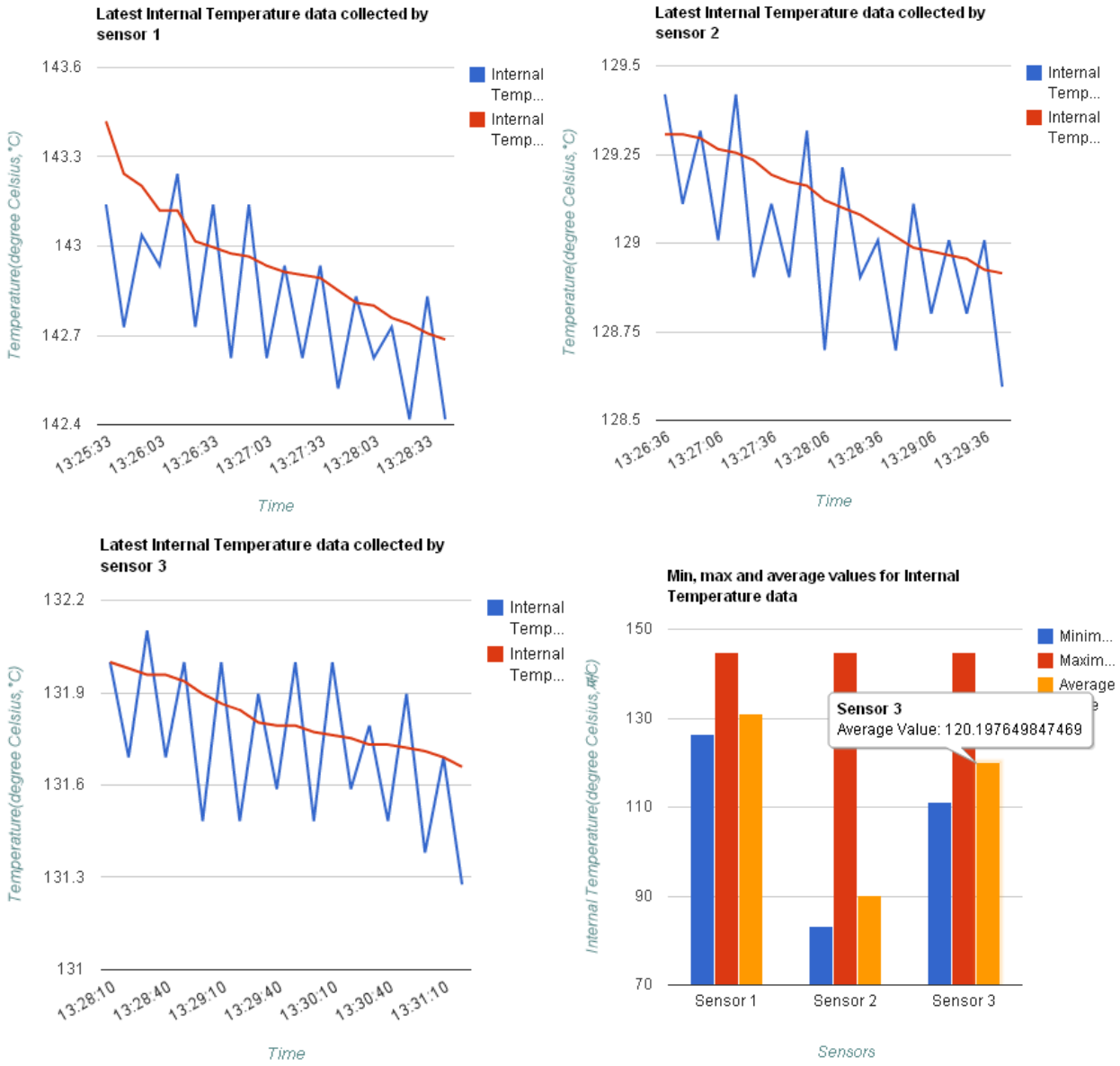
Εικόνα 7.3 ΓραφήματαTSR για όλες τις μονάδες-αισθητήρες.

Εικόνα 7.4 Γραφήματα PAR για όλες τις μονάδες-αισθητήρες.

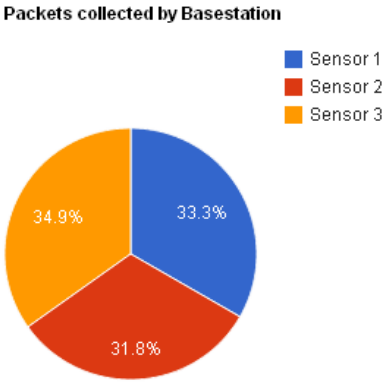


Εικόνα 7.5 Γραφήματα εσωτερικής τάσης για όλες τις μονάδες-αισθητήρες.

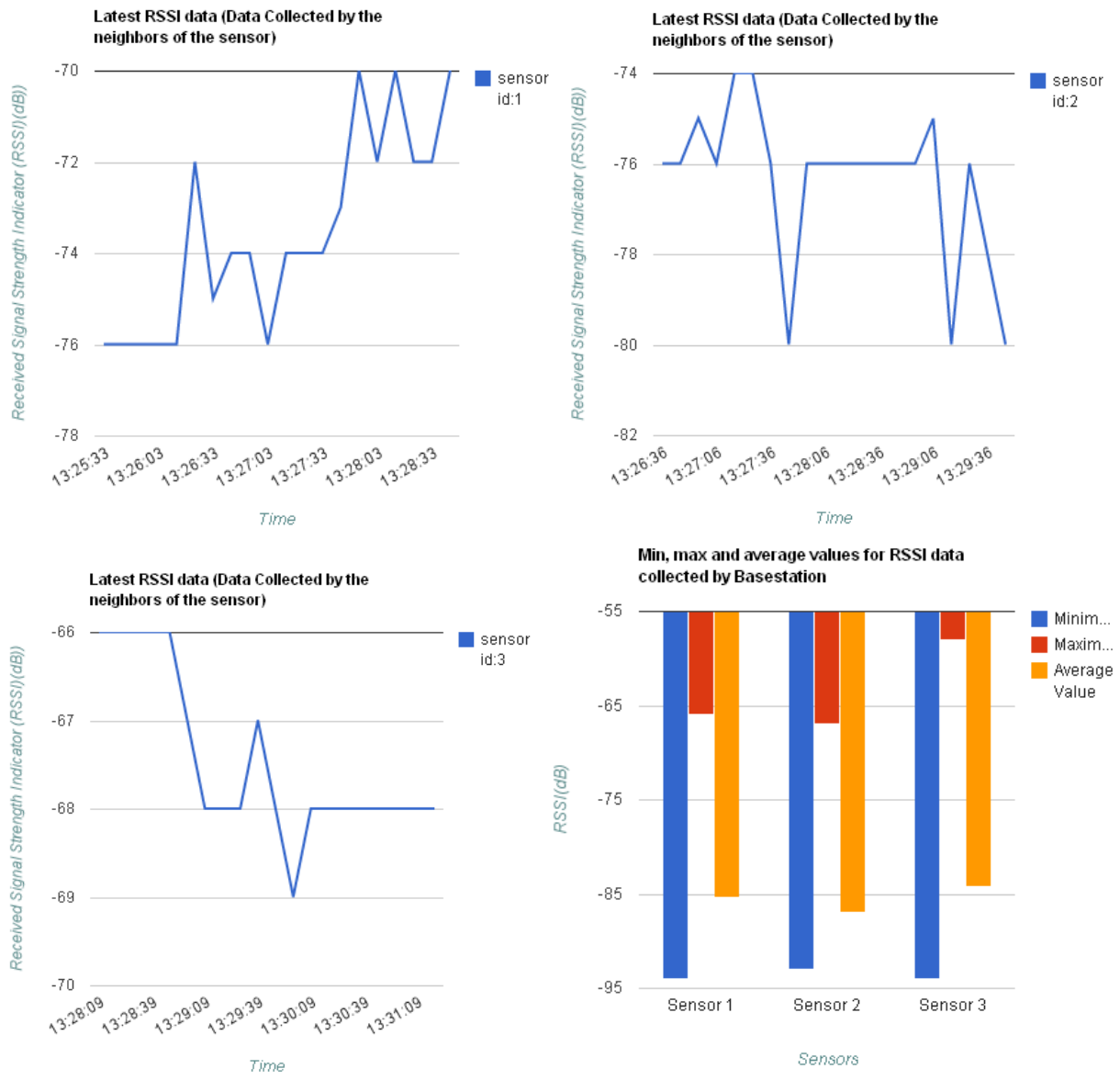
Η κάθε μονάδα αισθητήρας τροφοδοτείται από δύο μπαταρίες AA 1.5V, οι οποίες παρουσιάζουν μη γραμμικότητα στην τάση τροφοδοσίας, με σταδιακή μείωση αυτής, γι'αυτό και παρατηρείται τέτοια καμπύλη στην έξοδο. Παράλληλα συνυπάρχει και το «φαινόμενο της επαναφόρτισης» (recovery effect) [72], κατά το οποίο ένα μέρος της χαμένης τάσης ανακτάται. Αν χρησιμοποιηθεί τροφοδοσία μέσω USB, οι τιμές της εσωτερικής τάσης είναι πολύ πιο σταθερές.



Εικόνα 7.6 Γραφήματα εσωτερικής θερμοκρασίας για όλες τις μονάδες-αισθητήρες.



Εικόνα 7.7 Προέλευση πακέτων που δέχεται η βάση-αισθητήρας



Εικόνα 7.8 Γραφήματα RSSI για όλες τις μονάδες-αισθητήρες.

Η συχνότητα δειγματοληψίας, όπως έχει αναφερθεί, είναι 1 sec. Μικρότερες τιμές έχουν ως αποτέλεσμα λανθασμένα αποτελέσματα(απώλεια πακέτων με βάση το ότι δεν είχε ολοκληρωθεί κάθε φορά η μετάδοση των προηγούμενων), αλλά η τιμή 1 sec, σύμφωνα με τις μετρήσεις (όπως φαίνονται και στις παραπάνω εικόνες) και με δοκιμές με μεγαλύτερες συχνότητες, δίνει ικανοποιητικά αποτελέσματα.

Ο χρήστης μπορεί επίση να αλλάξει το μέγιστο αριθμό βημάτων (στην εφαρμογή είναι ίσος με 1 0,) αν ό μως το αυξήσει κατά πολύ, θα πρέπει να αυξήσει και το μέγεθο ς το υ Payload. Ο αριθμός βημάτων που αναθέρθηκε, σημαίνει ότι η εφαρμογή στην παρούσα μορφή της μπορεί να τρέξει σε δίκτυο με μέχρι και 11 αισθητήρες.

Επίσης αξίζει να τονιστεί ότι ενώ έγινε επιλογή αποστολής του μέσου όρου 10 μετρήσεων ανά πακέτο, ο χρήστης μπορεί να επιλέξει μικρότερο ή μεγαλύτερο αριθμό μετρήσεων, αλλάζοντας την κατάλληλη σταθερα, αν και 10 είναι μία καλή μέση τιμή.

Πέρα από την εφαρμογή που αναπτύχθηκε, ο κάθε κόμβος θα μπορούσε εκτός από το να στέλνει συνεχώς δεδομένα σε κάποιο σταθμό βάσης, να τα επεξεργάζεται πρώτα και να μεταδίδει ασύρματα μόνο συγκεκριμένες τιμές που προκύπτουν από την επεξεργασία αυτή ή να ειδοποιεί για καταστάσεις κινδύνου όταν αυτό απαιτείται. Η πρακτική αυτή θα μπορούσε φυσικά να μειώσει την κατανάλωση ενέργειας, εφόσον το πιο ενεργοβόρο τμήμα ενός τέτοιου συστήματος είναι το τμήμα μετάδοσης δεδομένων. Επιπλέον θα μπορούσε να μειώσει πιθανή συμφόρηση του δικτύου λόγω αυξημένης κίνησης δεδομένων, με ότι αυτό συνεπάγεται για την ποιότητα του. Βέβαια η ενσωμάτωση τοπικής επεξεργασίας σήματος και η παράλληλη μετάδοσή τους αυξάνει και την απασχόληση του επεξεργαστή, ενώ εισάγει και επιπλέον παράγοντες ως προς την συμπεριφορά και τις απαιτήσεις, θέματα που οφείλει να εξετάσει ο σχεδιαστής ενός συστήματος προτού υλοποιήσει την εφαρμογή του, αντισταθμίζοντας τα οφέλη και τα μειονεκτήματα που μπορεί να προκύψουν από μια τέτοια κίνησή για το συγκεκριμένο τύπο της εφαρμογής που καλείται να υλοποιήσει.

Η τάση που επικρατεί αναδεικνύει το σημαντικό ρόλο που μπορούν να διαδραματίσουν τα δίκτυα αισθητήρων στη ζωή μας γενικότερα, στο άμεσο μέλλον. Μάλιστα, υπάρχουν κάποιες βασικές απαιτήσεις τις οποίες τα ασύρματα δίκτυα αισθητήρων μπορούν και καλούνται να ικανοποιήσουν. Αυτές περιλαμβάνουν την απαίτηση χρήσης μικρών, φορητών αισθητήρων, τη δυνατότητα αξιόπιστης και αποδοτικής ασύρματης επικοινωνίας, τη δυνατότητα ταυτόχρονης λήψης δεδομένων σε πολλαπλούς αποδέκτες, την ικανότητα αυτοοργάνωσης σε περιπτώσεις αλλαγής και μετακίνησης κόμβων καθώς και την απαίτηση για ασφάλεια στη διακίνηση των ‘ευαίσθητων’ δεδομένων, με τη χρήση αλγορίθμων κρυπτογραφίας. Το σίγουρο είναι ότι η προοπτική της χρήσης ασύρματων δικτύων αισθητήρων, θα μπορούσε να επιδράσει θετικά σε ένα μεγάλο πλήθος εφαρμογών και να συμβάλει στη βελτίωση της ποιότητας της ζωής του ανθρώπου.

