Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά Συστήματα

Σχεδιασμός και Ανάπτυξη Ενός Ολοκληρωμένου Συστήματος Παρακολούθησης Περιβαλλοντικών Συνθηκών Μέσω Ενός Ασύρματου Δικτύου Αισθητήρων

Θωμάς Καρανίκας

Επιβλέπων Καθηγητής
Χρήστος Γκουμόπουλος

Μάϊος 2013

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Σχεδιασμός και Ανάπτυξη Ενός Ολοκληρωμένου Συστήματος Παρακολούθησης Περιβαλλοντικών Συνθηκών Μέσω Ενός Ασύρματου Δικτύου Αισθητήρων

Θωμάς Καρανίκας

Επιβλέπων Καθηγητής
Χρήστος Γκουμόπουλος

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάϊος 2013

Περίληψη

Η εξέλιξη της τεχνολογίας και οι ανάγκες της κοινωνίας για συνεχόμενη πληροφόρηση έχουν οδηγήσει στη δημιουργία αισθητήρων για τη μέτρηση και παρακολούθηση διαφόρων συνθηκών. Στην εξέλιξη αυτού του γεγονότος έχουν κάνει την εμφάνισή τους τα τελευταία χρόνια τα ασύρματα δίκτυα αισθητήρων (Wireless Sensor Networks). Σήμερα αποτελούν ένα ενεργό τομέα έρευνας ώστε να υπάρξει βελτίωση των συγκεκριμένων τεχνολογιών αλλά και των τρόπων χρησιμοποίησής τους.

Σκοπός της παρούσας διατριβής είναι η δημιουργία ενός ασύρματου δικτύου αισθητήρων που αποτελείται από αυτόνομους κόμβους-αισθητήρες, οι οποίοι είναι υπεύθυνοι για την παρακολούθηση των περιβαλλοντικών συνθηκών. Στη συνέχεια μελετάται ο τρόπος μετάδοσης προς ένα συγκεκριμένο κέντρο συλλογής των δεδομένων, την περαιτέρω επεξεργασίας τους, την αποθήκευσή τους και την απεικόνισή τους σε μορφή φιλική προς τον χρήστη. Τα μεγέθη τα οποία μετρούνται είναι η θερμοκρασία, η υγρασία, το φως και η υγρασία εδάφους.

Αρχικά μελετάμε τον κόμβο αισθητήρα και τα συστατικά του μέρη. Στη συνέχεια μελετάμε τα ασύρματα δίκτυα αισθητήρων. Μέσα σε αυτό το πλαίσιο αναφέρουμε τις υπάρχουσες εφαρμογές των συγκεκριμένων δικτύων στη σύγχρονη εποχή και αναλύουμε τους τρόπους επικοινωνίας και της κατανάλωσης ενέργειας καθώς και τις διάφορες προκλήσεις και απαιτήσεις τους.

Στη συνέχεια γίνεται παρουσίαση της πλατφόρμας Micaz της εταιρίας xbow, η οποία αποτελεί την μονάδα που θα χρησιμοποιηθεί για τη δημιουργία του ασύρματου δικτύου της παρούσας διατριβής, περιλαμβάνοντας την περιγραφή των συστατικών και χαρακτηριστικών της.

Για το σχεδιασμό και την διαχείριση των πλατφορμών χρησιμοποιήθηκε το λειτουργικό σύστημα TinyOS που χρησιμοποιεί την γλώσσα προγραμματισμού NesC καθώς και το λογισμικό Moteworks το οποίο δίνεται από την εταιρία που κατασκεύασε τους κόμβους την Crossbow.

Summary

The evolution of technology and the needs of society for continuous information have lead to the creation of sensors for measuring and monitoring various conditions. In the evolution of this event have appeared in the last years the WSNs (Wireless Sensor Networks). Today they are an active area of research in order to be improved the specific technologies and the ways we use them.

The purpose of this thesis is to create a wireless sensor network for monitoring environmental conditions. Then study the broadcast for a particular center data collection, their further processing, storage and their presentation in a user-friendly form. We are measure temperature, light and soil moisture.

Initially we study the sensor node and all of its components. Then we study the wireless sensor networks. Within this context we mention the existing applications of these networks in the modern era and analyze the ways of communication and energy consumption as well as the various challenges and requirements.

Next we present the platform Micaz of the company xbow, which is the unit that will be used to create a wireless network of this dissertation, including the description of the components and characteristics.

For the design and management of the operating platforms was used TinyOS using NesC programming language and the software Moteworks which is given by the company that built the nodes the xbow.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Χρήστο Γκουμόπουλο για την ανάθεση της παρούσας διατριβής και για την ευκαιρία που μου παρείχε ώστε να μπορέσω να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, καθώς και για την καθοδήγηση και υποστήριξη κατά τη διάρκεια της υλοποίησής της.

Επίσης, θα ήθελα να ευχαριστήσω μέσα από την καρδιά μου την μητέρα μου για την ηθική και ουσιαστική υποστήριξη και συμπαράσταση σε όλη την διάρκεια της φοίτησής μου και της πίστης της σε μένα. Σε αυτήν χρωστάω τα πάντα.

Περιεχόμενα

1 Εισαγωγή 1

1.1 Σκοπός της Μεταπτυχιακής Διατριβής 1

1.2 Περιγραφή Συστήματος 2

2 Ασύρματα Δίκτυα Αισθητήρων 4

2.1 Ασύρματα Δίκτυα Αισθητήρων 4

2.1.1 Γενικά 4

2.1.2 Χαρακτηριστικά 6

2.1.3 Προβλήματα και Περιορισμοί 7

2.2 Εφαρμογές Ασύρματων Δικτύων Αισθητήρων 8

2.2.1 Περιβαλλοντικές Εφαρμογές 9

2.2.2 Στρατιωτικές Εφαρμογές 11

2.1.3 Εφαρμογές Υγείας 12

2.2.4 Οικιακές Εφαρμογές 13

2.2.5 Εφαρμογές για τον Έλεγχο της Κίνησης 14

2.2.6 Βιομηχανικές Εφαρμογές 14

2.3 Μοντέλα Επικοινωνίας στα Ασύρματα Δίκτυα Αισθητήρων 15

2.4 Αρχιτεκτονική στα Ασύρματα Δίκτυα Αισθητήρων 16

2.4.1 Το Φυσικό Επίπεδο 17

2.4.2 Το Επίπεδο Ζεύξης 17

2.4.3 Το Επίπεδο Δικτύου 18

2.4.4 Το Επίπεδο Μεταφοράς 19

2.4.5 Το Επίπεδο Εφαρμογών 19

2.5 Τεχνικές προκλήσεις και απαιτήσεις των Ασύρματων Δικτύων Αισθητήρων 20

2.5.1 Τεχνικές προκλήσεις των Ασύρματων Δικτύων Αισθητήρων 20

2.5.2 Στόχοι σχεδιασμού των ασύρματων δικτύων αισθητήρων 22

3 Το υλικό 26

3.1 Η Πλατφόρμα Micaz 27

3.1.1 Τεχνικά χαρακτηριστικά του Micaz 27

3.1.2 Ο μικροελεγκτής ATmega128L 29

3.1.3 Ο πομποδέκτης CC2420 RF 33

3.1.4 Τροφοδοσία και Κατανάλωση Ενέργειας 36
3.2 Η πλατφόρμα MIB520CB 37
3.3 Η πλακέτα αισθητήρων MDA100CA 38
3.3.1 Ο αισθητήρας θερμοκρασίας του MDA100CA. 39
3.3.2 Ο αισθητήρας φωτός του MDA100CA. 40
3.4 Η πλακέτα αισθητήρων MDA300CA. 41
3.5 Ο Αισθητήρας υγρασίας ECH2O-10H. 42
3.6 Συνδεσμολογία του υλικού 44

4 Το Λειτουργικό TinyOS 47
4.1 Εισαγωγή 47
4.2 Απαιτήσεις σχετικά με την σχεδίαση λειτουργικών συστημάτων για Ασύρματα Δίκτυα Αισθητήρων 48
4.3 Η δημιουργία του TinyOS 49
4.4 Η Αρχιτεκτονική του TinyOS 50
4.5 Τα βασικά στοιχεία του TinyOs 51
4.5.1 Τα Συστατικά 51
4.5.2 Εντολές 52
4.5.3 Γεγονότα 52
4.5.4 Διεργασίες 53
4.5.5 Χρονοπρογραμματιστής 53
4.6 Η γλώσσα προγραμματισμού NesC 54
4.6.1 Διεπαφές 54
4.6.2 Modules 55
4.6.3 Configuration 57
4.6.4 Οι Διεργασίες στην NesC 59
4.6.4 Async συναρτήσεις και Atomic δηλώσεις 60

5 Το Πρωτόκολλο Επικοινωνίας XMesh 62
5.1 Εισαγωγή 62
5.2 Μονάδες Δικτύου 63
5.3 Χαρακτηριστικά 63

5.3.1 TrueMesh 64
5.3.2 Quality of Service (QoS) υπηρεσίες 64
5.3.3 Πολλαπλά είδη μετάδοσης 65
5.3.4 Health Diagnostics 65
5.3.5 Ενεργειακές Καταστάσεις 65
5.3.6 OTAP(Over the Air Programming) 66
5.3.7 Συγχρονισμός στον χρόνο 66
5.3.8 Watch Dog 66
5.4 Διαχείριση Ενέργειας 66
5.4.1 XMesh HP 67
5.4.2 XMesh LP 67
5.4.3 XMesh ELP 68
5.5 Σχηματισμός Multi-Hop Δικτύου 69
5.5.1 Εκτίμηση Σήματος 69
5.5.2 Επιλογή Γονέα 70
5.5.3 Route Update Messages 70
5.6 Αποστολή και Λήψη Πακέτων 71
5.6.1 XMesh Messaging API 71
5.6.1.1 MhopSend Interface 72
5.6.1.2 Receive και ReceiveAck Interface 73
5.6.1.3 Intercept και Snoop Interface 74
5.6.1.4 PromisciousSniff Interface 74
5.6.1.5 RouteControl Interface 74
5.6.2 Health Packet 75

6 Απαιτήσεις και Σχεδιασμός Συστήματος 76
6.1 Επίπεδο 1: Απαιτήσεις και Σχεδιασμός του λογισμικού των μονάδων 77
61.1 Λογισμικό Κόμβων 77
6.1.2 Λογισμικό σταθμού-βάσης 80
6.2 Επίπεδο 2: Απαιτήσεις και Σχεδιασμός της βάσης δεδομένων 81
6.2.1 Επικοινωνία σταθμού-βάσης με βάση δεδομένων 81
6.2.2 Απαιτήσεις βάσης δεδομένων 81
6.2.3 Πίνακες βάσης δεδομένων 82
6.3 Επίπεδο 3: Απαιτήσεις και Σχεδιασμός του ιστότοπου 85

7 Υλοποίηση Συστήματος 87
7.1 Υλοποίηση επιπέδου 1 88
7.1.1 Εργαλεία υλοποίησης επιπέδου 1 88
7.1.2 Λογισμικό κόμβων-αισθητήρων 90
7.1.2.1 Λογισμικό κόμβου-αισθητήρα MDA100 91
7.1.2.2 Λογισμικό κόμβου-αισθητήρα MDA300 94
7.1.3 Λογισμικό σταθμού-βάσης 97
7.2 Υλοποίηση επιπέδου 2 101
7.3 Υλοποίηση επιπέδου 3 103

8 Συμπεράσματα 107
8.1 Γενικά Συμπεράσματα 108
8.2 Συμπεράσματα από την Χρήση του Συστήματος 109
8.3 Προβλήματα και Τρόποι Αντιμετώπισής 110
8.4 Μελλοντική Εργασία 110

Βιβλιογραφία 111

A Κώδικας Motes A-1
A.1 Κώδικας Κόμβου-αισθητήρα MDA100 ..... A-1
A.1.1 MDA100.nc A-1
A.1.2 MDA100M.nc A-3
A.1.3 Sesnsorapp.h A-7
A.1.4 Makefile A-8
A.2 Κώδικας Κόμβου-αισθητήρα MDA300 ..... A-8
A.2.1 MDA300.nc A-8
A.2.2 MDA300M.nc A-9
A.2.3 SesnsorApp.h A-13
A.2.4 Makefile A-14
A.3 Κώδικας Σταθμού-Βάσης A-15
A.3.1 XMeshBaseStation.nc A-15

A.3.2 XMeshBaseStationM.nc..... A-16
A.3.3 Sesnsorapp.h..... A-17
A.3.4 Makefile..... A-18
Β Πρόγραμμα Δημιουργίας Βάσης Δεδομένων..... Β-1
Γ Κώδικας Δημιουργίας Ιστότοπου..... Γ-1
Γ.1 Κώδικας CSS..... Γ-1
Γ.2 Κώδικας Index.html..... Γ-6
Γ.3 Κώδικας Temperature.php..... Γ-8
Γ.4 Κώδικας getlinechartdatatemp.php..... Γ-13
Γ.5 Κώδικας chartindextemp.php..... Γ-15
Γ.6 Κώδικας light.php..... Γ-19
Γ.7 Κώδικας getlinechartdatalight.php..... Γ-24
Γ.8 Κώδικας chartindexlight.php..... Γ-26
Γ.9 Κώδικας moisture.php..... Γ-32
Γ.10 Κώδικας getlinechartdatamoisture.php..... Γ-39
Γ.11 Κώδικας chartindexmoisture.php..... Γ-41
Γ.12 Κώδικας health.php..... Γ-45

Κεφάλαιο 1
Εισαγωγή

Η ραγδαία εξέλιξη των ασύρματων τεχνολογιών καθώς και η σημαντική αύξηση των απαιτήσεων των χρηστών για πρόσβαση στις πληροφορίες με τις κινητές συσκευές (κινητά, ταμπλέτες, φορητοί υπολογιστές) έχουν κάνει τις ασύρματες επικοινωνίες ένα απαραίτητο μέσο για την μεταφορά της πληροφορίας σχεδόν σε όλους τους τομείς της καθημερινότητας. Επίσης η ανάγκη για την εξοικονόμηση ενέργειας αλλά και της μείωσης του όγκου των συσκευών έχει δώσει την δυνατότητα για ανάπτυξη πολλών συσκευών χαμηλού κόστους και ισχύος που υλοποιούν ένα πλήθος εφαρμογών.