ΠΑΡΑΡΤΗΜΑ Α: Κώδικας Αισθητήρα

SampleToRadio.h

```
 // $Id: SampleToRadio.h,v 1.4 2006/12/12 18:22:52 vlahan Exp $


#ifndef SAMPLETORADIO_H
#define SAMPLETORADIO_H


enum {
     AM_SAMPLETORADIO = 6,
     TIMER_PERIOD_MILLI = 50,//250
     /* Number of readings per message. If you increase this, you may have to
     increase the message_t size. */
     NREADINGS = 10,
     /* Type of readings taken per measurement.
     max=6 (humidity,temperature,TSR,PAR,Internal Temperature, Internal
Voltage)*/
     TYPEREADINGS = 6,
     RETRY_NO = 3,   // Total number of retries before declaring operation
failure
     /* Default sampling period. */
     DEFAULT_INTERVAL = 1024,
     // DEFAULT_INTERVAL = 1024
     AM_OSCILLOSCOPE = 0x93,
     DYMO_HOPLIMIT=10
     };


typedef nx_struct SampleToRadioMsg {
     nx_uint16_t readings[TYPEREADINGS]; /*Pinakas me metriseis diastasis
ari8mos metrisewn ana paketo pou stelnetai epi ari8mos typwn
metrisewn(temp,hum etc)*/
     nx_uint8_t hops;
     nx_uint16_t nodeid[DYMO_HOPLIMIT+1]; //max 10 hops-11 nodes
     nx_uint16_t rssi[DYMO_HOPLIMIT];  //msg sent max 10 times
} SampleToRadioMsg;


#endif
```

SampleToRadioAppC.nc

```
#include <Timer.h>
#include "SampleToRadio.h"

#define CC2420_DEF_RFPOWER 1
#define MAX_TABLE_SIZE 10

configuration SampleToRadioAppC {}

implementation {
          components MainC, LedsC, SampleToRadioC as App,
          new TimerMilliC() as Timer0, new TimerMilliC() as Timer1,
          ActiveMessageC,
          DymoNetworkC,
          SerialActiveMessageC,                    // Serial messaging
          new SerialAMSenderC(AM_SAMPLETORADIO);   // Sends to the serial port

          components new SensirionSht11C() as Sensor,
          new HamamatsuS1087ParC() as PhotoPar,
          new HamamatsuS10871TsrC() as PhotoTsr,
          new Msp430InternalVoltageC() as InternalVoltage,
          new Msp430InternalTemperatureC() as InternalTemperature;

          components CC2420ActiveMessageC;

          components new PoolC(message_t, 10) as UARTMessagePoolP,
          new QueueC(message_t*, 10) as UARTQueueP;

          App.UARTMessagePool -> UARTMessagePoolP;
          App.UARTQueue -> UARTQueueP;
          App.Boot -> MainC;
          App.Leds -> LedsC;
          App.Timer0 -> Timer0;

          App.SplitControl -> DymoNetworkC;
          App.Packet-> DymoNetworkC;
          App.MHPacket -> DymoNetworkC;
```

```
        App.Receive-> DymoNetworkC.Receive[1];
        App.Intercept -> DymoNetworkC.Intercept[1];
        App.MHSend -> DymoNetworkC.MHSend[1];

        App.SerialPacket -> SerialAMSenderC;
        App.SerialControl -> SerialActiveMessageC;
        App.SerialSend -> SerialAMSenderC.AMSend;

        App.ReadT -> Sensor.Temperature;
        App.ReadH -> Sensor.Humidity;
        App.ReadTsr -> PhotoTsr;
        App.ReadPar -> PhotoPar;
        App.ReadInVolt -> InternalVoltage;
        App.ReadInTemp -> InternalTemperature;
        App.CC2420Packet -> CC2420ActiveMessageC.CC2420Packet;

        #ifdef DYMO_MONITORING
             App.DymoMonitor -> DymoNetworkC;
        #endif
}
```

SampletoRadioC.nc

```
#include <Timer.h>
#include "SampleToRadio.h"
#include "routing_table.h"

module SampleToRadioC {
     //Interfaces for initialization:
     uses interface Boot;
     uses interface SplitControl;
     uses interface SplitControl as SerialControl;
     // nterfaces for communication:
     //Interfaces for multihop communication:
     uses interface AMPacket as MHPacket;
     uses interface AMSend as MHSend;
     uses interface Packet;
     uses interface Receive;
     uses interface Intercept;
```

```
    uses interface CC2420Packet;
    //interfaces for Serial communication
    uses interface Packet as SerialPacket;
    uses interface AMSend as SerialSend;
    uses interface Queue<message_t *> as UARTQueue;
    uses interface Pool<message_t> as UARTMessagePool;
    //Interfaces for sensor readings:
    uses interface Read<uint16_t> as ReadT;
    uses interface Read<uint16_t> as ReadH;
    uses interface Read<uint16_t> as ReadTsr;
    uses interface Read<uint16_t> as ReadPar;
    uses interface Read<uint16_t> as ReadInVolt;
    uses interface Read<uint16_t> as ReadInTemp;
    // Miscellany:
    uses interface Leds;
    uses interface Timer<TMilli> as Timer0;

#ifdef DYMO_MONITORING
    uses interface DymoMonitor;
#endif

}

implementation {

    uint8_t reading;        /* 0 to NREADINGS */
    uint8_t retry_no = 0;   // Number of retries so far
    uint8_t uartlen,i,j,len;
    nx_uint16_t sensedata[TYPEREADINGS][NREADINGS]; //data storing
    nx_uint32_t temp;
    nx_uint16_t * payload;
    SampleToRadioMsg* sethops;
    SampleToRadioMsg* sensemsg;
    SampleToRadioMsg* in;
    SampleToRadioMsg* out;
    bool busy = FALSE;
    bool alldone = FALSE;
    bool sendbusy=FALSE, uartbusy=FALSE;
    bool rootme=FALSE;
```

```
     message_t btr;
     message_t sendbuf;
     message_t uartbuf;
     message_t packet;

     //MH and AM addresses are the same
     enum {
          ORIGIN = 1,
          TARGET = 500, //basestation id=500
    };

     // On bootup, initialize radio and serial communications, and our
     // own state variables.
     event void Boot.booted() {
          sensemsg = (SampleToRadioMsg*)(call Packet.getPayload(&btr,
sizeof(SampleToRadioMsg)));
          sensemsg->nodeid[0] = TOS_NODE_ID;
          call SerialControl.start();
          call SplitControl.start();
    }

     //Radio Communication - Multihop
     event void SplitControl.startDone(error_t err) {
          if (err == SUCCESS) {}
          else {
               call SplitControl.start();
          }
    }

     //Serial communication - Root sends generated packets to the UART
     event void SerialControl.startDone(error_t err) {
          if (err == SUCCESS) {
               out = (SampleToRadioMsg*)call SerialSend.getPayload(&uartbuf,
sizeof(SampleToRadioMsg));
               // This is how to set yourself as a root to the collection
layer: (id=500)
               if (sensemsg->nodeid[0] % 500 == 0){
                    rootme=TRUE;
               }
```

```
            else { //If you are not the root, start timer
                 if (call Timer0.isRunning())
                      call Timer0.stop();
                 call Timer0.startPeriodic(DEFAULT_INTERVAL);
            }
        }
        else {
        call SerialControl.start();
        }
    }

  event void SerialControl.stopDone(error_t err) {}

  event void SplitControl.stopDone(error_t e){}

/****************************************/
/*code executed only by basestation*/

  // Use LEDs to report various status issues.
  static void fatal_problem() {
    call Leds.led0On();
    call Leds.led1On();
    call Leds.led2On();
    call Timer0.stop();
  }

    static void report_problem() {
//   call Leds.led0Toggle();
//   call Leds.led1Toggle();
//   call Leds.led2Toggle();
    }
    static void report_sent() {
    }
    static void report_received() {
    }

    task void uartSendTask() { //Send to UART
        if (call SerialSend.send(0xffff, &uartbuf, uartlen) != SUCCESS) {
            report_problem();
```

```
     }
     else {
          uartbusy = TRUE; //UART busy sending
     }
}

task void uartToQueue() {
     if (uartbusy == FALSE) {
     //UART is free. Send message.
               memcpy(out, in, sizeof(SampleToRadioMsg));
               uartlen = sizeof(SampleToRadioMsg);
               post uartSendTask(); //post task:send message to UART
}
     else {
     // The UART is busy; queue up messages and service them when the
     // UART becomes free.
     message_t *newmsg = call UARTMessagePool.get();
     if (newmsg == NULL) {
     // drop the message on the floor if we run out of queue space.
     report_problem();
     return;
     }
     //Serial port busy, so enqueue.
     if (out == NULL) {
          return;
     }
     memcpy(out, in, sizeof(SampleToRadioMsg));

     if (call UARTQueue.enqueue(newmsg) != SUCCESS) {
     // drop the message on the floor and hang if we run out of
     // queue space without running out of queue space first (this
     // should not occur).
     call UARTMessagePool.put(newmsg);
     fatal_problem();
          return;
          }
}

     return;
```

```
    }

    event void SerialSend.sendDone(message_t *msg, error_t error) {
        uartbusy = FALSE; //UART send done
        if (call UARTQueue.empty() == FALSE) {
            // We just finished a UART send, and the uart queue is
            // non-empty.  Let's start a new send.
            message_t *queuemsg = call UARTQueue.dequeue();
            if (queuemsg == NULL) {
                fatal_problem();
                return;
            }
            memcpy(&uartbuf, queuemsg, sizeof(message_t));
            if (call UARTMessagePool.put(queuemsg) != SUCCESS) {
                fatal_problem();
                return;
            }
            post uartSendTask();
        }
    }
/*****************************************************************/
/*****************************************************************/

    task void write() {
        //if you aren't the root, sent the message over radio
        if (rootme==FALSE){
            //try to send the message up to RETRY_NO times
            if ((retry_no <= RETRY_NO)) {
                if (!busy) {
                    if (call MHSend.send(TARGET, &btr,
sizeof(SampleToRadioMsg)) != SUCCESS) {
                        retry_no++;
                        post write();
                    }
                    else
                        call Leds.led0Toggle(); //led on to indicate send
failure
                }
                else {
                    busy=TRUE;
```

```
                    retry_no=0;
                }
            }
        }
    }

    task void sample() {
        if (alldone == TRUE){
            //if the previous sampling is over, increase the counter
            reading++;
            if (reading == NREADINGS) {
                //if 10 readings are completed calculate the average and
send over radio
                call Leds.led0Toggle();
                for (j=0; j<TYPEREADINGS; j++) {
                    temp=0;
                    for (i=0; i<NREADINGS; i++) {
                        temp=temp+sensedata[j][i];  //sum
                    }
                    sensemsg->readings[j]=temp/NREADINGS; //average
                }
                reading=0;
                post write();   // Writes sensor samples collected in this
session to flash
                alldone = FALSE;
            }
        }
    }

  /**
   * Starts sampling all sensors
   */

    event void Timer0.fired() {
        if (rootme==FALSE){  //If you aren't the root sample the sensor
            call ReadH.read();
            call ReadT.read();
            call ReadTsr.read();
            call ReadPar.read();
            call ReadInTemp.read();
```

```
          call ReadInVolt.read();
     }
}

//events to sample the sensor
event void ReadH.readDone(error_t result, uint16_t data) {
     if (result == SUCCESS){
          if (!busy) {
               sensedata[0][reading] = data;
               alldone = FALSE;
          }
     }
     else sensedata[0][reading] = 0xFFFF;
}

event void ReadT.readDone(error_t result, uint16_t data) {
     if (result == SUCCESS){
          if (!busy) {
               sensedata[1][reading] = data;
               alldone = FALSE;
          }
     }
     else sensedata[1][reading] = 0xFFFF;
}

event void ReadTsr.readDone(error_t result, uint16_t data) {
     if (result == SUCCESS){
          if (!busy) {
               sensedata[2][reading] = data;
               alldone = FALSE;
          }
     }
     else sensedata[2][reading] = 0xFFFF;
}

event void ReadPar.readDone(error_t result, uint16_t data) {
     if (result == SUCCESS){
          if (!busy) {
               sensedata[3][reading] = data;
```

```
                alldone = FALSE;
            }
        }
        else sensedata[3][reading] = 0xFFFF;
    }

    event void ReadInTemp .readDone(error_t result, uint16_t data) {
        if (result == SUCCESS){
            if (!busy) {
                sensedata[4][reading] = data;
                alldone = FALSE;
            }
        }
        else sensedata[4][reading] = 0xFFFF;
    }

    event void ReadInVolt.readDone(error_t result, uint16_t data) {
        if (result == SUCCESS){
            if (!busy) {
                sensedata[5][reading] = data;
                alldone = TRUE; //sampling is done
                post sample();  //check if 10 readings are completed and
send results over radio
            }
        }
        else {
            sensedata[5][reading] = 0xFFFF;
            alldone = FALSE;
        }
    }

    //led 2 on when send over radio is done
    event void MHSend.sendDone(message_t* msg, error_t err) {
        call Leds.led2Toggle();
        if (&btr == msg) {
            busy = FALSE;
    }
    }
```

```
    //led 2 on when a packet is received by the root
    event message_t* Receive.receive(message_t* msg, void *payload2, uint8_t
tolen) {
         call Leds.led2Toggle();
         in = (SampleToRadioMsg*)payload2;
         len=tolen;
         if (call MHPacket.address() != TARGET) {
              return msg;
         }
         in->rssi[in->hops] =(nx_uint16_t)(call CC2420Packet.getRssi(msg));
//get rssi value
         (in->hops)++;  //increase number of hops
         in->nodeid[in->hops] = TOS_NODE_ID;   //store node id
         post uartToQueue();  //send packet to UART
         call Leds.led0Toggle(); //led 2 on when a packet is send to UART
         return msg;
    }

    //received a message not meant for me
    //get the payload of the packet, set rssi value and my node id for this
hop
    //increase hop count and forward the message
    event bool Intercept.forward(message_t * msg, void * payload3, uint8_t
len1){
         call Leds.led1Toggle(); //led 1 on
         sethops = (SampleToRadioMsg*)payload3;
         sethops->rssi[sethops->hops] = (nx_uint16_t)(call
CC2420Packet.getRssi(msg));
         (sethops->hops)++;
         sethops->nodeid[sethops->hops] = TOS_NODE_ID;
         return TRUE;
    }

#ifdef DYMO_MONITORING

    event void DymoMonitor.msgReceived(message_t * msg){
         call Leds.led2Toggle();
    }

    event void DymoMonitor.msgSent(message_t * msg){
         call Leds.led2Toggle();
```

```
    }

    event void DymoMonitor.routeDiscovered(uint32_t delay, addr_t target){
        call Leds.led2Toggle();
    }

#endif

}
```

Makefile

```
COMPONENT=SampleToRadioAppC

CFLAGS += -I$(TOSDIR)/lib/net/tymo -I$(TOSDIR)/lib/net/tymo/dymo  -
I$(TOSDIR)/lib/net/tymo/mh -I$(TOSDIR)/chips/cc2420
DOCDIR=~/tymo/web/html-doc/

include $(MAKERULES)
```

CC2424.h

```
/*
 * Copyright (c) 2005-2006 Arch Rock Corporation
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * - Redistributions of source code must retain the above copyright
 *   notice, this list of conditions and the following disclaimer.
 * - Redistributions in binary form must reproduce the above copyright
 *   notice, this list of conditions and the following disclaimer in the
 *   documentation and/or other materials provided with the
 *   distribution.
 * - Neither the name of the Arch Rock Corporation nor the names of
 *   its contributors may be used to endorse or promote products derived
 *   from this software without specific prior written permission.
```

```
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE
 * ARCHED ROCK OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE
 *
 * @author Jonathan Hui <jhui@archrock.com>
 * @author David Moss
 * @version $Revision: 1.19 $ $Date: 2009-09-17 23:36:36 $
 */

#ifndef __CC2420_H__
#define __CC2420_H__

typedef uint8_t cc2420_status_t;

#if defined(TFRAMES_ENABLED) && defined(IEEE154FRAMES_ENABLED)
#error "Both TFRAMES and IEEE154FRAMES enabled!"
#endif

/**
 * CC2420 header definition.
 *
 * An I-frame (interoperability frame) header has an extra network
 * byte specified by 6LowPAN
 *
 * Length = length of the header + payload of the packet, minus the size
 *   of the length byte itself (1).  This is what allows for variable
 *   length packets.
 *
 * FCF = Frame Control Field, defined in the 802.15.4 specs and the
```

```
 *   CC2420 datasheet.
 *
 * DSN = Data Sequence Number, a number incremented for each packet sent
 *   by a particular node.  This is used in acknowledging that packet,
 *   and also filtering out duplicate packets.
 *
 * DestPan = The destination PAN (personal area network) ID, so your
 *   network can sit side by side with another TinyOS network and not
 *   interfere.
 *
 * Dest = The destination address of this packet. 0xFFFF is the broadcast
 *   address.
 *
 * Src = The local node ID that generated the message.
 *
 * Network = The TinyOS network ID, for interoperability with other types
 *   of 802.15.4 networks.
 *
 * Type = TinyOS AM type.  When you create a new AMSenderC(AM_MYMSG),
 *   the AM_MYMSG definition is the type of packet.
 *
 * TOSH_DATA_LENGTH defaults to 28, it represents the maximum size of
 * the payload portion of the packet, and is specified in the
 * tos/types/message.h file.
 *
 * All of these fields will be filled in automatically by the radio stack
 * when you attempt to send a message.
 */
/**
 * CC2420 Security Header
 */
typedef nx_struct security_header_t {
  nx_uint8_t secLevel:3;
  nx_uint8_t keyMode:2;
  nx_uint8_t reserved:3;
  nx_uint32_t frameCounter;
  nx_uint8_t keyID[1]; // One byte for now
} security_header_t;
```

```
typedef nx_struct cc2420_header_t {
  nxle_uint8_t length;
  nxle_uint16_t fcf;
  nxle_uint8_t dsn;
  nxle_uint16_t destpan;
  nxle_uint16_t dest;
  nxle_uint16_t src;
  /** CC2420 802.15.4 header ends here */
#ifdef CC2420_HW_SECURITY
  security_header_t secHdr;
#endif

#ifndef TFRAMES_ENABLED
  /** I-Frame 6LowPAN interoperability byte */
  nxle_uint8_t network;
#endif

  nxle_uint8_t type;
} cc2420_header_t;

/**
 * CC2420 Packet Footer
 */
typedef nx_struct cc2420_footer_t {
} cc2420_footer_t;

/**
 * CC2420 Packet metadata. Contains extra information about the message
 * that will not be transmitted.
 *
 * Note that the first two bytes automatically take in the values of the
 * FCS when the payload is full. Do not modify the first two bytes of
metadata.
 */
typedef nx_struct cc2420_metadata_t {
  nx_uint8_t rssi;
  nx_uint8_t lqi;
  nx_uint8_t tx_power;
  nx_bool crc;
```

```
  nx_bool ack;
  nx_bool timesync;
  nx_uint32_t timestamp;
  nx_uint16_t rxInterval;

  /** Packet Link Metadata */
#ifdef PACKET_LINK
  nx_uint16_t maxRetries;
  nx_uint16_t retryDelay;
#endif
} cc2420_metadata_t;


typedef nx_struct cc2420_packet_t {
  cc2420_header_t packet;
  nx_uint8_t data[];
} cc2420_packet_t;


#ifndef TOSH_DATA_LENGTH
#define TOSH_DATA_LENGTH 126
#endif

#ifndef CC2420_DEF_CHANNEL
#define CC2420_DEF_CHANNEL 26
#endif

#ifndef CC2420_DEF_RFPOWER
#define CC2420_DEF_RFPOWER 31
#endif

/**
 * Ideally, your receive history size should be equal to the number of
 * RF neighbors your node will have
 */
#ifndef RECEIVE_HISTORY_SIZE
#define RECEIVE_HISTORY_SIZE 4
#endif
```

```
/**
 * The 6LowPAN NALP ID for a TinyOS network is 63 (TEP 125).
 */
#ifndef TINYOS_6LOWPAN_NETWORK_ID
#define TINYOS_6LOWPAN_NETWORK_ID 0x3f
#endif

enum {
  // size of the header not including the length byte
  MAC_HEADER_SIZE = sizeof( cc2420_header_t ) - 1,
  // size of the footer (FCS field)
  MAC_FOOTER_SIZE = sizeof( uint16_t ),
  // MDU
  MAC_PACKET_SIZE = MAC_HEADER_SIZE + TOSH_DATA_LENGTH + MAC_FOOTER_SIZE,

  CC2420_SIZE = MAC_HEADER_SIZE + MAC_FOOTER_SIZE,
};

enum cc2420_enums {
  CC2420_TIME_ACK_TURNAROUND = 7, // jiffies
  CC2420_TIME_VREN = 20,          // jiffies
  CC2420_TIME_SYMBOL = 2,         // 2 symbols / jiffy
  CC2420_BACKOFF_PERIOD = ( 20 / CC2420_TIME_SYMBOL ), // symbols
  CC2420_MIN_BACKOFF = ( 20 / CC2420_TIME_SYMBOL ),  // platform specific?
  CC2420_ACK_WAIT_DELAY = 256,    // jiffies
};

enum cc2420_status_enums {
  CC2420_STATUS_RSSI_VALID = 1 << 1,
  CC2420_STATUS_LOCK = 1 << 2,
  CC2420_STATUS_TX_ACTIVE = 1 << 3,
  CC2420_STATUS_ENC_BUSY = 1 << 4,
  CC2420_STATUS_TX_UNDERFLOW = 1 << 5,
  CC2420_STATUS_XOSC16M_STABLE = 1 << 6,
};

enum cc2420_config_reg_enums {
  CC2420_SNOP = 0x00,
  CC2420_SXOSCON = 0x01,
```

```
CC2420_STXCAL = 0x02,
CC2420_SRXON = 0x03,
CC2420_STXON = 0x04,
CC2420_STXONCCA = 0x05,
CC2420_SRFOFF = 0x06,
CC2420_SXOSCOFF = 0x07,
CC2420_SFLUSHRX = 0x08,
CC2420_SFLUSHTX = 0x09,
CC2420_SACK = 0x0a,
CC2420_SACKPEND = 0x0b,
CC2420_SRXDEC = 0x0c,
CC2420_STXENC = 0x0d,
CC2420_SAES = 0x0e,
CC2420_MAIN = 0x10,
CC2420_MDMCTRL0 = 0x11,
CC2420_MDMCTRL1 = 0x12,
CC2420_RSSI = 0x13,
CC2420_SYNCWORD = 0x14,
CC2420_TXCTRL = 0x15,
CC2420_RXCTRL0 = 0x16,
CC2420_RXCTRL1 = 0x17,
CC2420_FSCTRL = 0x18,
CC2420_SECCTRL0 = 0x19,
CC2420_SECCTRL1 = 0x1a,
CC2420_BATTMON = 0x1b,
CC2420_IOCFG0 = 0x1c,
CC2420_IOCFG1 = 0x1d,
CC2420_MANFIDL = 0x1e,
CC2420_MANFIDH = 0x1f,
CC2420_FSMTC = 0x20,
CC2420_MANAND = 0x21,
CC2420_MANOR = 0x22,
CC2420_AGCCTRL = 0x23,
CC2420_AGCTST0 = 0x24,
CC2420_AGCTST1 = 0x25,
CC2420_AGCTST2 = 0x26,
CC2420_FSTST0 = 0x27,
CC2420_FSTST1 = 0x28,
CC2420_FSTST2 = 0x29,
```

```
  CC2420_FSTST3 = 0x2a,
  CC2420_RXBPFTST = 0x2b,
  CC2420_FMSTATE = 0x2c,
  CC2420_ADCTST = 0x2d,
  CC2420_DACTST = 0x2e,
  CC2420_TOPTST = 0x2f,
  CC2420_TXFIFO = 0x3e,
  CC2420_RXFIFO = 0x3f,
};

enum cc2420_ram_addr_enums {
  CC2420_RAM_TXFIFO = 0x000,
  CC2420_RAM_RXFIFO = 0x080,
  CC2420_RAM_KEY0 = 0x100,
  CC2420_RAM_RXNONCE = 0x110,
  CC2420_RAM_SABUF = 0x120,
  CC2420_RAM_KEY1 = 0x130,
  CC2420_RAM_TXNONCE = 0x140,
  CC2420_RAM_CBCSTATE = 0x150,
  CC2420_RAM_IEEEADR = 0x160,
  CC2420_RAM_PANID = 0x168,
  CC2420_RAM_SHORTADR = 0x16a,
};

enum cc2420_nonce_enums {
  CC2420_NONCE_BLOCK_COUNTER = 0,
  CC2420_NONCE_KEY_SEQ_COUNTER = 2,
  CC2420_NONCE_FRAME_COUNTER = 3,
  CC2420_NONCE_SOURCE_ADDRESS = 7,
  CC2420_NONCE_FLAGS = 15,
};

enum cc2420_main_enums {
  CC2420_MAIN_RESETn = 15,
  CC2420_MAIN_ENC_RESETn = 14,
  CC2420_MAIN_DEMOD_RESETn = 13,
  CC2420_MAIN_MOD_RESETn = 12,
  CC2420_MAIN_FS_RESETn = 11,
  CC2420_MAIN_XOSC16M_BYPASS = 0,
```

```
};

enum cc2420_mdmctrl0_enums {
  CC2420_MDMCTRL0_RESERVED_FRAME_MODE = 13,
  CC2420_MDMCTRL0_PAN_COORDINATOR = 12,
  CC2420_MDMCTRL0_ADR_DECODE = 11,
  CC2420_MDMCTRL0_CCA_HYST = 8,
  CC2420_MDMCTRL0_CCA_MOD = 6,
  CC2420_MDMCTRL0_AUTOCRC = 5,
  CC2420_MDMCTRL0_AUTOACK = 4,
  CC2420_MDMCTRL0_PREAMBLE_LENGTH = 0,
};

enum cc2420_mdmctrl1_enums {
  CC2420_MDMCTRL1_CORR_THR = 6,
  CC2420_MDMCTRL1_DEMOD_AVG_MODE = 5,
  CC2420_MDMCTRL1_MODULATION_MODE = 4,
  CC2420_MDMCTRL1_TX_MODE = 2,
  CC2420_MDMCTRL1_RX_MODE = 0,
};

enum cc2420_rssi_enums {
  CC2420_RSSI_CCA_THR = 8,
  CC2420_RSSI_RSSI_VAL = 0,
};

enum cc2420_syncword_enums {
  CC2420_SYNCWORD_SYNCWORD = 0,
};

enum cc2420_txctrl_enums {
  CC2420_TXCTRL_TXMIXBUF_CUR = 14,
  CC2420_TXCTRL_TX_TURNAROUND = 13,
  CC2420_TXCTRL_TXMIX_CAP_ARRAY = 11,
  CC2420_TXCTRL_TXMIX_CURRENT = 9,
  CC2420_TXCTRL_PA_CURRENT = 6,
  CC2420_TXCTRL_RESERVED = 5,
  CC2420_TXCTRL_PA_LEVEL = 0,
};
```

```
enum cc2420_rxctrl0_enums {
  CC2420_RXCTRL0_RXMIXBUF_CUR = 12,
  CC2420_RXCTRL0_HIGH_LNA_GAIN = 10,
  CC2420_RXCTRL0_MED_LNA_GAIN = 8,
  CC2420_RXCTRL0_LOW_LNA_GAIN = 6,
  CC2420_RXCTRL0_HIGH_LNA_CURRENT = 4,
  CC2420_RXCTRL0_MED_LNA_CURRENT = 2,
  CC2420_RXCTRL0_LOW_LNA_CURRENT = 0,
};

enum cc2420_rxctrl1_enums {
  CC2420_RXCTRL1_RXBPF_LOCUR = 13,
  CC2420_RXCTRL1_RXBPF_MIDCUR = 12,
  CC2420_RXCTRL1_LOW_LOWGAIN = 11,
  CC2420_RXCTRL1_MED_LOWGAIN = 10,
  CC2420_RXCTRL1_HIGH_HGM = 9,
  CC2420_RXCTRL1_MED_HGM = 8,
  CC2420_RXCTRL1_LNA_CAP_ARRAY = 6,
  CC2420_RXCTRL1_RXMIX_TAIL = 4,
  CC2420_RXCTRL1_RXMIX_VCM = 2,
  CC2420_RXCTRL1_RXMIX_CURRENT = 0,
};

enum cc2420_rsctrl_enums {
  CC2420_FSCTRL_LOCK_THR = 14,
  CC2420_FSCTRL_CAL_DONE = 13,
  CC2420_FSCTRL_CAL_RUNNING = 12,
  CC2420_FSCTRL_LOCK_LENGTH = 11,
  CC2420_FSCTRL_LOCK_STATUS = 10,
  CC2420_FSCTRL_FREQ = 0,
};

enum cc2420_secctrl0_enums {
  CC2420_SECCTRL0_RXFIFO_PROTECTION = 9,
  CC2420_SECCTRL0_SEC_CBC_HEAD = 8,
  CC2420_SECCTRL0_SEC_SAKEYSEL = 7,
  CC2420_SECCTRL0_SEC_TXKEYSEL = 6,
  CC2420_SECCTRL0_SEC_RXKEYSEL = 5,
```

```
  CC2420_SECCTRL0_SEC_M = 2,
  CC2420_SECCTRL0_SEC_MODE = 0,
};

enum cc2420_secctrl1_enums {
  CC2420_SECCTRL1_SEC_TXL = 8,
  CC2420_SECCTRL1_SEC_RXL = 0,
};

enum cc2420_battmon_enums {
  CC2420_BATTMON_BATT_OK = 6,
  CC2420_BATTMON_BATTMON_EN = 5,
  CC2420_BATTMON_BATTMON_VOLTAGE = 0,
};

enum cc2420_iocfg0_enums {
  CC2420_IOCFG0_BCN_ACCEPT = 11,
  CC2420_IOCFG0_FIFO_POLARITY = 10,
  CC2420_IOCFG0_FIFOP_POLARITY = 9,
  CC2420_IOCFG0_SFD_POLARITY = 8,
  CC2420_IOCFG0_CCA_POLARITY = 7,
  CC2420_IOCFG0_FIFOP_THR = 0,
};

enum cc2420_iocfg1_enums {
  CC2420_IOCFG1_HSSD_SRC = 10,
  CC2420_IOCFG1_SFDMUX = 5,
  CC2420_IOCFG1_CCAMUX = 0,
};

enum cc2420_manfidl_enums {
  CC2420_MANFIDL_PARTNUM = 12,
  CC2420_MANFIDL_MANFID = 0,
};

enum cc2420_manfidh_enums {
  CC2420_MANFIDH_VERSION = 12,
  CC2420_MANFIDH_PARTNUM = 0,
};
```

```
enum cc2420_fsmtc_enums {
  CC2420_FSMTC_TC_RXCHAIN2RX = 13,
  CC2420_FSMTC_TC_SWITCH2TX = 10,
  CC2420_FSMTC_TC_PAON2TX = 6,
  CC2420_FSMTC_TC_TXEND2SWITCH = 3,
  CC2420_FSMTC_TC_TXEND2PAOFF = 0,
};

enum cc2420_sfdmux_enums {
  CC2420_SFDMUX_SFD = 0,
  CC2420_SFDMUX_XOSC16M_STABLE = 24,
};

enum cc2420_security_enums{
  CC2420_NO_SEC = 0,
  CC2420_CBC_MAC = 1,
  CC2420_CTR = 2,
  CC2420_CCM = 3,
  NO_SEC = 0,
  CBC_MAC_4 = 1,
  CBC_MAC_8 = 2,
  CBC_MAC_16 = 3,
  CTR = 4,
  CCM_4 = 5,
  CCM_8 = 6,
  CCM_16 = 7
};
norace uint8_t SECURITYLOCK = 0;

enum
{
  CC2420_INVALID_TIMESTAMP  = 0x80000000L,
};

#endif
```