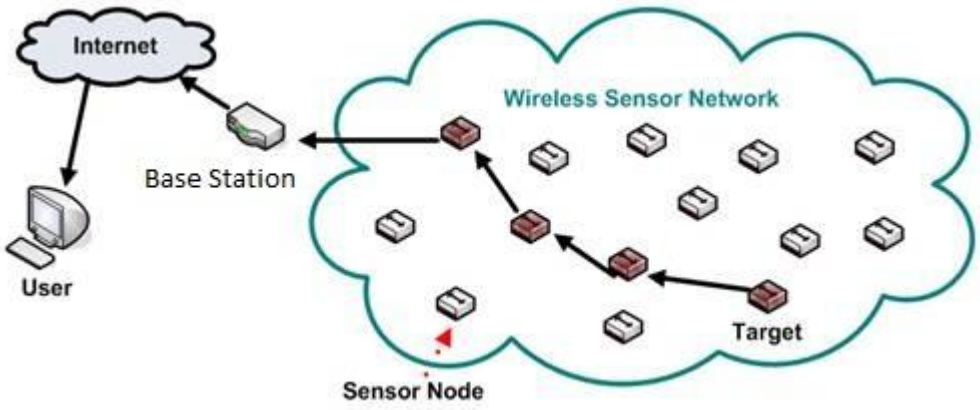
1.1. Σκοπός της μεταπτυχιακής διατριβής

Σκοπός της παρούσας μεταπτυχιακής διατριβής είναι ο σχεδιασμός και η υλοποίηση ενός ασύρματου δικτύου αισθητήρων για την παρακολούθηση περιβαλλοντικών συνθηκών. Το σύστημα θα μας παρέχει μετρήσεις Θερμοκρασίας, Υγρασίας Εδάφους και Φωτός. Τα δεδομένα θα αποστέλλονται από τους κόμβους-αισθητήρες που θα κάνουν τις μετρήσεις σε ένα σταθμό-βάση, είτε απευθείας ο κόμβος-αισθητήρας στον σταθμό-βάση είτε ο κόμβος-αισθητήρας σε ένα άλλο κόμβο-αισθητήρα και από εκεί στον σταθμό-βάση. Στην συνέχεια θα δημιουργήσουμε ένα λογισμικό με το οποίο θα γίνεται η απεικόνιση των μετρήσεων στην κονσόλα του λειτουργικού

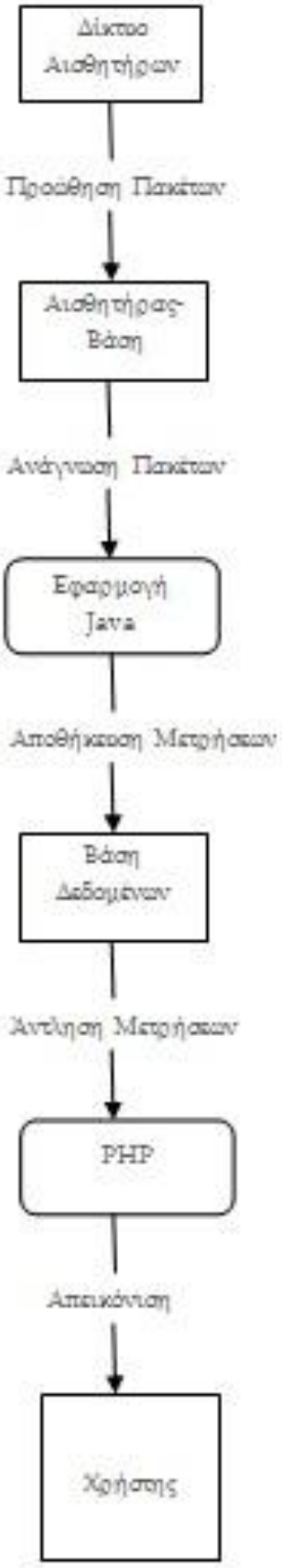
και η εξαγωγή τους σε μια βάση δεδομένων . Η υλοποίηση θα ολοκληρωθεί με την απεικόνιση της γραφικής αναπαράστασης της πληροφορίας για πληρέστερη κατανόηση από το τελικό χρήστη. Επιπλέον θα διερευνηθούν οι διαθέσιμες τεχνολογίες και η πρακτική εφαρμογή τους.

1.2. Περιγραφή συστήματος

Το ασύρματο δίκτυο αισθητήρων του συστήματος αποτελείται από μονάδες της εταιρίας crossbow που έχουν πάνω τους μονάδες αισθητήρες, μονάδα επεξεργασίας καθώς και μονάδα ασύρματης επικοινωνίας. Αυτές οι μονάδες σχηματίζουν ένα δίκτυο το οποίο προωθεί τις μετρήσεις του σε μια μονάδα που βρίσκεται συνδεδεμένη στον υπολογιστή μέσω του σταθμού βάσης. Ο προγραμματισμός τους υλοποιήθηκε σε γλώσσα προγραμματισμού NesC και αξιοποιεί υπηρεσίες του λειτουργικού συστήματος TinyOS 1.x ενώ για την υλοποίηση της multi-hop λειτουργίας χρησιμοποιήθηκε το πρωτόκολλο XMesh. Οι μετρήσεις που φτάνουν στο σταθμό-βάση επεξεργάζονται και καταχωρούνται σε μια βάση δεδομένων, η οποία δημιουργήθηκε με PostgreSQL, χρησιμοποιώντας μια java εφαρμογή. Επιπλέον η εξαγωγή των δεδομένων από τη βάση δεδομένων στον server υλοποιείται με τη γλώσσα προγραμματισμού PHP.Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική του συστήματος και το διάγραμμα ροής αντίστοιχα.



Εικόνα 1.1 Αρχιτεκτονική συστήματος



Εικόνα 1.2 Διάγραμμα ροής συστήματος

Κεφάλαιο 2
Ασύρματα Δίκτυα Αισθητήρων

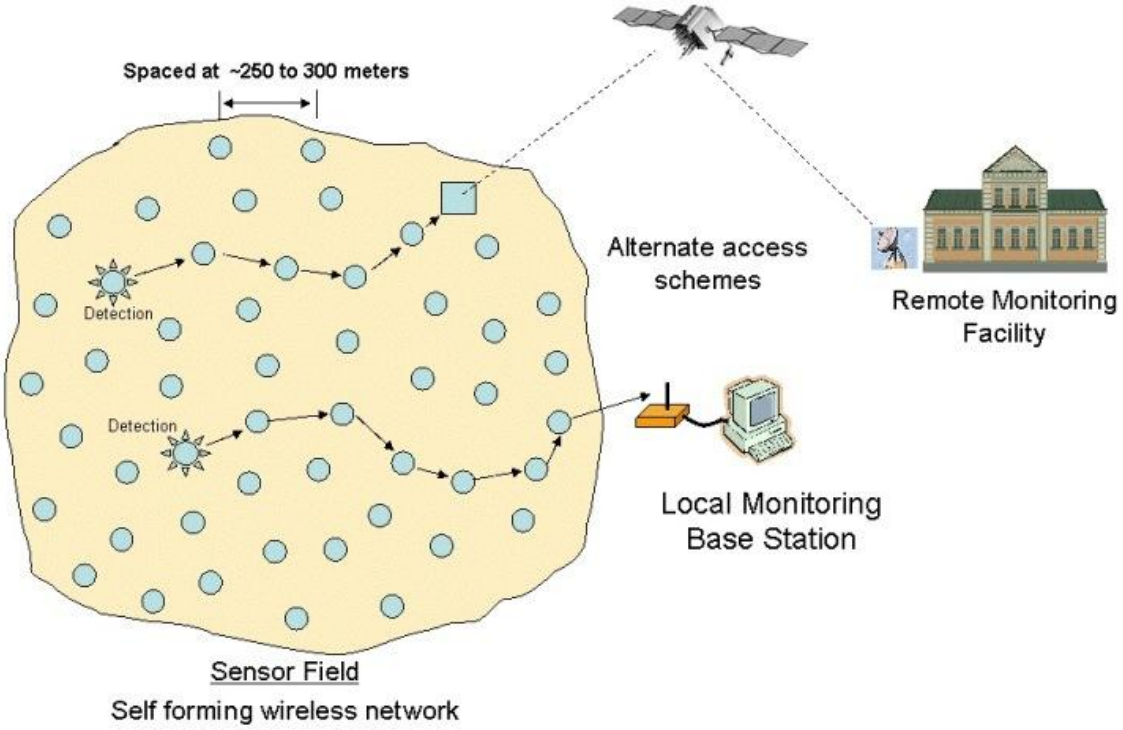
2.1 Ασύρματα δίκτυα αισθητήρων

2.1.1 Γενικά

Ένα ασύρματο δίκτυο αισθητήρων είναι ένα δίκτυο που αποτελείται από πολλές μικρές ανεξάρτητες μονάδες αισθητήρων κατάλληλα τοποθετημένες είτε μέσα στο φαινόμενο παρατήρησης είτε πολύ κοντά σε αυτό. Συνδυάζει ένα ευρύ φάσμα των τεχνολογιών της πληροφορικής όσον αφορά το υλικό, το λογισμικό, τις δικτυακές τεχνολογίες καθώς και τις μεθοδολογίες προγραμματισμού.