routing.h

```
/*
 * Copyright (c) 2007 Romain Thouvenin <romain.thouvenin@gmail.com>
 * Published under the terms of the GNU General Public License (GPLv2).
 */

#ifndef _DYMO_ROUTING_H_
#define _DYMO_ROUTING_H_

#include "AM.h"

typedef am_addr_t addr_t;
typedef nx_am_addr_t nx_addr_t;
typedef uint16_t seqnum_t;
typedef nx_uint16_t nx_seqnum_t;

#ifndef MAX_TABLE_SIZE
#define MAX_TABLE_SIZE 5
#endif

#ifndef DYMO_HOPLIMIT
#define DYMO_HOPLIMIT 10
#endif

#ifndef DYMO_ROUTE_AGE_MAX
#define DYMO_ROUTE_AGE_MAX 300000
#endif

#ifndef DYMO_ROUTE_TIMEOUT
#define DYMO_ROUTE_TIMEOUT 10000
#endif

#ifndef DYMO_APPEND_INFO
#define DYMO_APPEND_INFO      0      //1 to append info to forwarded RMs
#endif

#ifndef DYMO_INTER_RREP
#define DYMO_INTER_RREP       1      //1 to allow intermediate RREP
```

```
#endif

#ifndef DYMO_FORCE_INTER_RREP
#define DYMO_FORCE_INTER_RREP 1      //1 to send intermediate RREP even
without target's seqnum in the RREQ
#endif

#ifndef DYMO_LINK_FEEDBACK
#define DYMO_LINK_FEEDBACK    1      //1 to use acks to detect broken links
#endif

enum {
  AM_MULTIHOP = 9,
  AM_DYMO = 8
};

typedef enum {
  DYMO_RREQ = 10,
  DYMO_RREP,
  DYMO_RERR
} dymo_msg_t;

//processing action
typedef enum {
  ACTION_KEEP,   //info is kept in the forwarded message
  //  ACTION_UPDATE, //info is kept, and updated with the provided info
  ACTION_DISCARD, //info is not kept in the forwarded message
  ACTION_DISCARD_MSG //The message won't be forwarded, no need to build a
forwarded message anymore
} proc_action_t;

typedef enum {
  FW_SEND,      //Put the message in the sending queue
  FW_RECEIVE,   //Give the message to the upper layer
  FW_WAIT,      //Retry later
  FW_DISCARD,   //Discard the message
} fw_action_t;



#endif
```

ΠΑΡΑΡΤΗΜΑ Β:Κώδικας Listen.java

```
import java.io.*;
import net.TinyOS.packet.*;
import net.TinyOS.util.*;
import net.TinyOS.message.*;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.lang.Byte;
import java.sql.*;
import java.lang.*;
import java.sql.Connection.*;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Listen {

     static Connection con=null;
     static String hexstr1,hexstr2,hexstr3,hexstr4,hexstr5,temp;
     static int temp2,readings,nodeid;
    public static void main(String args[]) throws IOException {
        try {
              //establish Database Connection
              Class.forName( "com.mysql.jdbc.Driver" ).newInstance();

     con=DriverManager.getConnection("jdbc:mysql://95.211.25.40:3306/dpapanik_
eco?"+"user=dpapanik_kate&password=katerina");
              System.out.println("Database connected");
        }
        catch ( Exception e ){
              System.out.println("Database not connected");
              System.err.println("Got an exception! ");
              System.err.println(e.getMessage());
        }
        String source = null;
        PacketSource reader;
        if (args.length == 2 && args[0].equals("-comm")) {
              source = args[1];
        }
          else if (args.length > 0) {
```

```
            System.err.println("usage: java Listen [-comm PACKETSOURCE]");
            System.err.println("       (default packet source from MOTECOM
environment variable)");
            System.exit(2);
         }
         //connect to mote
        if (source == null) {
            reader = BuildSource.makePacketSource();
        }
        else {
         reader = BuildSource.makePacketSource(source);
        }
         if (reader == null) {
            System.err.println("Invalid packet source (check your MOTECOM
environment variable)");
            System.exit(2);
         }

         try {
            int[] data= new int[6];
            int[] rssi=new int[10];
            int[] hops=new int[11];
            double[] measure=new double[8];
            double[] movavg=new double[7];
            String s[],line1;
            int involt;
            int tempid=0;
            int hopnum=0;
            double tempavg=0;
            reader.open(PrintStreamMessenger.err);
            System.out.println("Reading...");
            for (;;) {
                //variable initialization
                for (int j=0; j<6; j++)
                    data[j]=0;
                for (int j=0; j<8; j++)
                    measure[j]=0;
                for (int j=0; j<10; j++)
                    rssi[j]=0;
                for (int j=0; j<11; j++)
```

```
                    hops[j]=0;
               //reading next packet
               byte[] packet = reader.readPacket();
               Dump.printPacket(System.out, packet);
               System.out.println();
          line1=byteArrayToHexString(packet);
               //****Data retrieval*****
               //Convert data from byte array to hex and then to decimal
               //readings
               for (int j=0; j<6; j++) {
                    hexstr1=line1.substring(16+j*4,20+j*4);
                    temp=hexstr1;
                    data[j]=hex2decimal(hexstr1);
          }
               //hop count
               hexstr4=line1.substring(40,42);
          hopnum=hex2decimal(hexstr4);
               //hops
               for (int j=0; j<hopnum+1; j++) {
               hexstr1=line1.substring(42+j*4,46+j*4);
                    temp=hexstr1;
                    hops[j]=hex2decimal(hexstr1);
          }
               //rssi
               for (int j=0; j<hopnum; j++) {
                    hexstr1=line1.substring(86+j*4,90+j*4);
                    temp=hexstr1;
                    rssi[j]=(hex2decimal(hexstr1)-65536-45);
          }
               //nodeid
               nodeid=hops[0];

               //data conversion from raw values to SI
               //humidity
               if (data[0]!=0)
               measure[0]=-4+0.0405*data[0]+(-
2.8*0.000001)*(data[0]*data[0]);
               //temperature
               if (data[1]!=0)
```

```
                measure[1]=-39.6+0.01*data[1];
                     //Internal Voltage
                     if (data[5]!=0)
                     measure[5]=2*data[5]*1.5/4096;
                     //TSR
                     if (data[2]!=0)
                     measure[2]=0.625*data[3]*measure[5]/100;
                     //PAR
                     if (data[3]!=0)
                     measure[3]=0.769*data[3]*measure[5]/100;
                     //Internal Temperature
                     if (data[4]!=0)
                     measure[4]=((measure[5]/2)-0.986)/0.00355;
                     //True Humidity
                     if (data[0]!=0)
                     measure[6]=measure[0]+(measure[1]-
25)*(0.01+0.00008*data[0]);


                     System.out.println();
                     System.out.println("RESULTS");
                     System.out.println("_________________________");
                     System.out.println("|Node id||Hops||rssi|");
                     System.out.println("_________________________");
                     System.out.println("|   "+nodeid+"||   "+hopnum+"||
"+rssi[hopnum-1]+"|");
                     System.out.println("_________________________");
                     System.out.println("___________________");
                     System.out.println("||Humidity     || "+measure[0]+" ||");
                     System.out.println("||True Humidity|| "+measure[6]+" ||");
                     System.out.println("||Temperature  || "+measure[1]+" ||");
                     System.out.println("||TSR          || "+measure[2]+" ||");
                     System.out.println("||PAR          || "+measure[3]+" ||");
                     System.out.println("||Int.Temperat.|| "+measure[4]+" ||");
                     System.out.println("||Int. Voltage || "+measure[5]+" ||");
                     System.out.println("___________________");

                     //Insert results into database
                     if (con!=null) {
                          System.out.println("Ready to execute SQL");
                          Statement stmt = con.createStatement();
```

```
                        ResultSet rec6=stmt.executeQuery("SELECT max(id) FROM
temp_results;");
                        //retrieve the id of the last entry
                        if (!rec6.next())
                             temp2=0; //if there is no other entry, id=0
                        else
                             temp2=rec6.getInt(1);
                        temp2++;
                        //new reading id=id+1
                        measure[7]=temp2;
                        //insert sensor id into db if it doesn't exist already
                        for (int j=0; j<hopnum+1; j++) {
                             ResultSet rec7=stmt.executeQuery("SELECT sensorid
FROM sensors WHERE sensorid="+hops[j]+";");
                             if (!rec7.next())
                                  int rs11 = stmt.executeUpdate("insert into
sensors values("+(int)hops[j]+", now() ,TRUE, Point(0,0));");
                        }
                        //calculate moving average for each type of readings
                        ResultSet rec10=stmt.executeQuery("SELECT COUNT(*) as
num FROM temp_results WHERE sensorid="+nodeid+";");
                        //if there are less than 9 readings from this sensor,
moving average=0
                        if (rec10.next())||(rec10.getInt(1)>8){
                             ResultSet rec11=stmt.executeQuery("SELECT id from
temp_results WHERE sensorid="+nodeid+" GROUP BY id DESC LIMIT 9;");
                             while (rec11.next())
                                  tempid=rec11.getInt(1);
                             ResultSet rec0=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM temp_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                             if (rec0.next()){
                                  tempavg=rec0.getFloat("sum");
                                  movavg[0]=(tempavg+measure[1])/10;
                             }
                             ResultSet rec1=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM hum_results  WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                             if (rec1.next()){
                                  tempavg=rec1.getFloat("sum");
                                  movavg[1]=(tempavg+measure[0])/10;
                             }
```

```
                              ResultSet rec8=stmt.executeQuery("SELECT
SUM(truevalue) as sum FROM hum_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                              if (rec8.next()){
                                   tempavg=rec8.getFloat("sum");
                                   movavg[6]=(tempavg+measure[6])/10;
                              }
                              ResultSet rec2=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM tsr_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                              if (rec2.next()){
                                   tempavg=rec2.getFloat("sum");
                                   movavg[2]=(tempavg+measure[2])/10;
                              }
                              ResultSet rec3=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM par_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                              if (rec3.next()){
                                   tempavg=rec3.getFloat("sum");
                                   movavg[3]=(tempavg+measure[3])/10;
                              }
                              ResultSet rec5=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM intemp_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                              if (rec5.next()){
                                   tempavg=rec5.getFloat("sum");
                                   movavg[4]=(tempavg+measure[4])/10;
                              }
                              ResultSet rec4=stmt.executeQuery("SELECT
SUM(svalue) as sum FROM involt_results WHERE sensorid="+nodeid+" and
id>="+tempid+";");
                              if (rec4.next()){
                                   tempavg=rec4.getFloat("sum");
                                   movavg[5]=(tempavg+measure[5])/10;
                              }
                         }
                         else {

     movavg[0]=movavg[1]=movavg[2]=movavg[3]=movavg[4]=movavg[5]=movavg[6]=0;
                         }
                         //Insert into db
                         int rs0 = stmt.executeUpdate("insert into temp_results
values("+(int)temp2+","+nodeid+","+measure[1]+",
now(),"+hopnum+","+movavg[0]+");");
```

```
                        int rs1 = stmt.executeUpdate("insert into hum_results
values("+(int)temp2+","+nodeid+","+measure[0]+","+measure[6]+",now(),"+hopnum
+","+movavg[1]+","+movavg[6]+");");
                        int rs2 = stmt.executeUpdate("insert into tsr_results
values("+(int)temp2+","+nodeid+","+measure[2]+",
now(),"+hopnum+","+movavg[2]+");");
                        int rs3 = stmt.executeUpdate("insert into par_results
values("+(int)temp2+","+nodeid+","+measure[3]+",
now(),"+hopnum+","+movavg[3]+");");
                        int rs4 = stmt.executeUpdate("insert into
intemp_results values("+(int)temp2+","+nodeid+","+measure[4]+",
now(),"+hopnum+","+movavg[4]+");");
                        int rs5 = stmt.executeUpdate("insert into
involt_results values("+(int)temp2+","+nodeid+","+measure[5]+",
now(),"+hopnum+","+movavg[5]+");");
                        for (int countj=0; countj<hopnum; countj++) {
                             int rs6 = stmt.executeUpdate("insert into
routing_info
values("+(int)temp2+","+countj+","+hops[countj]+","+hops[(countj+1)]+","+rssi
[countj]+");");
                        }
                        int rs7 = stmt.executeUpdate("update sensors set
activetime=now() where sensorid=500;");
                        System.out.println("SQL execution done");

     System.out.println("_____
_________");
                    }
               }
          }
          catch (Exception e) {
               System.err.println("Error on " + reader.getName() + ": " + e);
          }
     }


     static String byteArrayToHexString(byte in[]) {
          byte ch = 0x00;
          int i = 0;
          if (in == null || in.length <= 0)
               return null;
          String pseudo[] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
"A", "B", "C", "D", "E", "F"};
          StringBuffer out = new StringBuffer(in.length * 2);
          while (i < in.length) {
               ch = (byte) (in[i] & 0xF0); // Strip off high nibble
```

```
                ch = (byte) (ch >>> 4);
                // shift the bits down
                ch = (byte) (ch & 0x0F);
                // must do this is high order bit is on!
                out.append(pseudo[ (int) ch]); // convert the nibble to a
String Character
                ch = (byte) (in[i] & 0x0F); // Strip off low nibble
                out.append(pseudo[ (int) ch]); // convert the nibble to a
String Character
                i++;
           }
           String rslt = new String(out);
           return rslt;
      }

      public static int hex2decimal(String s) {
           String digits = "0123456789ABCDEF";
           s = s.toUpperCase();
           int val = 0;
           for (int i = 0; i < s.length(); i++) {
                char c = s.charAt(i);
                int d = digits.indexOf(c);
                val = 16*val + d;
           }
           return val;
      }
}
```

ΠΑΡΑΡΤΗΜΑ Γ:Κώδικας Δημιουργίας Βάσης

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';


DROP SCHEMA IF EXISTS `dpapanik_eco` ;
CREATE SCHEMA IF NOT EXISTS `dpapanik_eco` ;
SHOW WARNINGS;
USE `dpapanik_eco` ;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`sensors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`sensors` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`sensors` (
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'unique id of each sensor that
exists or has existed in the network' ,
  `activetime` TIMESTAMP NOT NULL ,
  `active` TINYINT(1)  NOT NULL DEFAULT false COMMENT 'value=false if the
sensor hasn\'t sent any packets for some time, value=true otherwise' ,
  `coord` POINT NULL ,
  UNIQUE INDEX (`sensorid` ASC) ,
  PRIMARY KEY (`sensorid`) )
COMMENT = 'ids of the sensors in the network';


SHOW WARNINGS;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`temp_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`temp_results` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`temp_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of
measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the
source of the measurement' ,
```

```
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `tsensor` (`sensorid` ASC) ,
  CONSTRAINT `tsensor`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE CASCADE
    ON UPDATE CASCADE);

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `dpapanik_eco`.`hum_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`hum_results` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`hum_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of
measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the
source of the measurement' ,
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `truevalue` FLOAT NOT NULL ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  `truemovavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_{8BC877EE-57A5-422D-B9F5-D83A7653CC31}` (`sensorid` ASC) ,
  CONSTRAINT `hsensor`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE);
```

```
SHOW WARNINGS;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`tsr_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`tsr_results` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`tsr_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of
measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the
source of the measurement' ,
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `tsensor` (`sensorid` ASC) ,
  CONSTRAINT `tsensor0`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE CASCADE
    ON UPDATE CASCADE);


SHOW WARNINGS;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`par_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`par_results` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`par_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of
measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the
source of the measurement' ,
```

```
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `tsensor` (`sensorid` ASC) ,
  CONSTRAINT `tsensor00`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE CASCADE
    ON UPDATE CASCADE);

SHOW WARNINGS;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`intemp_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`intemp_results` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`intemp_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of
measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the
source of the measurement' ,
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `tsensor` (`sensorid` ASC) ,
  CONSTRAINT `tsensor01`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE CASCADE
    ON UPDATE CASCADE);

SHOW WARNINGS;
```

```
-- -----------------------------------------------------
-- Table `dpapanik_eco`.`involt_results`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`involt_results` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`involt_results` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of measurement' ,
  `sensorid` INT UNSIGNED NOT NULL COMMENT 'id of the sensor which is the source of the measurement' ,
  `svalue` FLOAT NOT NULL COMMENT 'measurent of temperature or humidity' ,
  `ttime` TIMESTAMP NOT NULL ,
  `hops` INT UNSIGNED NOT NULL DEFAULT 0 ,
  `movavg` FLOAT NULL DEFAULT 0 ,
  UNIQUE INDEX (`id` ASC) ,
  PRIMARY KEY (`id`) ,
  INDEX `tsensor` (`sensorid` ASC) ,
  CONSTRAINT `tsensor1`
    FOREIGN KEY (`sensorid` )
    REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
    ON DELETE CASCADE
    ON UPDATE CASCADE);


SHOW WARNINGS;


-- -----------------------------------------------------
-- Table `dpapanik_eco`.`routing_info`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`routing_info` ;


SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`routing_info` (
  `id` BIGINT UNSIGNED NOT NULL COMMENT 'unique id for each packet of measurement' ,
  `hop_no` INT UNSIGNED NOT NULL COMMENT 'number of hops of each packet' ,
  `origin` INT UNSIGNED NOT NULL COMMENT 'sensor id of the first hop of the packet' ,
  `destination` INT UNSIGNED NOT NULL COMMENT 'sensor id of the second hop of the packet' ,
```

```
`rssi` BIGINT NULL ,
PRIMARY KEY (`id`, `hop_no`) ,
INDEX `torigin` (`origin` ASC) ,
INDEX `tdest` (`destination` ASC) ,
INDEX `id2` (`id` ASC) ,
INDEX `id3` (`id` ASC) ,
INDEX `id4` (`id` ASC) ,
INDEX `id5` (`id` ASC) ,
INDEX `id6` (`id` ASC) ,
CONSTRAINT `id`
  FOREIGN KEY (`id` )
  REFERENCES `dpapanik_eco`.`temp_results` (`id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `torigin`
  FOREIGN KEY (`origin` )
  REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `tdest`
  FOREIGN KEY (`destination` )
  REFERENCES `dpapanik_eco`.`sensors` (`sensorid` )
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `id2`
  FOREIGN KEY (`id` )
  REFERENCES `dpapanik_eco`.`hum_results` (`id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `id3`
  FOREIGN KEY (`id` )
  REFERENCES `dpapanik_eco`.`tsr_results` (`id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT `id4`
  FOREIGN KEY (`id` )
  REFERENCES `dpapanik_eco`.`par_results` (`id` )
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
```

```
  CONSTRAINT `id5`
    FOREIGN KEY (`id` )
    REFERENCES `dpapanik_eco`.`intemp_results` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `id6`
    FOREIGN KEY (`id` )
    REFERENCES `dpapanik_eco`.`involt_results` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
COMMENT = 'info on the route of each packet';

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `dpapanik_eco`.`sensor_coord`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `dpapanik_eco`.`sensor_coord` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `dpapanik_eco`.`sensor_coord` (
  `sensorid` INT NOT NULL ,
  `coord` POINT NOT NULL ,
  PRIMARY KEY (`sensorid`) ,
  INDEX `idsens` () ,
  CONSTRAINT `idsens`
    FOREIGN KEY ()
    REFERENCES `dpapanik_eco`.`sensors` ()
    ON DELETE CASCADE
    ON UPDATE CASCADE)
COMMENT = 'Coordinates of the sensors in the network';

SHOW WARNINGS;
USE `dpapanik_eco`;

DELIMITER $$

USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert2` $$
```

```
SHOW WARNINGS$$
USE `dpapanik_eco`$$



CREATE TRIGGER on_insert2 AFTER INSERT ON temp_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE (TIMEDIFF(NEW.ttime,
sensors.activetime)>'02:00:00' and sensors.sensorid<>500);
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE
sensors.sensorid=NEW.sensorid;
END;$$


SHOW WARNINGS$$


DELIMITER ;


DELIMITER $$


USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`SYNTAX_ERROR_1` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









CREATE TRIGGER on_insert1 AFTER INSERT ON hum_results
FOR EACH ROW
BEGIN
    IF (SELECT sensors.sensorid FROM sensors,hum_results WHERE
sensors.sensorid=NEW.sensorid)=null THEN
            INSERT INTO sensors SET sensorid=NEW.sensorid,
activetime=NEW.ttime, active=TRUE$$
```

```
SHOW WARNINGS$$

USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`SYNTAX_ERROR_2` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









    ELSE
        UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE
sensors.sensorid=hum_results.sensorid$$

SHOW WARNINGS$$

USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`SYNTAX_ERROR_3` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









    END IF$$

SHOW WARNINGS$$

USE `dpapanik_eco`$$
```

```
DROP TRIGGER IF EXISTS `dpapanik_eco`.`SYNTAX_ERROR_4` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









END$$

SHOW WARNINGS$$

USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`SYNTAX_ERROR_5` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









//$$

SHOW WARNINGS$$

USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert3` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$
```

```
CREATE TRIGGER on_insert3 AFTER INSERT ON hum_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE (TIMEDIFF(NEW.ttime,
sensors.activetime)>'02:00:00' and sensors.sensorid<>500);
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE
sensors.sensorid=NEW.sensorid;
END;$$


SHOW WARNINGS$$


DELIMITER ;


DELIMITER $$


USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert4` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$
```

```
CREATE TRIGGER on_insert4 AFTER INSERT ON tsr_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE (TIMEDIFF(NEW.ttime,
sensors.activetime)>'02:00:00' and sensors.sensorid<>500);
```

```
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE
sensors.sensorid=NEW.sensorid;
END;$$


SHOW WARNINGS$$


DELIMITER ;


DELIMITER $$


USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert5` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$









CREATE TRIGGER on_insert5 AFTER INSERT ON par_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE (TIMEDIFF(NEW.ttime,
sensors.activetime)>'02:00:00' and sensors.sensorid<>500);
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE
sensors.sensorid=NEW.sensorid;
END;$$


SHOW WARNINGS$$


DELIMITER ;


DELIMITER $$


USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert7` $$
```

```
SHOW WARNINGS$$
USE `dpapanik_eco`$$



CREATE TRIGGER on_insert7 AFTER INSERT ON intemp_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE
(TIMEDIFF(NEW.ttime,sensors.activetime)>'02:00:00' and
sensors.sensorid<>500);
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE


sensors.sensorid=NEW.sensorid;
END;$$


SHOW WARNINGS$$


DELIMITER ;


DELIMITER $$


USE `dpapanik_eco`$$
DROP TRIGGER IF EXISTS `dpapanik_eco`.`on_insert6` $$
SHOW WARNINGS$$
USE `dpapanik_eco`$$



CREATE TRIGGER on_insert6 AFTER INSERT ON involt_results
FOR EACH ROW
BEGIN
    UPDATE sensors SET active=FALSE WHERE
(TIMEDIFF(NEW.ttime,sensors.activetime)>'02:00:00' and
sensors.sensorid<>500);
    UPDATE sensors SET activetime=NEW.ttime, active=TRUE WHERE


sensors.sensorid=NEW.sensorid;
END;$$
```

```
SHOW WARNINGS$$

DELIMITER ;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

ΠΑΡΑΡΤΗΜΑ Δ: Κώδικας Google maps(maps2.php)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<head>
    <title>Χάρτης Αισθητήρων</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAMw4C6G5CCLD
bw_n5P0DdfxRY1uzXaFal13dUfGrdRd6R-i2HJBQfJio4RbsOlQfC-xgKfT_e245gug"
type="text/javascript"></script>
<script type="text/javascript">


     function createMarker(point,html) {
       var marker = new GMarker(point);
       GEvent.addListener(marker, "click", function() {
        marker.openInfoWindowHtml(html);
      });
       return marker;
      }



         function initialize() {
     if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map_canvas"),{ size: new
GSize(500,400) } );
          map.removeMapType(G_HYBRID_MAP);
          map.setCenter(new GLatLng(37.975352, 23.784832),18);
          map.setMapType(G_HYBRID_MAP);
               map.setUIToDefault();
          var mapControl = new GMapTypeControl();
          map.addControl(mapControl);
          map.addControl(new GLargeMapControl());




 <?php
    $con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
    if (!$con)
    {
```

```
          die('Could not connect: ' . mysql_error());
     }
     $db_selected=mysql_select_db('dpapanik_eco', $con);
     if (!$db_selected) {
          die ('Can\'t use sensorsdb : ' . mysql_error());
     }
//echo "Can you see me?";
     $exe1="SELECT sensorid,activetime,active,X(coord),Y(coord) FROM
sensors;";
     $result1 = mysql_query($exe1, $con)or die(mysql_error());
     while(list($sid,$stime,$sactive,$lat,$long) = mysql_fetch_row($result1)){
               if ($sid==500) {
                    $sid2=8;
                    $sid3=90;
                    $sid4="- Basestation";
               }else{
                    $sid2=$sid+4;
                    $sid3=86+$sid;
                    $sid4=$sid;
               }
               echo "\n";
               echo "var point = new GLatLng(".$lat.",".$long.");\n";
               echo "var marker = createMarker(point,'<span
style=\'color:black;\'><a
href=\'index.php?option=com_content&view=article&id=8".$sid2."&sensor=".$sid.
"&Itemid=".$sid3."\' target=\'_self\'>"."Sensor ".$sid4."</a><br />"."Last
active on:".$stime."</span>');\n";
              echo "map.addOverlay(marker);\n";
              echo "\n";
     }
     mysql_close($con);
?>
}
    }


    </script>
</head>
<body onload="initialize()" onunload="GUnload()">
 <div id="map_canvas" style="width: 500px; height: 500px"></div>
</body>
</html>
```

ΠΑΡΑΡΤΗΜΑ Ε: Κώδικας δημιουργίας διαγραμμάτων

dbcontemp.php

```
<meta http-equiv="refresh" content="60">
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,TIME(ttime) as stime,movavg FROM
temp_results WHERE sensorid=".$sensori." ORDER BY id DESC
LIMIT 20");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
     $data[$sensors]['movavg'] = $row['movavg'];
     $sensors--;
     }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
```

```
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'Temperature');
               data.addColumn('number', 'Temperature Moving average(10
readings)');

               data.addRows(20);
               data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
               data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
               data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);

               data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
               data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
               data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);

               data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
               data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
               data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);

               data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
               data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
               data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);

               data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
               data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
               data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);

               data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
               data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
               data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);

               data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
               data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
               data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);

               data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
```

```
data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);

data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);

data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);

data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);

data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);

data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);
```

```
                data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
                data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
                data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);

                data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
                data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
                data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);

                data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
                data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
                data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);

                var options = {title:'Latest Temperature data collected by
sensor <?php echo $sensori; ?>',
                               width:450,
                               height:450,
                               vAxis:{title: 'Temperature(degree Celsius,°C)',
                                    titleTextStyle: {color: '#5c8c8c'},
                                    titlePosition: 'out',
                                    format:'##.##',},
                                    hAxis: {title: 'Time',
                                    titleTextStyle: {color: '#5c8c8c'},
                                    titlePosition: 'out'}
                               };
                var chart = new
google.visualization.LineChart(document.getElementById('chart_div'));
                chart.draw(data, options);
          }
</script>
<div id="chart_div"></div>
```

dbconhum.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;
```

```
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,truevalue,TIME(ttime) as stime,
movavg, truemovavg FROM hum_results WHERE sensorid=".$sensori." ORDER BY id
DESC
LIMIT 20");

  while($row = mysql_fetch_array($result))
    {
    $data[$sensors]=array();
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
    $data[$sensors]['truevalue'] = $row['truevalue'];
    $data[$sensors]['movavg'] = $row['movavg'];
    $data[$sensors]['truemovavg'] = $row['truemovavg'];
     $sensors--;
    }

mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'Humidity');
```

```
            data.addColumn('number', 'Humidity moving average (10
readings)');
            data.addColumn('number', 'True Humidity');
            data.addColumn('number', 'True Humidity moving average (10
readings)');

          data.addRows(20);
          data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
          data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
          data.setValue(0, 3, <?php echo $data[0]['truevalue']; ?>);
          data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);
          data.setValue(0, 4, <?php echo $data[0]['truemovavg']; ?>);

          data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
          data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
          data.setValue(1, 3, <?php echo $data[1]['truevalue']; ?>);
          data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);
          data.setValue(1, 4, <?php echo $data[1]['truemovavg']; ?>);

          data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
          data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
          data.setValue(2, 3, <?php echo $data[2]['truevalue']; ?>);
          data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);
          data.setValue(2, 4, <?php echo $data[2]['truemovavg']; ?>);

          data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
          data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
          data.setValue(3, 3, <?php echo $data[3]['truevalue']; ?>);
          data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);
          data.setValue(3, 4, <?php echo $data[3]['truemovavg']; ?>);

          data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
          data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
          data.setValue(4, 3, <?php echo $data[4]['truevalue']; ?>);
          data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);
          data.setValue(4, 4, <?php echo $data[4]['truemovavg']; ?>);

          data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
          data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
          data.setValue(5, 3, <?php echo $data[5]['truevalue']; ?>);
```

```
data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);
data.setValue(5, 4, <?php echo $data[5]['truemovavg']; ?>);

data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
data.setValue(6, 3, <?php echo $data[6]['truevalue']; ?>);
data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);
data.setValue(6, 4, <?php echo $data[6]['truemovavg']; ?>);

data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
data.setValue(7, 3, <?php echo $data[7]['truevalue']; ?>);
data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);
data.setValue(7, 4, <?php echo $data[7]['truemovavg']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 3, <?php echo $data[8]['truevalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);
data.setValue(8, 4, <?php echo $data[8]['truemovavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 3, <?php echo $data[9]['truevalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);
data.setValue(9, 4, <?php echo $data[9]['truemovavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 3, <?php echo $data[10]['truevalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);
data.setValue(10, 4, <?php echo $data[10]['truemovavg']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
data.setValue(11, 3, <?php echo $data[11]['truevalue']; ?>);
data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);
data.setValue(11, 4, <?php echo $data[11]['truemovavg']; ?>);
```

```
data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
data.setValue(12, 3, <?php echo $data[12]['truevalue']; ?>);
data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);
data.setValue(12, 4, <?php echo $data[12]['truemovavg']; ?>);

data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
data.setValue(13, 3, <?php echo $data[13]['truevalue']; ?>);
data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);
data.setValue(13, 4, <?php echo $data[13]['truemovavg']; ?>);

data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
data.setValue(14, 3, <?php echo $data[14]['truevalue']; ?>);
data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);
data.setValue(14, 4, <?php echo $data[14]['truemovavg']; ?>);

data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
data.setValue(15, 3, <?php echo $data[15]['truevalue']; ?>);
data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);
data.setValue(15, 4, <?php echo $data[15]['truemovavg']; ?>);

data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
data.setValue(16, 3, <?php echo $data[16]['truevalue']; ?>);
data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);
data.setValue(16, 4, <?php echo $data[16]['truemovavg']; ?>);

data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
data.setValue(17, 3, <?php echo $data[17]['truevalue']; ?>);
data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);
data.setValue(17, 4, <?php echo $data[17]['truemovavg']; ?>);

data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
data.setValue(18, 3, <?php echo $data[18]['truevalue']; ?>);
```

```
          data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);
          data.setValue(18, 4, <?php echo $data[18]['truemovavg']; ?>);

          data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
          data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
          data.setValue(19, 3, <?php echo $data[19]['truevalue']; ?>);
          data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);
          data.setValue(19, 4, <?php echo $data[19]['truemovavg']; ?>);


               var options = {'title':'Latest Humidity data collected by
sensor <?php echo $sensori; ?>',
                         'width':450,
                         'height':450,
                         vAxis:{title: 'Humidity(%RH)',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out',
                              format:'##,###%'},
                              hAxis: {title: 'Time',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out'}
                         };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div2'));
          chart.draw(data, options);
          }
</script>
<div id="chart_div2"></div>
```

dbcontsr.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
```

```
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,TIME(ttime) as stime, movavg FROM
tsr_results WHERE sensorid=".$sensori." ORDER BY id DESC
LIMIT 20");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
     $data[$sensors]['movavg'] = $row['movavg'];
     $sensors--;
     }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'TSR');
               data.addColumn('number', 'TSR moving average(10 readings)');

               data.addRows(20);
               data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
               data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
               data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);

               data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
               data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
```