Η αρχική θέση των αισθητήρων δεν είναι απαραίτητο να είναι σχεδιασμένη ή προκαθορισμένη. Αυτό επιτρέπει την ανάπτυξη του ασύρματου δικτύου αισθητήρων σε τυχαίες θέσεις σε περιβάλλοντα μη προσπελάσιμα από τον άνθρωπο. Αυτό έχει σαν προϋπόθεση ότι τα πρωτόκολλα αλλά και οι αλγόριθμοι των δικτύων αυτών να οργανώνονται από μόνα τους και να

αναμεταδίδονται σε γειτονικούς αισθητήρες μέχρι να φτάσουν στον επιθυμητό προορισμό. Ένα τέτοιο παράδειγμα δίνεται στην παρακάτω εικόνα.



Εικόνα 2.1 Ασύρματο Δίκτυο Αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων έχουν την ικανότητα να αναπτυχθούν σε αφιλόξενες για τον άνθρωπο περιοχές και να παραμείνουν για χρόνια παρακολουθώντας και κάνοντας μετρήσεις χωρίς να χρειάζονται επαναφόρτιση οι πηγές ενέργειας των αισθητήρων.

Ένα σημαντικό χαρακτηριστικό των συγκεκριμένων δικτύων είναι η συνεχής επικοινωνία μεταξύ των κόμβων-αισθητήρων. Ο επεξεργαστής που υπάρχει σε κάθε κόμβο-αισθητήρα δίνει την δυνατότητα αντί να στείλει κατευθείαν τα δεδομένα σε έναν καθορισμένο κόμβο που έχει αναλάβει την συλλογή τους, να εκτελεί καθορισμένους απλούς υπολογισμούς και στην συνέχεια να αποστέλλει μόνο τα απαραίτητα και μερικώς επεξεργασμένα δεδομένα. Η δυνατότητα της τοπικής επεξεργασίας και αποθήκευσης των δεδομένων επιτρέπει στις μονάδες να εκτελέσουν πολύπλοκες λειτουργίες σύμφωνα με την εκάστοτε εφαρμογή ποτ υλοποιούν.

Ο τελικός προορισμός των μετρήσεων στα ασύρματα δίκτυα αισθητήρων είναι ο σταθμός-βάση ο οποίος είναι πάνω σε υπολογιστικά συστήματα τα οποία του παρέχουν μεγάλα ενεργειακά αποθέματα και την δυνατότητα περαιτέρω επεξεργασίας των δεδομένων.

Η προώθηση των πακέτων προτιμάται να γίνεται σταδιακά από κόμβο σε κόμβο και όχι απευθείας ο κάθε κόμβος στον σταθμό βάση διότι εκτός του ότι μπορεί να μην είναι εφικτός ένας τέτοιος τρόπος αποστολής είναι ταυτόχρονα και πιο αποδοτικός ενεργειακά.

2.1.2 Χαρακτηριστικά

Ένα ασύρματο δίκτυο αισθητήρων χαρακτηρίζεται από το χρόνος ζωής του, την επεκτασιμότητα του, την κάλυψη που παρέχει, την ακρίβεια των μετρήσεων αλλά και την ασφάλεια που μπορεί να παρέχει.

Ο χρόνος ζωής του δικτύου είναι από τα σημαντικότερα χαρακτηριστικά και ίσως ο κυριότερος περιοριστικός παράγοντας στην διάρκεια ζωής ενός τέτοιου δικτύου είναι η παροχή ενέργειας του συστήματος. Εδώ πρέπει να επισημάνουμε ότι υπάρχουν εφαρμογές στις οποίες το κρίσιμο χαρακτηριστικό δεν είναι ο μέσος χρόνος ζωής ενός κόμβου αλλά ο ελάχιστος εκτιμώμενος χρόνος ζωής.

Ένας επόμενος σημαντικός παράγοντας είναι η κάλυψη του δικτύου καθώς και η επεκτασιμότητα του. Είναι σημαντικό για τον τελικό χρήστη να μπορεί να αναπτύξει δίκτυα τα οποία καλύπτουν μια ευρεία περιοχή παρατήρησης. Επίσης θα πρέπει να υποστηρίζουν την ανάπτυξη από λίγους κόμβους-αισθητήρες μέχρι και εκατοντάδες ίσως και χιλιάδες ανάλογα την εφαρμογή καθώς και την προσθήκη νέων κόμβων-αισθητήρων χωρίς να διαταράσσεται η λειτουργία του δικτύου.

Ένα επιπλέον χαρακτηριστικό είναι η ακρίβεια των μετρήσεων. Οι μετρήσεις που λαμβάνουν οι αισθητήρες μπορεί να περιέχουν ανακρίβειες λόγω του θορύβου.

Τέλος τα ασύρματα δίκτυα αισθητήρων πρέπει να είναι ικανά να διατηρούν την πληροφορία που συλλέγουν κρυφή σε μη εξουσιοδοτημένους χρήστες. Είναι προφανές οπότε πως το δίκτυο θα πρέπει να παρέχει την δυνατότητα υποστήριξης μηχανισμών κρυπτογράφησης και αυθεντικοποίησης[01].

2.1.3 Προβλήματα και Περιορισμοί

Τα ασύρματα δίκτυα αισθητήρων παρουσιάζουν διαφορετικούς περιορισμούς σε σύγκριση με τους περιορισμούς που συναντιόνται στα παραδοσιακά δίκτυα. Το πιο σημαντικό από αυτά είναι η ενέργεια. Τα ασύρματα δίκτυα αισθητήρων αποτελούνται από συσκευές που πρέπει να είναι ενεργές αρκετή ώρα με μικρές μπαταρίες. Οι κύριες πηγές κατανάλωσης ενέργειας είναι το άνοιγμα/κλείσιμο των αισθητήρων για την μέτρηση των φαινομένων, η μετάδοση των δεδομένων η οποία εξαρτάται από το μέγεθος του πακέτου καθώς και η αναμονή για την λήψη των δεδομένων.

Επίσης υπάρχουν προβλήματα λόγω των περιορισμών που υπάρχουν στους αισθητήρες όπως οι χαμηλές υπολογιστικές δυνατότητες και η μικρή μνήμη. Σε αυτό το τομέα τα τελευταία χρόνια οι δυνατότητες των αισθητήρων έχουν αυξηθεί σημαντικά με αποτέλεσμα να έχουν δημιουργηθεί αλγόριθμοι με μικρές απαιτήσεις μνήμης.

2.2. Εφαρμογές Ασύρματων Δικτύων Αισθητήρων

Τα ασύρματα δίκτυα αισθητήρων μπορούν να συλλέγουν πληροφορίες που αφορούν ένα ευρύ φάσμα περιβαλλοντολογικών συνθηκών όπως και φυσικών φαινομένων όπως [02]:

- Θερμοκρασία
- Υγρασία
- Φώς
- Παρουσία ή Κίνηση αντικειμένου
- Πίεση
- Διάρθρωση εδάφους
- Επίπεδα θορύβου
- Την παρουσία ή απουσία προκαθορισμένων ειδών αντικειμένων
- Γενικά χαρακτηριστικά κάποιου αντικειμένου όπως ταχύτητα, κατεύθυνση, μέγεθος.