```
data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);

data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);

data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);

data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);

data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);

data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);

data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
```

```
            data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
            data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);

            data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
            data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
            data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);

            data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
            data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
            data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);

            data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
            data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
            data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);

            data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
            data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
            data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);

            data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
            data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
            data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);

            data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
            data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
            data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);

            data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
            data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
            data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);

            data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
            data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
            data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);

            var options = {'title':'Latest TSR data collected by sensor
<?php echo $sensori; ?>',
                           'width':450,
                           'height':450,
```

```
                        vAxis:{title: 'TSR(lux)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'#.###'},
                             hAxis: {title: 'Time',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div3'));
          chart.draw(data, options);
          }
</script>
<div id="chart_div3"></div>
```

dbconpar.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,TIME(ttime) as stime, movavg FROM
par_results WHERE sensorid=".$sensori." ORDER BY id DESC
LIMIT 20");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
```

```
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
     $data[$sensors]['movavg'] = $row['movavg'];
     $sensors--;
     }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'PAR');
               data.addColumn('number', 'PAR moving average (10 readings)');
               //var maxt=<?php echo $maxtimenum;?>;

               data.addRows(20);
               data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
               data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
               data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);

               data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
               data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
               data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);

               data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
               data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
               data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);

               data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
               data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
               data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);

               data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
               data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
```

```
data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);

data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);

data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);

data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);

data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);

data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);

data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
```

```
            data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
            data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);

            data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
            data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
            data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);

            data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
            data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
            data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);

            data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
            data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
            data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);

            data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
            data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
            data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);

            data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
            data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
            data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);

            var options = {'title':'Latest PAR data collected by sensor
<?php echo $sensori; ?>',
                        'width':450,
                        'height':450,
                        vAxis:{title: 'PAR(lux)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'#.###'},
                             hAxis: {title: 'Time',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div4'));
          chart.draw(data, options);
          }
</script>
```

```
<div id="chart_div4"></div>
```

dbconinvolt.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,TIME(ttime) as stime, movavg FROM
involt_results WHERE sensorid=".$sensori." ORDER BY id DESC
LIMIT 20");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
     $data[$sensors]['movavg'] = $row['movavg'];
     $sensors--;
     }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
```

```
        {
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Time');
            data.addColumn('number', 'Internal Voltage');
            data.addColumn('number', 'Internal Voltage moving average (10
readings)');

            data.addRows(20);
            data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
            data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
            data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);

            data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
            data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
            data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);

            data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
            data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
            data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);

            data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
            data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
            data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);

            data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
            data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
            data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);

            data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
            data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
            data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);

            data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
            data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
            data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);

            data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
            data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
            data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);
```

```
data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);

data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);

data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);

data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);

data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);

data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);

data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
```

```
            data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);

            data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
            data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
            data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);

            data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
            data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
            data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);

            var options = {'title':'Latest Internal Volt data collected by
sensor <?php echo $sensori; ?>',
                        'width':450,
                        'height':450,
                        vAxis:{title: 'Volt(Volts)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'#.###'},
                             hAxis: {title: 'Time',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
         var chart = new
google.visualization.LineChart(document.getElementById('chart_div6'));
         chart.draw(data, options);
         }
</script>
<div id="chart_div6"></div>
```

dbconintemp.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
```

```
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
  $result = mysql_query("SELECT svalue,TIME(ttime) as stime, movavg FROM
intemp_results WHERE sensorid=".$sensori." ORDER BY id DESC
LIMIT 20");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
    $data[$sensors]['svalue'] = $row['svalue'];
    $data[$sensors]['stime'] = $row['stime'];
     $data[$sensors]['movavg'] = $row['movavg'];
     $sensors--;
     }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'Internal Temperature');
               data.addColumn('number', 'Internal Temperature moving average
(10 readings)');

               data.addRows(20);
               data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
               data.setValue(0, 1, <?php echo $data[0]['svalue']; ?>);
               data.setValue(0, 2, <?php echo $data[0]['movavg']; ?>);

               data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
```

```
data.setValue(1, 1, <?php echo $data[1]['svalue']; ?>);
data.setValue(1, 2, <?php echo $data[1]['movavg']; ?>);

data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
data.setValue(2, 1, <?php echo $data[2]['svalue']; ?>);
data.setValue(2, 2, <?php echo $data[2]['movavg']; ?>);

data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
data.setValue(3, 1, <?php echo $data[3]['svalue']; ?>);
data.setValue(3, 2, <?php echo $data[3]['movavg']; ?>);

data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
data.setValue(4, 1, <?php echo $data[4]['svalue']; ?>);
data.setValue(4, 2, <?php echo $data[4]['movavg']; ?>);

data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
data.setValue(5, 1, <?php echo $data[5]['svalue']; ?>);
data.setValue(5, 2, <?php echo $data[5]['movavg']; ?>);

data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
data.setValue(6, 1, <?php echo $data[6]['svalue']; ?>);
data.setValue(6, 2, <?php echo $data[6]['movavg']; ?>);

data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
data.setValue(7, 1, <?php echo $data[7]['svalue']; ?>);
data.setValue(7, 2, <?php echo $data[7]['movavg']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['svalue']; ?>);
data.setValue(8, 2, <?php echo $data[8]['movavg']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['svalue']; ?>);
data.setValue(9, 2, <?php echo $data[9]['movavg']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['svalue']; ?>);
data.setValue(10, 2, <?php echo $data[10]['movavg']; ?>);
```

```
            data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
            data.setValue(11, 1, <?php echo $data[11]['svalue']; ?>);
            data.setValue(11, 2, <?php echo $data[11]['movavg']; ?>);

            data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
            data.setValue(12, 1, <?php echo $data[12]['svalue']; ?>);
            data.setValue(12, 2, <?php echo $data[12]['movavg']; ?>);

            data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
            data.setValue(13, 1, <?php echo $data[13]['svalue']; ?>);
            data.setValue(13, 2, <?php echo $data[13]['movavg']; ?>);

            data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
            data.setValue(14, 1, <?php echo $data[14]['svalue']; ?>);
            data.setValue(14, 2, <?php echo $data[14]['movavg']; ?>);

            data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
            data.setValue(15, 1, <?php echo $data[15]['svalue']; ?>);
            data.setValue(15, 2, <?php echo $data[15]['movavg']; ?>);

            data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
            data.setValue(16, 1, <?php echo $data[16]['svalue']; ?>);
            data.setValue(16, 2, <?php echo $data[16]['movavg']; ?>);

            data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
            data.setValue(17, 1, <?php echo $data[17]['svalue']; ?>);
            data.setValue(17, 2, <?php echo $data[17]['movavg']; ?>);

            data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
            data.setValue(18, 1, <?php echo $data[18]['svalue']; ?>);
            data.setValue(18, 2, <?php echo $data[18]['movavg']; ?>);

            data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
            data.setValue(19, 1, <?php echo $data[19]['svalue']; ?>);
            data.setValue(19, 2, <?php echo $data[19]['movavg']; ?>);

            var options = {'title':'Latest Internal Temperature data
collected by sensor <?php echo $sensori; ?>',
                           'width':450,
```

```
                        'height':450,
                        vAxis:{title: 'Temperature(degree Celsius,°C)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'##.##'},
                             hAxis: {title: 'Time',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div5'));
          chart.draw(data, options);
          }
</script>
<div id="chart_div5"></div>
```

dbconrssi.php

```
<?php

$sensori=$_GET['sensor'];

$data=array();
$sensors=19;

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $data[$sensors]=array();

  $result = mysql_query("select rssi,TIME(ttime) as stime from routing_info
inner join temp_results on routing_info.id=temp_results.id WHERE
```

```
routing_info.origin=".$sensori." and routing_info.hop_no=0 ORDER BY
routing_info.id DESC
LIMIT 20;");

  while($row = mysql_fetch_array($result)){
    $data[$sensors]=array();
    $data[$sensors]['rssi'] = $row['rssi'];
    $data[$sensors]['stime'] = $row['stime'];
     $sensors--;
  }
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'sensor id:<?php echo $sensori; ?>');
          data.addRows(20);
          data.setValue(0, 0, '<?php echo $data[0]['stime']; ?>');
          data.setValue(0, 1, <?php echo $data[0]['rssi']; ?>);

          data.setValue(1, 0, '<?php echo $data[1]['stime']; ?>');
          data.setValue(1, 1, <?php echo $data[1]['rssi']; ?>);

          data.setValue(2, 0, '<?php echo $data[2]['stime']; ?>');
          data.setValue(2, 1, <?php echo $data[2]['rssi']; ?>);

          data.setValue(3, 0, '<?php echo $data[3]['stime']; ?>');
          data.setValue(3, 1, <?php echo $data[3]['rssi']; ?>);

          data.setValue(4, 0, '<?php echo $data[4]['stime']; ?>');
          data.setValue(4, 1, <?php echo $data[4]['rssi']; ?>);

          data.setValue(5, 0, '<?php echo $data[5]['stime']; ?>');
```

```
data.setValue(5, 1, <?php echo $data[5]['rssi']; ?>);

data.setValue(6, 0, '<?php echo $data[6]['stime']; ?>');
data.setValue(6, 1, <?php echo $data[6]['rssi']; ?>);

data.setValue(7, 0, '<?php echo $data[7]['stime']; ?>');
data.setValue(7, 1, <?php echo $data[7]['rssi']; ?>);

data.setValue(8, 0, '<?php echo $data[8]['stime']; ?>');
data.setValue(8, 1, <?php echo $data[8]['rssi']; ?>);

data.setValue(9, 0, '<?php echo $data[9]['stime']; ?>');
data.setValue(9, 1, <?php echo $data[9]['rssi']; ?>);

data.setValue(10, 0, '<?php echo $data[10]['stime']; ?>');
data.setValue(10, 1, <?php echo $data[10]['rssi']; ?>);

data.setValue(11, 0, '<?php echo $data[11]['stime']; ?>');
data.setValue(11, 1, <?php echo $data[11]['rssi']; ?>);

data.setValue(12, 0, '<?php echo $data[12]['stime']; ?>');
data.setValue(12, 1, <?php echo $data[12]['rssi']; ?>);

data.setValue(13, 0, '<?php echo $data[13]['stime']; ?>');
data.setValue(13, 1, <?php echo $data[13]['rssi']; ?>);

data.setValue(14, 0, '<?php echo $data[14]['stime']; ?>');
data.setValue(14, 1, <?php echo $data[14]['rssi']; ?>);

data.setValue(15, 0, '<?php echo $data[15]['stime']; ?>');
data.setValue(15, 1, <?php echo $data[15]['rssi']; ?>);

data.setValue(16, 0, '<?php echo $data[16]['stime']; ?>');
data.setValue(16, 1, <?php echo $data[16]['rssi']; ?>);

data.setValue(17, 0, '<?php echo $data[17]['stime']; ?>');
data.setValue(17, 1, <?php echo $data[17]['rssi']; ?>);

data.setValue(18, 0, '<?php echo $data[18]['stime']; ?>');
```

```
          data.setValue(18, 1, <?php echo $data[18]['rssi']; ?>);

          data.setValue(19, 0, '<?php echo $data[19]['stime']; ?>');
          data.setValue(19, 1, <?php echo $data[19]['rssi']; ?>);

               var options = {'title':'Latest RSSI data (Data Collected by the
neighbors of the sensor)',
                         'width':450,
                         'height':450,
                         vAxis:{title: 'Received Signal Strength Indicator
(RSSI)(dB))',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out',
                              format:'#####'},
                              hAxis: {title: 'Time',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out'}
                         };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div8'));
          chart.draw(data, options);
          }
</script>
<div id="chart_div8"></div>
```

dbcounts.php

```
<meta http-equiv="refresh" content="60">
<?php

$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
```

```
}

$data=array();
$count=array();
$i=1;
$data[1]=1;
$data[2]=2;
$data[3]=3;
$count[1]=0;
$count[2]=0;
$count[3]=0;
$result = mysql_query("SELECT sensorid, COUNT(*) as counts FROM temp_results
GROUP BY sensorid ASC");


  while($row = mysql_fetch_array($result))
    {
    $data[$i]= $row['sensorid'];
    $count[$data[$i]]= $row['counts'];
    $i++;
    }
mysql_close($con);
?>


<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensor id');
               data.addColumn('number', 'Number of packets received');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor <?php echo $data[1]; ?>');
          data.setValue(0, 1, <?php echo $count[$data[1]]; ?>);

          data.setValue(1, 0, 'Sensor <?php echo $data[2]; ?>');
          data.setValue(1, 1, <?php echo $count[$data[2]]; ?>);
```

```
          data.setValue(2, 0, 'Sensor <?php echo $data[3]; ?>');
          data.setValue(2, 1, <?php echo $count[$data[3]]; ?>);

               var options = {'title':'Packets collected by Basestation',
                         'width':450,
                         'height':450,
                         };
          var chart = new
google.visualization.PieChart(document.getElementById('chart_div'));
          chart.draw(data, options);
//
          }
</script>
<div id="chart_div"></div>
```