Οι κόμβοι-αισθητήρες μπορούν να χρησιμοποιηθούν για συνεχή ανίχνευση, ανίχνευση συμβάντων καθώς και για τον εντοπισμό θέσης. Η ιδέα της μικρό-ανίχνευσης και της ασύρματης σύνδεσης αυτών των κόμβων υπόσχεται πολλές νέες περιοχές εφαρμογών. Οι εφαρμογές των ασύρματων δικτύων αισθητήρων μπορούν να ομαδοποιηθούν σε στρατιωτικές, βιομηχανικές, υγείας, περιβάλλοντος, οικιακές και εμπορικές. Παρακάτω παραθέτονται μερικές από τις εφαρμογές[03].

2.2.1 Περιβαλλοντικές Εφαρμογές

Κάποιες περιβαλλοντικές εφαρμογές περιλαμβάνουν την παρακολούθηση των κινήσεων των πουλιών, ζώων, την παρακολούθηση των περιβαλλοντικών συνθηκών που επηρεάζουν την χλωρίδα και την πανίδα, την ύδρευση, την ακριβή γεωργία, την βιολογική και περιβαλλοντολογική παρακολούθηση της θάλασσας, του εδάφους και του αέρα, την παρακολούθηση για φωτιές στα δάση, την ανίχνευση πλημμυρών κάποια από τα οποία αναλύουμε παρακάτω:

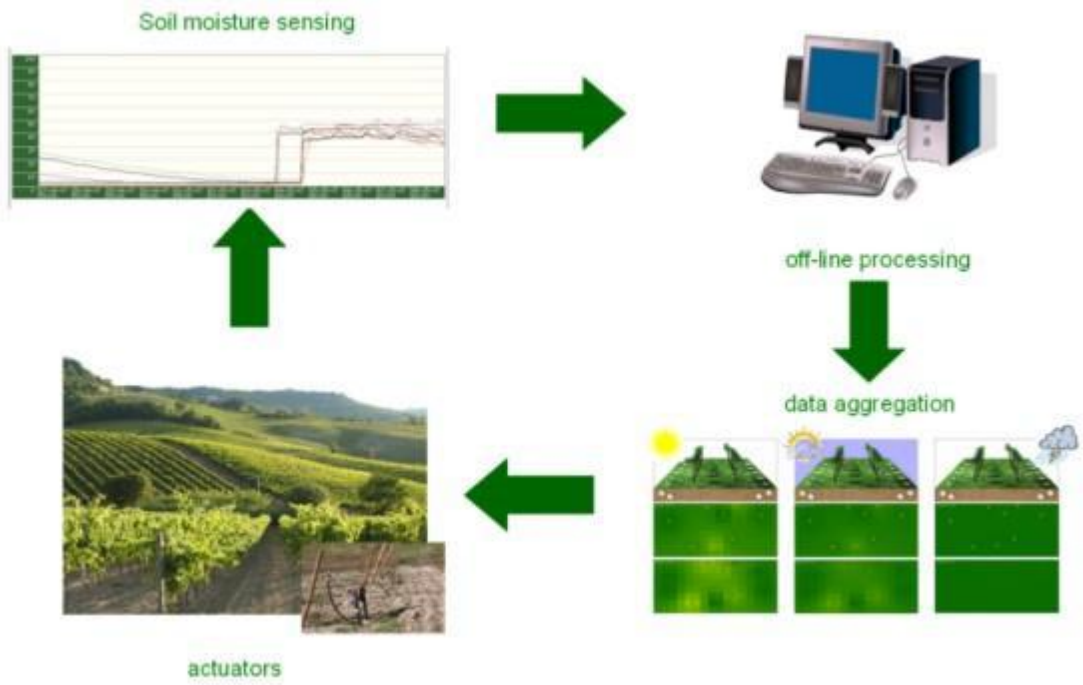
Γεωργία: Στις γεωργικές εφαρμογές η χρήση των ασύρματων δικτύων αισθητήρων συνδέεται με τον εκσυγχρονισμό της παραγωγής και σχετίζεται με την υποστήριξη εφαρμογών ακριβείας για την ορθολογική ρίψη λιπασμάτων, νερού κ.α. στις φυτείες όποτε και όπου είναι απαραίτητο στην αναγκαία ποσότητα.

Εικόνα 2.2 Δίκτυο αισθητήρων για την παρακολούθηση της γεωργικής καλλιέργειας

Μία ενδιαφέρουσα εφαρμογή είναι η ανάπτυξη ενός ασύρματου δικτύου αισθητήρων για την άρδευση καλλιεργειών με στόχο τη δημιουργία ενός ολοκληρωμένου συστήματος υποβοήθησης λήψης αποφάσεων ώστε να συνδράμει προς την κατεύθυνση της αποδοτικότερης χρήσης των

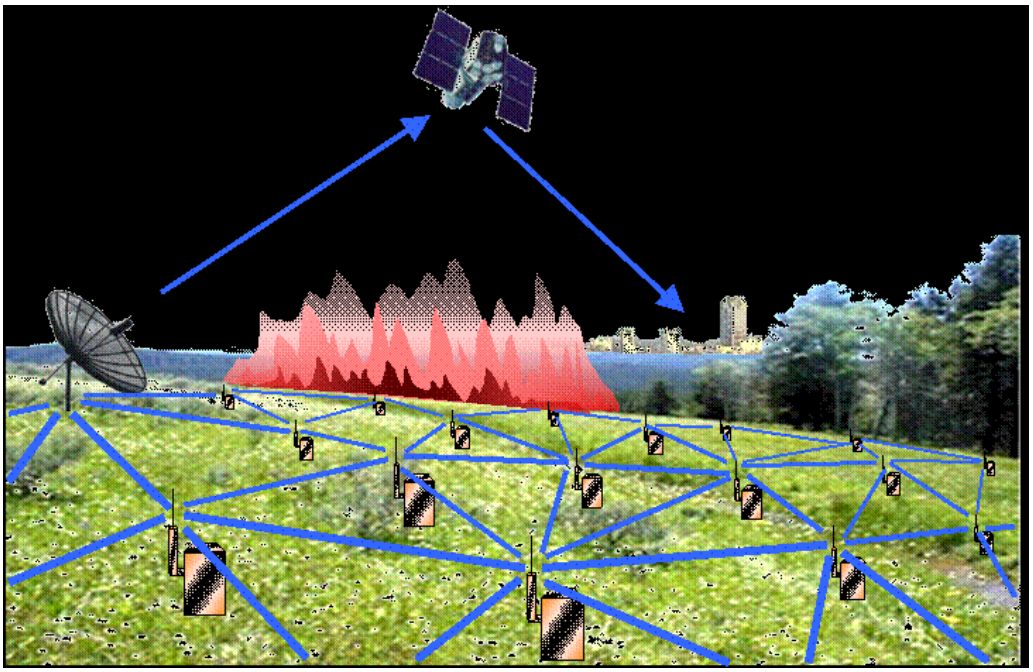
υδάτινων πόρων και προστασίας του περιβάλλοντος καθώς και στην βελτίωση της ποιότητας των καλλιεργειών. Πιο συγκεκριμένα το εν λόγω σύστημα βάσει των μετρήσεων της υγρασίας του εδάφους και της θερμοκρασίας της ατμόσφαιρας που λαμβάνει από το εγκατεστημένο στον αγρό ασύρματο δίκτυο αισθητήρων όπως και της μετεωρολογικής πρόβλεψης και των χαρακτηριστικών του αγρού (όπως είναι η καλλιέργεια, ο τύπος εδάφους κ.α.) επιτρέπει στον καλλιεργητή να γνωρίζει την ποσότητα νερού που απαιτείται για τη βέλτιστη άρδευση του ανά περιοχή του αγρού.

Στο συγκεκριμένο σύστημα πέτυχε διεθνή διάκριση το Πολυτεχνείο Κρήτης το 2011 για την ανάπτυξη μιας πρωτοποριακής τεχνολογίας για τα ασύρματα δίκτυα αισθητήρων, πολύ χαμηλού κόστους, που επιτρέπει τη ρύθμιση και τον έλεγχο της κατανάλωσης νερού. Η μέθοδος αυτή είναι ιδανική για το αυτόματο πότισμα φυτών σε αγροτικές καλλιέργειες καθώς επιτυγχάνεται σε μεγάλο βαθμό εξοικονόμηση ενέργειας μέσω της τεχνικής της ανάκλασης σημάτων από τον πομπό που είναι πάνω στο φυτό στον δέκτη που είναι στον υπολογιστή μέσω των κόμβων-αισθητήρων για τις ποσότητες νερού που χρειάζονται για την άρδευση.[03]



Εικόνα 2.2 Εγκατάσταση αισθητήρα για τον έλεγχο της υγρασίας του εδάφους.

Ανίχνευση δασικών πυρκαγιών: Καθώς οι κόμβοι-αισθητήρια μπορούν να εγκατασταθούν είτε στρατηγικά είτε τυχαία σε ένα δάσος μπορούν να αναμεταδώσουν την ακριβή προέλευση της φωτιάς στους άμεσα ενδιαφερόμενους προτού η πυρκαγιά να προλάβει να εξαπλωθεί ανεξέλεγκτα.



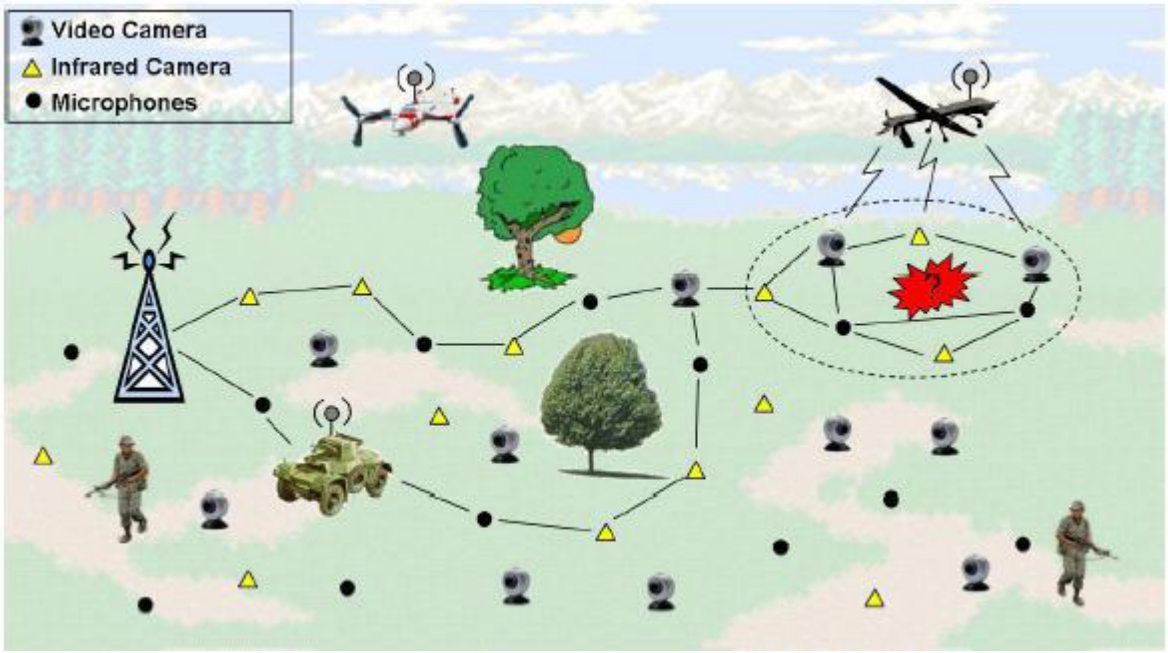
Εικόνα 2.2 Δίκτυο αισθητήρων στη δασική πυρασφάλεια

2.2.2 Στρατιωτικές Εφαρμογές

Τα ασύρματα δίκτυα αισθητήρων μπορούν να αποτελέσουν ένα ενσωματωμένο κομμάτι των στρατιωτικών συστημάτων, διαταγών, ελέγχου, επικοινωνιών, υπολογισμού, παρακολούθησης και στόχευσης. Καθώς τα δίκτυα αισθητήρων βασίζονται στην πυκνή χωρική εγκατάσταση, η καταστροφή μερικών κόμβων από εχθρικές δυνάμεις δεν επηρεάζει μια στρατιωτική επιχείρηση σε τέτοιο βαθμό όσο η καταστροφή των παραδοσιακών αισθητήρων, κάνοντας την χρήση των δικτύων αισθητήρων ιδανική για τα πεδία των μαχών.

Μπορεί να γίνει παρακολούθηση του εξοπλισμού και των πυρομαχικών των στρατιωτικών δυνάμεων. Κάθε στρατιώτης, όχημα, εξοπλισμός μπορεί να εξοπλιστεί με αισθητήρες που θα αναφέρουν την κατάστασή του.

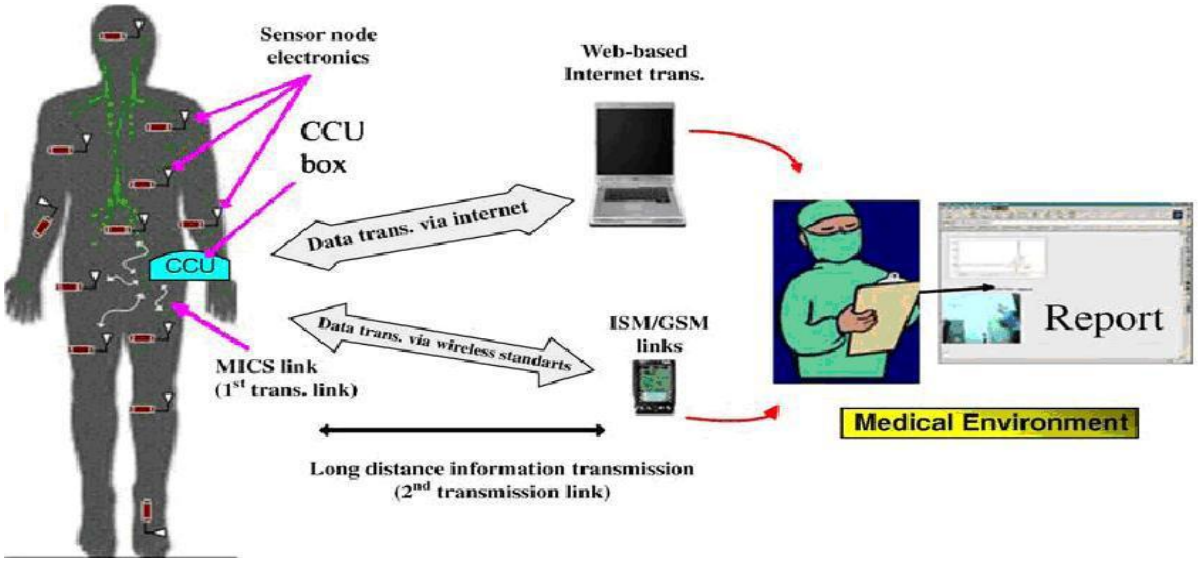
Επίσης μπορεί να γίνει παρακολούθηση του πεδίου της μάχης. Μονοπάτια και δρομολόγια προσέγγισης μπορούν γρήγορα να καλυφθούν με δίκτυα αισθητήρων και να παρακολουθούνται για εχθρικές δραστηριότητες[04].