dbconrssibase.php

```
<?php

//echo "<h1>".$sensori."</h1>";

//$chart = new GoogChart();
$data=array();
$sens=array();
$sensors=0;
$timenum=array();
$maxtimenum=0;
$tmp=0;
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
//echo "I'm here!";

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
```

```
    die ('Can\'t use sensorsdb : ' . mysql_error());
}
//echo "Can you see me?";
echo "<br />";

//$result2 = mysql_query("SELECT sensorid FROM sensors WHERE active=1");

//while($row2 = mysql_fetch_array($result2)) {

  $timenum[$row2['$sensori']]=0;
  $data[$row2['$sensori']]=array();
  $sens[$sensors]=$row2['$sensori'];

  $result = mysql_query("select rssi,origin as stime from routing_info inner
join temp_results on routing_info.id=temp_results.id WHERE
routing_info.destination=500 ORDER BY routing_info.id DESC LIMIT 20;");

  while($row = mysql_fetch_array($result))
    {
    $data[$row2['sensori']][$timenum[$row2['sensori']]]=array();
    $data[$row2['sensori']][$timenum[$row2['sensori']]]['rssi'] =
$row['rssi'];
    $data[$row2['sensori']][$timenum[$row2['sensori']]]['stime'] =
$row['stime'];
//      echo $data[$row2['sensori']][$timenum[$row2['sensori']]]['rssi'];
 //     echo "<br />";
    $timenum[$row2['sensori']]++;
    }
  $sensors++;
  if ($maxtimenum<$timenum[$row2['sensori']])
{$maxtimenum=$timenum[$row2['sensori']];}
//}
$flag2=1;
//echo "Can you see me?";
mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
```

```
          function func1()
          {
//             alert("calling Google!");
          if (<?php echo $timenum[$sens[0]];?>==20) {
          var sensors=<?php echo $sensors;?>;
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Time');
               data.addColumn('number', 'sensor id:<?php echo $sensori; ?>');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(20);
          data.setValue(0, 0, '<?php echo $data[$sens[0]][0]['stime']; ?>');
          data.setValue(0, 1, <?php echo $data[$sens[0]][0]['rssi']; ?>);

          data.setValue(1, 0, '<?php echo $data[$sens[0]][1]['stime']; ?>');
          data.setValue(1, 1, <?php echo $data[$sens[0]][1]['rssi']; ?>);

          data.setValue(2, 0, '<?php echo $data[$sens[0]][2]['stime']; ?>');
          data.setValue(2, 1, <?php echo $data[$sens[0]][2]['rssi']; ?>);

          data.setValue(3, 0, '<?php echo $data[$sens[0]][3]['stime']; ?>');
          data.setValue(3, 1, <?php echo $data[$sens[0]][3]['rssi']; ?>);

          data.setValue(4, 0, '<?php echo $data[$sens[0]][4]['stime']; ?>');
          data.setValue(4, 1, <?php echo $data[$sens[0]][4]['rssi']; ?>);

          data.setValue(5, 0, '<?php echo $data[$sens[0]][5]['stime']; ?>');
          data.setValue(5, 1, <?php echo $data[$sens[0]][5]['rssi']; ?>);

          data.setValue(6, 0, '<?php echo $data[$sens[0]][6]['stime']; ?>');
          data.setValue(6, 1, <?php echo $data[$sens[0]][6]['rssi']; ?>);

          data.setValue(7, 0, '<?php echo $data[$sens[0]][7]['stime']; ?>');
          data.setValue(7, 1, <?php echo $data[$sens[0]][7]['rssi']; ?>);

          data.setValue(8, 0, '<?php echo $data[$sens[0]][8]['stime']; ?>');
          data.setValue(8, 1, <?php echo $data[$sens[0]][8]['rssi']; ?>);

          data.setValue(9, 0, '<?php echo $data[$sens[0]][9]['stime']; ?>');
          data.setValue(9, 1, <?php echo $data[$sens[0]][9]['rssi']; ?>);
```

```
        data.setValue(10, 0, '<?php echo $data[$sens[0]][10]['stime']; ?>');
        data.setValue(10, 1, <?php echo $data[$sens[0]][10]['rssi']; ?>);

        data.setValue(11, 0, '<?php echo $data[$sens[0]][11]['stime']; ?>');
        data.setValue(11, 1, <?php echo $data[$sens[0]][11]['rssi']; ?>);

        data.setValue(12, 0, '<?php echo $data[$sens[0]][12]['stime']; ?>');
        data.setValue(12, 1, <?php echo $data[$sens[0]][12]['rssi']; ?>);

        data.setValue(13, 0, '<?php echo $data[$sens[0]][13]['stime']; ?>');
        data.setValue(13, 1, <?php echo $data[$sens[0]][13]['rssi']; ?>);

        data.setValue(14, 0, '<?php echo $data[$sens[0]][14]['stime']; ?>');
        data.setValue(14, 1, <?php echo $data[$sens[0]][14]['rssi']; ?>);

        data.setValue(15, 0, '<?php echo $data[$sens[0]][15]['stime']; ?>');
        data.setValue(15, 1, <?php echo $data[$sens[0]][15]['rssi']; ?>);

        data.setValue(16, 0, '<?php echo $data[$sens[0]][16]['stime']; ?>');
        data.setValue(16, 1, <?php echo $data[$sens[0]][16]['rssi']; ?>);

        data.setValue(17, 0, '<?php echo $data[$sens[0]][17]['stime']; ?>');
        data.setValue(17, 1, <?php echo $data[$sens[0]][17]['rssi']; ?>);

        data.setValue(18, 0, '<?php echo $data[$sens[0]][18]['stime']; ?>');
        data.setValue(18, 1, <?php echo $data[$sens[0]][18]['rssi']; ?>);

        data.setValue(19, 0, '<?php echo $data[$sens[0]][19]['stime']; ?>');
        data.setValue(19, 1, <?php echo $data[$sens[0]][19]['rssi']; ?>);

//        alert("Ready to draw!");
             var options = {'title':'Latest RSSI data Collected by the
basestation)',
                        'width':450,
                        'height':450,
                        vAxis:{title: 'Received Signal Strength Indicator
(RSSI)(dB))',
                             titleTextStyle: {color: '#5c5c5c'},
                             titlePosition: 'out',
```

```
                            format:'#####'},
                            hAxis: {title: 'Time',
                            titleTextStyle: {color: '#5c5c5c'},
                            titlePosition: 'out'}
                        };
          var chart = new
google.visualization.LineChart(document.getElementById('chart_div8'));
          chart.draw(data, options);
          }
          else { alert("Not enough data to display");}
//
          }
</script>
<div id="chart_div8"></div>
```

dbminmax.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM temp_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
```

```
  $data3[0]=$row['aver'];


  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM temp_results WHERE sensorid=2");


  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];


  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM temp_results WHERE sensorid=3");


  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];



mysql_close($con);
?>


<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
               data.addColumn('number', 'Average Value');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor 1');
          data.setValue(0, 1, <?php echo $data1[0]; ?>);
          data.setValue(0, 2, <?php echo $data2[0]; ?>);
          data.setValue(0, 3, <?php echo $data3[0]; ?>);
```

```
          data.setValue(1, 0, 'Sensor 2');
          data.setValue(1, 1, <?php echo $data1[1]; ?>);
          data.setValue(1, 2, <?php echo $data2[1]; ?>);
          data.setValue(1, 3, <?php echo $data3[1]; ?>);

          data.setValue(2, 0, 'Sensor 3');
          data.setValue(2, 1, <?php echo $data1[2]; ?>);
          data.setValue(2, 2, <?php echo $data2[2]; ?>);
          data.setValue(2, 3, <?php echo $data3[2]; ?>);



               var options = {'title':'Min, max and average values for
Temperature data',
                         'width':450,
                         'height':450,
                              vAxis:{title: 'Temperature(degree Celsius,°C)',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out',
                              format:'##,##'},
                              hAxis: {title: 'Sensors',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out'}
                         };
          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div1'));
          chart.draw(data, options);


          }
</script>
<div id="chart_div1"></div>
```

dbminmax2.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
```

```
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM hum_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM hum_results WHERE sensorid=2");

  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM hum_results WHERE sensorid=3");

  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];



mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

```
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
               data.addColumn('number', 'Average Value');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor 1');
          data.setValue(0, 1, <?php echo $data1[0]; ?>);
          data.setValue(0, 2, <?php echo $data2[0]; ?>);
          data.setValue(0, 3, <?php echo $data3[0]; ?>);

          data.setValue(1, 0, 'Sensor 2');
          data.setValue(1, 1, <?php echo $data1[1]; ?>);
          data.setValue(1, 2, <?php echo $data2[1]; ?>);
          data.setValue(1, 3, <?php echo $data3[1]; ?>);

          data.setValue(2, 0, 'Sensor 3');
          data.setValue(2, 1, <?php echo $data1[2]; ?>);
          data.setValue(2, 2, <?php echo $data2[2]; ?>);
          data.setValue(2, 3, <?php echo $data3[2]; ?>);


               var options = {'title':'Min, max and average values for
Humidity data',
                         'width':450,
                         'height':450,
                         vAxis:{title: 'Humidity(%RH)',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out',
                              format:'##,###%'},
                              hAxis: {title: 'Sensors',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out'}
```

```
                        };
          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div2'));
          chart.draw(data, options);


          }
</script>
<div id="chart_div2"></div>
```

dbminmax3.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $result = mysql_query("SELECT min(truevalue) as minim, max(truevalue) as
maxim, avg(truevalue) as aver FROM hum_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];

  $result = mysql_query("SELECT min(truevalue) as minim, max(truevalue) as
maxim, avg(truevalue) as aver FROM hum_results WHERE sensorid=2");
```

```
  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];

  $result = mysql_query("SELECT min(truevalue) as minim, max(truevalue) as
maxim, avg(truevalue) as aver FROM hum_results WHERE sensorid=3");

  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];

mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
               data.addColumn('number', 'Average Value');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor 1');
          data.setValue(0, 1, <?php echo $data1[0]; ?>);
          data.setValue(0, 2, <?php echo $data2[0]; ?>);
          data.setValue(0, 3, <?php echo $data3[0]; ?>);

          data.setValue(1, 0, 'Sensor 2');
          data.setValue(1, 1, <?php echo $data1[1]; ?>);
          data.setValue(1, 2, <?php echo $data2[1]; ?>);
          data.setValue(1, 3, <?php echo $data3[1]; ?>);
```

```
          data.setValue(2, 0, 'Sensor 3');
          data.setValue(2, 1, <?php echo $data1[2]; ?>);
          data.setValue(2, 2, <?php echo $data2[2]; ?>);
          data.setValue(2, 3, <?php echo $data3[2]; ?>);


               var options = {'title':'Min, max and average values for True
Humidity data',
                         'width':450,
                         'height':450,
                         vAxis:{title: 'True Humidity(%RH)',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out',
                              format:'##,###%'},
                              hAxis: {title: 'Sensors',
                              titleTextStyle: {color: '#5c8c8c'},
                              titlePosition: 'out'}
                         };
          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div3'));
          chart.draw(data, options);


          }
</script>
<div id="chart_div3"></div>
```

dbminmax4.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
```

```
$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}


  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM tsr_results WHERE sensorid=1");


  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];


  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM tsr_results WHERE sensorid=2");


  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];


  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM tsr_results WHERE sensorid=3");


  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];



mysql_close($con);
?>


<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
```

```
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Sensors');
            data.addColumn('number', 'Minimum Value');
            data.addColumn('number', 'Maximum Value');
            data.addColumn('number', 'Average Value');
            //var maxt=<?php echo $maxtimenum;?>;
        data.addRows(3);
        data.setValue(0, 0, 'Sensor 1');
        data.setValue(0, 1, <?php echo $data1[0]; ?>);
        data.setValue(0, 2, <?php echo $data2[0]; ?>);
        data.setValue(0, 3, <?php echo $data3[0]; ?>);

        data.setValue(1, 0, 'Sensor 2');
        data.setValue(1, 1, <?php echo $data1[1]; ?>);
        data.setValue(1, 2, <?php echo $data2[1]; ?>);
        data.setValue(1, 3, <?php echo $data3[1]; ?>);

        data.setValue(2, 0, 'Sensor 3');
        data.setValue(2, 1, <?php echo $data1[2]; ?>);
        data.setValue(2, 2, <?php echo $data2[2]; ?>);
        data.setValue(2, 3, <?php echo $data3[2]; ?>);


            var options = {'title':'Min, max and average values for TSR
data',
                    'width':450,
                    'height':450,
                    vAxis:{title: 'TSR(lux)',
                         titleTextStyle: {color: '#5c8c8c'},
                         titlePosition: 'out',
                         format:'#.###'},
                         hAxis: {title: 'Sensors',
                         titleTextStyle: {color: '#5c8c8c'},
                         titlePosition: 'out'}
                    };
        var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div4'));
        chart.draw(data, options);

        }
```

```
</script>
<div id="chart_div4"></div>
```

dbminmax5.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM par_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM par_results WHERE sensorid=2");

  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];
```

```
  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM par_results WHERE sensorid=3");

  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];


mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
               data.addColumn('number', 'Average Value');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor 1');
          data.setValue(0, 1, <?php echo $data1[0]; ?>);
          data.setValue(0, 2, <?php echo $data2[0]; ?>);
          data.setValue(0, 3, <?php echo $data3[0]; ?>);

          data.setValue(1, 0, 'Sensor 2');
          data.setValue(1, 1, <?php echo $data1[1]; ?>);
          data.setValue(1, 2, <?php echo $data2[1]; ?>);
          data.setValue(1, 3, <?php echo $data3[1]; ?>);

          data.setValue(2, 0, 'Sensor 3');
          data.setValue(2, 1, <?php echo $data1[2]; ?>);
          data.setValue(2, 2, <?php echo $data2[2]; ?>);
          data.setValue(2, 3, <?php echo $data3[2]; ?>);
```

```
            var options = {'title':'Min, max and average values for PAR
data',
                        'width':450,
                        'height':450,
                        vAxis:{title: 'PAR(lux)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'#.###'},
                             hAxis: {title: 'Sensors',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div5'));
          chart.draw(data, options);


          }
</script>
<div id="chart_div5"></div>
```

dbminmax6.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
```

```
}

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM involt_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM involt_results WHERE sensorid=2");

  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM involt_results WHERE sensorid=3");

  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];



mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
```

```
            data.addColumn('number', 'Average Value');
            //var maxt=<?php echo $maxtimenum;?>;
        data.addRows(3);
        data.setValue(0, 0, 'Sensor 1');
        data.setValue(0, 1, <?php echo $data1[0]; ?>);
        data.setValue(0, 2, <?php echo $data2[0]; ?>);
        data.setValue(0, 3, <?php echo $data3[0]; ?>);

        data.setValue(1, 0, 'Sensor 2');
        data.setValue(1, 1, <?php echo $data1[1]; ?>);
        data.setValue(1, 2, <?php echo $data2[1]; ?>);
        data.setValue(1, 3, <?php echo $data3[1]; ?>);

        data.setValue(2, 0, 'Sensor 3');
        data.setValue(2, 1, <?php echo $data1[2]; ?>);
        data.setValue(2, 2, <?php echo $data2[2]; ?>);
        data.setValue(2, 3, <?php echo $data3[2]; ?>);



            var options = {'title':'Min, max and average values for
Internal Voltage data',
                       'width':450,
                       'height':450,
                       vAxis:{title: 'Volt(Volts)',
                            titleTextStyle: {color: '#5c8c9c'},
                            titlePosition: 'out',
                            format:'#.###'},
                            hAxis: {title: 'Sensors',
                            titleTextStyle: {color: '#5c8c8c'},
                            titlePosition: 'out'}
                       };
        var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div6'));
        chart.draw(data, options);


        }
</script>
<div id="chart_div6"></div>
```

dbminmax7.php

```
<?php

$data1=array();
$data2=array();
$data3=array();
$con = mysql_connect('95.211.25.40','dpapanik_kate','katerina');
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