Εικόνα 2.3 Αναπαράσταση στρατιωτικών εφαρμογών

2.2.3 Εφαρμογές Υγείας

Μερικές από τις εφαρμογές των ασύρματων δικτύων αισθητήρων είναι η παροχή μέσων αλληλεπίδρασης για άτομα με ειδικές ανάγκες, η παρακολούθηση ασθενών, η διάγνωση, η διαχείριση των φαρμάκων σε νοσοκομεία. Ο εφοδιασμός κάθε ασθενή σε ένα νοσοκομείο με μικρούς φορητούς αισθητήρες για καταγραφή ζωτικών παραμέτρων θα μπορούσε να δώσει τη δυνατότητα σε ιατρούς και νοσηλευτικό προσωπικό να παρακολουθούν συνεχώς την κατάσταση του ασθενή από μακριά και να επεμβαίνουν εγκαίρως σε περίπτωση που χρειάζεται. Τέτοια συστήματα που είδη χρησιμοποιούνται είναι το ηλεκτροκαρδιογράφημα, το ηλεκτρομυογράφημα, η θερμοκρασία του ασθενούς, η αναπνοή, η πίεση κ.α. Η παρακολούθηση μέσω ασύρματων αισθητήρων των καθημερινών δραστηριοτήτων του ανθρώπου όπως ο τρόπος οδήγησης, ο χρόνος ντυσίματος και μαγειρέματος παρόλο που εκ πρώτης όψεως φαντάζει αχρείαστο σε βάθος χρόνου για το κάθε άτομο ξεχωριστά μπορεί να οδηγήσει στη διάγνωση νευρικών διαταραχών όπως το Parkinson ή το Alzheimer[06].



Εικόνα 2.4 Σύστημα αισθητήρων για την παρακολούθηση ασθενών

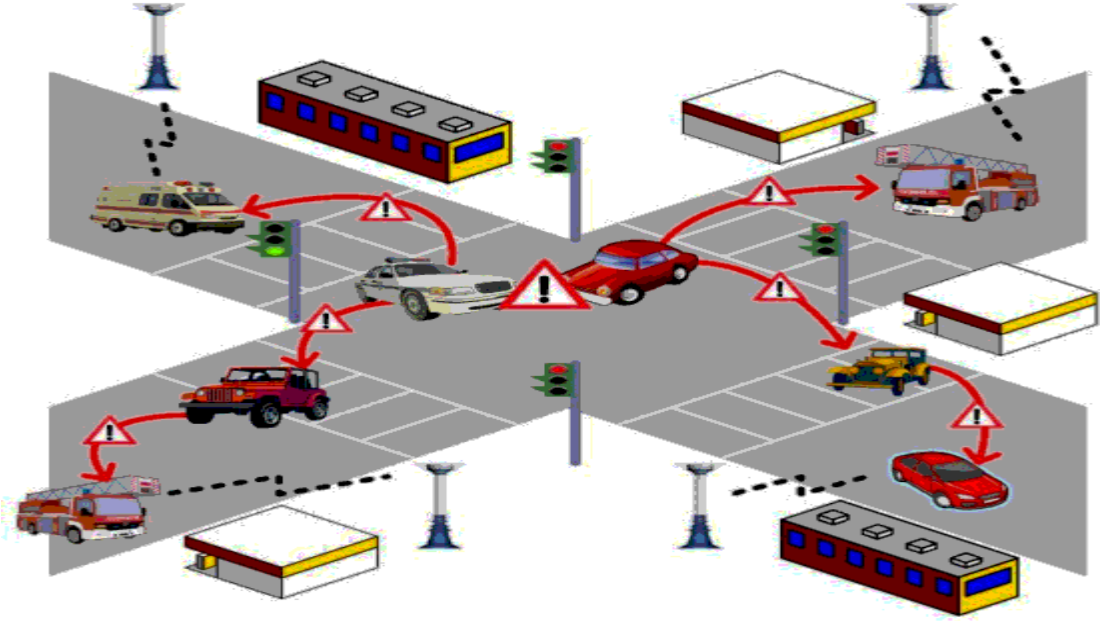
2.2.4 Οικιακές Εφαρμογές

Τα ασύρματα δίκτυα αισθητήρων μπορούν να παρέχουν στους κατοίκους ενός σπιτιού εξοικονόμηση ενέργειας, έλεγχο συσκευών καθώς και αυτοματοποιημένες διαδικασίες. Δύνεται η δυνατότητα αυτόματης ενεργοποίησης και απενεργοποίησης του φωτισμού, της αυτόματης ρύθμισης της θερμοκρασίας ανάλογα με τις εξωτερικές κλιματολογικές συνθήκες, έλεγχος όλων των ηλεκτρικών συσκευών με το πάτημα ενός κουμπιού καθώς και σε συστήματα ασφάλειας σε περίπτωση ανίχνευσης κίνησης ή ανοίγματος παραθύρου ή πόρτας[08].

Εικόνα 2.5 Σύστημα ελέγχου σπιτιού

2.2.5 Εφαρμογές για τον έλεγχο της κίνησης

Τα ασύρματα δίκτυα αισθητήρων μπορούν να χρησιμοποιηθούν για την παρακολούθηση και τον έλεγχο της οδικής κυκλοφορίας των οχημάτων. Μπορούμε να ελέγξουμε τα φανάρια για πιο αποδοτική χρήση τους καθώς και με βιντεοκάμερες να παρακολουθούμε τμήματα οδών με έντονη συμφόρηση ή τμήματα οδών με υψηλά ποσοστά τροχαίων ατυχημάτων.



Εικόνα 2.6 Παρακολούθηση κίνησης οχημάτων

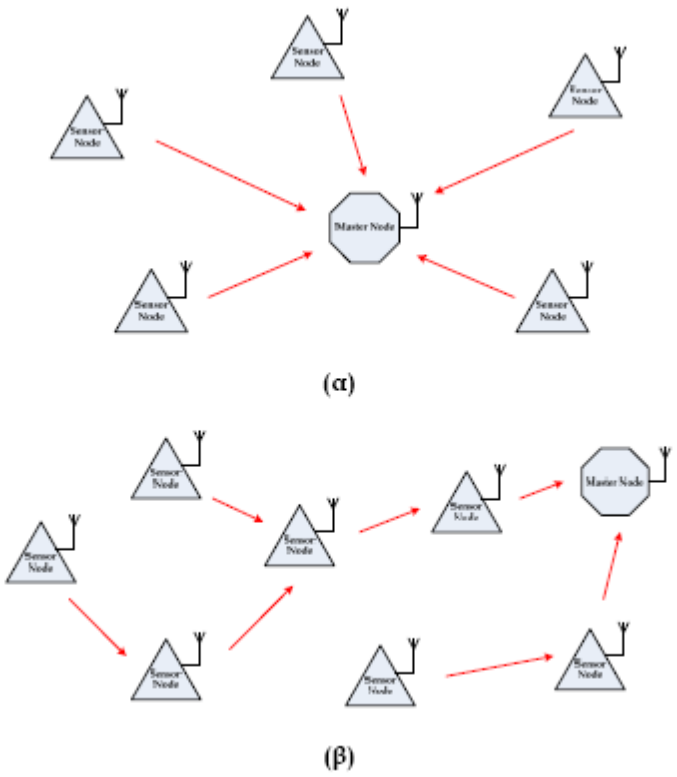
2.2.6 Βιομηχανικές Εφαρμογές

Ο έλεγχος των συστημάτων και των εφαρμογών στις βιομηχανίες παίζει σημαντικό ρόλο στην ορθή λειτουργία της παραγωγής, στην μείωση του κόστους της παραγωγής, της βελτίωσης των μηχανημάτων καθώς και της απόδοσης του χρήστη. Οι αισθητήρες μπορούν να χρησιμοποιηθούν σε μέρη που μπορεί να είναι επικίνδυνα για τον άνθρωπο εξαιτίας π.χ. της θερμοκρασίας, της τοξικότητας ή σε μέρη που δεν μπορεί να παραβρεθεί όπως το εσωτερικό των μηχανών.