$db_selected=mysql_select_db('dpapanik_eco', $con);
if (!$db_selected) {
    die ('Can\'t use sensorsdb : ' . mysql_error());
}

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM intemp_results WHERE sensorid=1");

  $row = mysql_fetch_array($result);
  $data1[0]=$row['minim'];
  $data2[0]=$row['maxim'];
  $data3[0]=$row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM intemp_results WHERE sensorid=2");

  $row = mysql_fetch_array($result);
  $data1[1]=$row['minim'];
  $data2[1] = $row['maxim'];
  $data3[1] = $row['aver'];

  $result = mysql_query("SELECT min(svalue) as minim, max(svalue) as maxim,
avg(svalue) as aver FROM intemp_results WHERE sensorid=3");

  $row = mysql_fetch_array($result);
  $data1[2]=$row['minim'];
```

```
  $data2[2] = $row['maxim'];
  $data3[2] = $row['aver'];


mysql_close($con);
?>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
          google.load("visualization", "1", {packages:["corechart"]});
          google.setOnLoadCallback(func1);
          function func1()
          {
               var data = new google.visualization.DataTable();
               data.addColumn('string', 'Sensors');
               data.addColumn('number', 'Minimum Value');
               data.addColumn('number', 'Maximum Value');
               data.addColumn('number', 'Average Value');
               //var maxt=<?php echo $maxtimenum;?>;
          data.addRows(3);
          data.setValue(0, 0, 'Sensor 1');
          data.setValue(0, 1, <?php echo $data1[0]; ?>);
          data.setValue(0, 2, <?php echo $data2[0]; ?>);
          data.setValue(0, 3, <?php echo $data3[0]; ?>);

          data.setValue(1, 0, 'Sensor 2');
          data.setValue(1, 1, <?php echo $data1[1]; ?>);
          data.setValue(1, 2, <?php echo $data2[1]; ?>);
          data.setValue(1, 3, <?php echo $data3[1]; ?>);

          data.setValue(2, 0, 'Sensor 3');
          data.setValue(2, 1, <?php echo $data1[2]; ?>);
          data.setValue(2, 2, <?php echo $data2[2]; ?>);
          data.setValue(2, 3, <?php echo $data3[2]; ?>);


               var options = {'title':'Min, max and average values for
Internal Temperature data',
                         'width':450,
```

```
                        'height':450,
                        vAxis:{title: 'Internal Temperature(degree
Celsius,째C)',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out',
                             format:'##.##'},
                             hAxis: {title: 'Sensors',
                             titleTextStyle: {color: '#5c8c8c'},
                             titlePosition: 'out'}
                        };
          var chart = new
google.visualization.ColumnChart(document.getElementById('chart_div7'));
          chart.draw(data, options);

          }
</script>
<div id="chart_div7"></div>
```

ΑΝΑΦΟΡΕΣ

[1] Flávia C. Delicato, Paulo F. Pires, Luiz Rust, Luci Pirmez , José Ferreira de Rezende : *Reflective Middleware for Wireless Sensor Networks*

[2] *The Handbook of Computer Networks*, John Willey & Sons Inc, 2007

[3] Gang Lu, Debraj De, Wen-Zhan Song, Behrooz Shirazi, "*A Wake-On Sensor Network*", Sensorweb Research Laboratory, Washington State University, Vancouver WA 98686-9600, US.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "*Wireless sensor network: a survey*", Computer Networks, 38(4), 393–422, March 2002.

[5] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kularni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "*Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking*", In Special Issue of Elsevier Computer Networks on Future.

[6] http://news.directindustry.com/press/mitsubishi-electric-europe/sensor-based-real-time-control-with-industrial-robots-from-mitsubishi-electric-12225-333905.html

[7] http://www.ferret.com.au/c/Endress-Hauser-Australia/Ultrasonic-Level-Sensors-from-Endress-Hauser-p16407

[8] http://home.cogeco.ca/~kilbrideconnectivity/smart_house.htm

[9] L. Schwiebert, S.K.S. Gupta, and J. Weinamann, "*Research challenges in wireless networks of biomedical sensors*", ACM SIGMOBILE 2001, 151–165, Rome, July 2001.

[10] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "*Energy-efficient communication protocol for wireless microsensor networks*", HICSS 2000, 8020–8029, Maui, January 2000.

[11] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "*Data gathering algorithms in sensor networks using energy metrics*", IEEE Trans. Parallel Distributed Syst., 13(9), 924–935, September 2002.

[12] Willig, R. Shah, J. Rabaey, and A. Wolisz, "*Altruists in the PicoRadio sensor network*", 4th IEEE Int. Workshop Factory Commun. Syst., 175–184, Sweden, August 2002.

[13] J. Agre and L. Clare, "*An integrated architecture for cooperative sensing networks*", Computer Mag., 33(5), 106–108, 2000.

[14] J. Mirkovic, G.P. Venkataramani, S. Lu, and L. Zhang, "*A self-organizing approach to data forwarding in large-scale sensor networks*", IEEE ICC, 5, 1357–1361, St. Petersburg, Russia, 2001.

[15] Chandrakasan et al. "*Design considerations for distributed microsensor systems*", IEEE 1999 Custom Integrated Circuits Conf., 279–286, San Diego, May 1999.

[16] J. Feng, F. Koushanfar, and M. Potkonjak, "*System-architectures for sensor networks issues, alternatives, and directions*", IEEE ICCD'02: VLSI Computers Processors, 226–231, Freiburg, Germany, 2002.

[17] W. Heinzelman, "*Application-specific protocol architecture for wireless networks*", Ph.D. dissertation, Massachusetts Institute of Technology, June 2000.

[18] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci “A Survey on Sensor Networks.”, August 2002.

[19] G. Pottie, L .Clareb, *"Wireless integrated network sensors: toward low-cost and robust self-organizing security networks"*, Proc. SPIE, Sensors, C3I, Vol. 3577, p. 86-95, 1999.

[20] A. Hac, “*Wireless Sensor Network Designs*”, Wiley & Sons, Ltd, 2003.

[21] I. Mohammad and I. Mahgoub, “*Handbook of sensor networks: compact wireless and wired sensing systems***”**, CRC Press, 2005.

[22] P. Kikiras, **“***Sensor Networks for Pervasive Computing***”**, PhD Thesis, National Technical University of Athens, 2005.

[23] S.S. Dhillon, K. Chakrabarty, and S.S. Iyengar, “*Sensor placement for grid coverage under imprecise detections*”, Int. Conf. Inf. Fusion (FUSION) 2002, 2, 1581–1587, Annapolis, 2002.

[24] S. Tilak, N.B. Abu–Ghazaleh, and W. Heinzelman, “*Infrastructure trade-offs for sensor networks*”, ACM WSNA’02, 49–58, September 2002.

[25] F. Pianegiani, A. Boni, M. Hu, D. Petri, **“***Energy-Aware Signals Classification in Ad-hoc Wireless Sensor Networks***”**, Instrumentation and Measurement Technology Conference, IMTC, Ottawa, Canada, May 17-19, 2005.

[26] F. Zhao, L. Guibas, “*Wireless Sensor Networks, First Edition: An Information Processing Approach***”**, Morgan Kaufmann, 2004.

[27] G. Pottie, L .Clareb, "*Wireless integrated network sensors: toward low-cost and robust self-organizing security networks*", Proc. SPIE, Sensors, C3I, Vol. 3577, p. 86-95, 1999.

[28] D. Estrin, D. Culler, K. Pister, and G. Sukhatme, **“***Connecting the Physical World with Pervasive Networks***”**, IEEE Pervasive Computing, pp. 59-69, January-March 2002.

[29] Z. Hu, B. Li, *"On the Fundamental Capacity and Lifetime Limits of Energy-Constrained Wireless Sensor Networks"*, Proceedings of the 10th IEEE Real- Time and Embedded Technology and Applications Symposium, pp.2-9, 25-28 May, Toronto, Canada, 2004.

[30] Γ. Μαζαράκης, Διδακτορική Διατριβή ‘ *Τεχνικές Χαμηλής Κατανάλωσης Ενέργειας για Ανίχνευση Προσώπων και Κατηγοριοποίηση Οχημάτων στα Ασύρματα Δίκτυα Αισθητήρων*’, Ε.Μ.Π, Αθήνα, Οκτώβριος 2007.

[31] F. Zhao et al., “*Collaborative signal and information processing: an information directed approach*”, Proc. IEEE, 91(8), 1199–1209, 2003.

[32] G. Gupta and M. Younis, “*Load-balanced clustering of wireless sensor networks*”, IEEE ICC 2003, 3, 1848–1852, May 2003.

[33] S. Bandyopadhyay and E.J. Coyle, “*An energy efficient hierarchical clustering algorithm for wireless sensor networks*”, IEEE INFOCOM 2003, 3, 1713–1723, San Francisco, March–April 2003.

[34] http://www.sensirion.com/en/01_humidity_sensors/02_humidity_sensor_sht11.htm

[35] http://jp.hamamatsu.com/resources/products/ssd/pdf/s1087_etc_kspd1039e01.pdf

[36] http://docs.TinyOS.net/tinywiki/index.php/Boomerang_ADC_Example

[37] Tmote Sky Datasheet [pdf]

[38] Tmote Sky Schematic [pdf]

[39] David Gay, Philip Levis, David Culler, and Eric Brewer, "*nesC 1.1 Language Reference Manual*", May 2003.

[40] J. Hill et al., "*TinyOS: operating system for sensor networks*", available at hppt://TinyOS.millennium.berkeley.edu, 2003.

[41] Crossbow Technology Inc. "*TinyOS Getting Started Guide*", available at www.xbow.com.

[42] Ruben Pinilla, Nacho Navarro, and Marisa Gil, "*TinyOS: A Case of Study for Adaptable Embedded Applications Based on Sensor Networks*", Jordi Girona, 1-3, Campus Nord, 08034 Barcelona, Spain.

[43] tinything.googlecode.com/files/**TinyOS_Tutorial_2**.pdf

[44] http://docs.TinyOS.net/tinywiki/index.php/Tymo

[45] C. Intanagonwiwat, R. Govindad, D. Estrin, "*Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*"

[46] F. Ye, A. Chen, S. Lu, L. Zhang, "*A Scalable Solution to Minimum Cost Forwarding in langeSensor Networks*", UCLA Computer Science Dep., Los Angeles, CA 90095-1596.

[47] C. E. Perkins, E. M. Royer, "*Ad-hoc On-Demand Distance Vector Routing*", Dep. Of Computer Science, University of California, Santa Barbara.

[48] I. D. Chakeres, E. M. Brlding-Royer, " *AODV Implementation Design and Performance Evaluation*", Dep. Of Computer Science, University of California, Santa Barbara

[49] I. D. Chakeres, C. E. Perkins, "*Dynamic MAMET On-demand Routing Protocol*", IETF Internet Draft, February 2008 (work in progress).

[50] J.J. Galvez, P. M. Ruiz, "*Design and Performance Evaluation of Multipath Extensions for DYMO Protocol*" Dep. Of Information and Communication Engineering, Computer Science Faculty, University of Murcia.

[51] http://tools.ietf.org/html/draft-ietf-manet-dymo-21

[52] Tobias Pilsak#1, Jan Luiken ter Haseborg#2 "*EMC investigations of wireless sensor networks within the engine room of vessels*" Institute of Measurement Technology and EMC,Hamburg University of Technology 2009

[53] http://www.vmware.com/products/player/

[54] http://releases.ubuntu.com/8.04/

[55] http://code.google.com/p/TinyOS-main/

[56] http://www.oracle.com/technetwork/java/javase/downloads/java-se-jdk-7-download-432154.html

[57] http://docs.TinyOS.net/tinywiki/index.php/Installing_TinyOS_2.1.1#Manual_installation_on_your_host_OS_with_RPMs

[58] http://docs.TinyOS.net/tinywiki/index.php/Installing_from_SVN/GIT

[59] http://www.ftdichip.com/Drivers/VCP.htm

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

[61] http://www.TinyOS.net/dist-2.0.0/tools/windows/msp430tools-base-0.1-20050607.cygwin.i386.rpm

[62] http://www.TinyOS.net/dist-2.0.0/tools/windows/msp430tools-python-tools-1.0-1.cygwin.noarch.rpm

[63] http://www.TinyOS.net/dist-2.0.0/tools/windows/msp430tools-binutils-2.16-20050607.cygwin.i386.rpm

[64] http://www.TinyOS.net/dist-2.0.0/tools/windows/msp430tools-gcc-3.2.3-20050607.cygwin.i386.rpm

[65] http://www.TinyOS.net/dist-2.1.0/tools/windows/msp430tools-libc-20080808-1.cygwin.i386.rpm

[66] http://TinyOS.stanford.edu/TinyOS-rpms/nesC-1.3.1-1.cygwin.i386.rpm

[67] http://www.TinyOS.net/dist-2.1.0/TinyOS/windows/TinyOS-deputy-1.1-1.cygwin.i386.rpm

[68] http://TinyOS.stanford.edu/TinyOS-rpms/TinyOS-tools-1.4.0-3.cygwin.i386.rpm

[69] http://TinyOS.stanford.edu/TinyOS-rpms/TinyOS-2.1.1-3.cygwin.noarch.rpm

[70] *'A Closer Look at the Advanced CODAS Moving Average Algorithm'*, DATAQ INSTRUMENTS

[71] *'Moving Average Filters'*, The Scientist and Engineer's Guide to Digital Signal Processing, Chapter 15

[72] S.Park, A. Savvides, M. Srivastava, "*Battery Capacity Measurement and Analysis using Lithium Coin Cell Battery*", University of California, LA. ISLPED '01, Aug 6-7, 2001, Huntington Beach, California, USA.

[73] Ι. Νέλλας, Σ. Πινάτσης, Διπλωματική Εργασία, ' *Μετάδοση και Επεξεργασία Ζωτικών Σημείων μέσω Ασύρματων Δικτύων Αισθητήρων*', Ε.Μ.Π., Αθήνα, Μάρτιος 2006.

[74] Γιώργος Π. Δρουσιώτης, Διπλωματική Εργασία '*Μέθοδοι Παρακολούθησης της Κατανάλωσης Ενέργειας σε Δίκτυα Αισθητήρων',* Ε.Μ.Π., Αθήνα, 2011

[75] Κωνσταντίνος Κωνσταντινίδης, Διπλωματική Εργασία ' *Κινητικότητα στα Ασύρματα Δίκτυα Αισθητήρων',* Πανεπιστήμιου Κύπρου, Ιούνιος 2009

[76] Κερασιώτης Φώτιος,Διπλωματική Εργασία '*Μελέτη και σχεδίαση ασύρματου δικτύου αισθητήρων με έμφαση σε εφαρμογή ιχνηλάτισης*', Πανεπιστήμιο Πάτρας, Πάτρα 2006

[77] Μαρία Ταραμίγκου, Διπλωματική εργασία '*Ενσωμάτωση Ασύρματων Αισθητήρων σε προγραμματιστικό περιβάλλον',* Ε.Μ.Π., Αθήνα, Σεπτέμβριος 2008

[78] Σπηλιόπουλος Χρήστος, Διπλωματική εργασία '*Ολοκληρωμένο σύστημα παρακολούθησης περιβαλλοντικών συνθηκών και τηλε-ειδοποίησης σε γεωργικές εφαρμογές'*