2.3 Μοντέλα επικοινωνίας στα Ασύρματα Δίκτυα Αισθητήρων

2.3.1 Τρόποι μετάδοσης πληροφορίας

Όπως όλα τα δίκτυα έτσι και τα ασύρματα δίκτυα αισθητήρων πρέπει να μπορούν να επικοινωνούν με κάποιον κεντρικό κόμβο στον οποίο θα μεταφέρουν τις πληροφορίες ή ακόμα και μεταξύ τους . Στα ασύρματα δίκτυα αισθητήρων μπορούμε να έχουμε δύο(2) τρόπους μετάδοσης της πληροφορίας: Α) Με ένα βήμα την φορά (Single-hop) και Β) με πολλαπλά βήματα (multi-hop). Στην εικόνα 2.7 δείχνουμε γραφικά τους τρείς τρόπους και αμέσως μετά κάνουμε μια αναφορά σε αυτούς[07].



Εικόνα 2.7 α) Επικοινωνία με ένα βήμα (single-hop) β) Επικοινωνία με πολλαπλά βήματα (multi-hop)

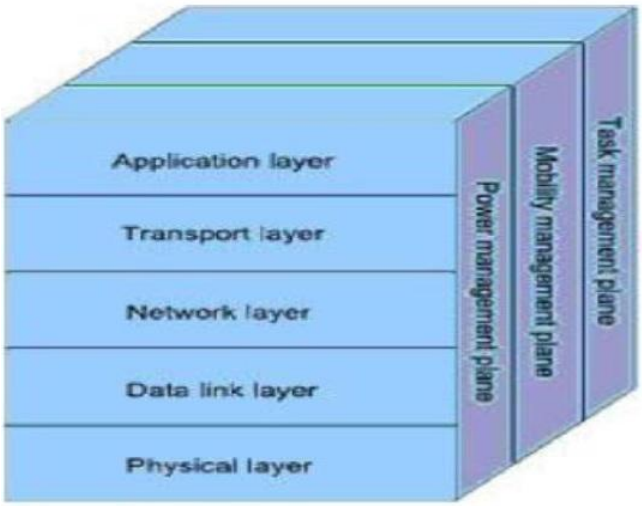
Στην περίπτωση που εφαρμόζεται η επικοινωνία single-hop όλοι οι κόμβοι-αισθητήρες πρέπει να είναι εντός της εμβέλειας του σταθμού-βάσης. Για να μπορέσουμε να πετύχουμε την μέγιστη δυνατή εμβέλεια θα πρέπει η εκπομπή να γίνει με την μέγιστη δυνατή ισχύ. Τα προβλήματα σε αυτή τη περίπτωση είναι σημαντικά. Καταρχήν έχουμε μεγάλο περιορισμό στην εμβέλεια του

δικτύου μας. Έχει αποδειχθεί ότι με αυτό τον τρόπο η ακτίνα επικοινωνίας είναι αρκετά μικρή. Επίσης η επεκτασιμότητα του δικτύου μας ορίζεται από την μέγιστη ακτίνα επικοινωνίας. Ένας άλλος παράγοντας που μας δημιουργεί πρόβλημα είναι η διάρκεια ζωής της ενέργειας που χρησιμοποιούμε καθώς όσο πιο μεγάλη ισχύ χρειαζόμαστε για να έχουμε το μέγιστο της εμβέλειας τόσο περισσότερη ενέργεια καταναλώνουμε. Από τα παραπάνω διαπιστώνουμε ότι η single-hop επικοινωνία είναι για ιδιαίτερες εφαρμογές και θα πρέπει να ληφθούν υπόψη όλοι οι παράγοντες.

Στην περίπτωση που εφαρμόζεται η επικοινωνία multi-hop δεν χρειάζεται ο κάθε κόμβος-αισθητήρας να είναι εντός της εμβέλειας του σταθμού-βάσης. Ο κάθε κόμβος-αισθητήρας στέλνει την πληροφορία σε κάποιον άλλο κόμβο-αισθητήρα και εκείνος σε κάποιον άλλο κόμβο-αισθητήρα μέχρι η πληροφορία να φτάσει στον σταθμό-βάσης. Με αυτό τον τρόπο και χρησιμοποιώντας ένα μεγάλο αριθμό από κόμβους-αισθητήρες μπορούμε να έχουμε καταρχήν μεγάλη ακτίνα κάλυψης του δικτύου μας. Επίσης μπορούμε να επεκτείνουμε την κάλυψη του δικτύου απλά και μόνο προσθέτοντας καινούριους κόμβους-αισθητήρες. Ένας άλλος σημαντικός παράγοντας για την χρήση της συγκεκριμένης επικοινωνίας είναι ότι επειδή ο κάθε κόμβος-αισθητήρας δεν απέχει μεγάλη απόσταση από κάποιον άλλο κόμβο-αισθητήρα δεν χρειάζεται να έχει μεγάλη ισχύ το σήμα για την μετάδοση της πληροφορίας άρα δεν υπάρχει και μεγάλη κατανάλωση ενέργειας. Φυσικά και αυτός ο τρόπος έχει τα αρνητικά του καθότι χρειάζονται περίπλοκοι αλγόριθμοι δρομολόγησης καθώς και μεγάλος αριθμός κόμβων-αισθητήρων τα οποία αποτελούν ένα σημαντικό κόστος[07].

2.4 Αρχιτεκτονική στα Ασύρματα Δίκτυα Αισθητήρων

Η αρχιτεκτονική του δικτύου των ασύρματων αισθητήρων απαιτεί την ύπαρξη, έστω και σε απλουστευμένη μορφή, ενός φυσικού επιπέδου (Physical Layer), ενός υπό-επιπέδου MAC και ενός επιπέδου ζεύξης (Link Layer) πρωτοκόλλων.



Εικόνα 2.8 Τα Layers στα Ασύρματα Δίκτυα Αισθητήρων

2.4.1 Το Φυσικό επίπεδο (Physical Layer)

Το φυσικό επίπεδο είναι υπεύθυνο για την επιλογή της συχνότητας, την παραγωγή συχνότητας μετάδοσης, την ανίχνευση σήματος, τη διαμόρφωσή του και για την κρυπτογράφηση των δεδομένων. Η διαμόρφωση και αποδιαμόρφωση του ψηφιακού σήματος εκτελείται από τον πομποδέκτη και δεν πρέπει να είναι ιδιαίτερα πολύπλοκη προκειμένου να υλοποιείται από hardware χαμηλού κόστους και κατανάλωσης με όσο το δυνατόν πιο αξιόπιστο αποτέλεσμα. Για τη μετάδοση των σημάτων, ως επί το πλείστον χρησιμοποιούνται τεχνικές εξάπλωσης φάσματος, οι οποίες ονομάζονται έτσι επειδή το βασικό στοιχείο της λειτουργίας τους έγκειται στο γεγονός ότι οι εκπεμπόμενες κυματομορφές καταλαμβάνουν μεγαλύτερο εύρος ζώνης (bandwidth) από ότι πραγματικά χρειάζεται για τη μετάδοση των δεδομένων. Οι τεχνικές αυτές χρησιμοποιούνται για διάφορους λόγους και ο βασικότερος θεωρείται ότι είναι η μείωση των παρεμβολών από άλλα σήματα, καθώς το σήμα δεν μεταδίδεται σε μια μόνο συχνότητα. Άλλοι λόγοι για τους οποίους χρησιμοποιήθηκαν κατά καιρούς τέτοιες τεχνικές είναι η ασφάλεια από υποκλοπές του σήματος, η αντίσταση στην εξασθένηση του σήματος, καθώς και η δυνατότητα χρήσης του μέσου από πολλές συσκευές ταυτόχρονα (multiple access).

2.4.2 Το Επίπεδο Ζεύξης Δεδομένων (Data Link Layer)

Το πρωτόκολλο του επιπέδου ζεύξης χρησιμοποιείται προκειμένου να μεταφέρει ένα δεδομενόγραμμα ή πακέτο επάνω σε μία ζεύξη. Ορίζει τη μορφή των πακέτων που ανταλλάσσονται ανάμεσα στους κόμβους, στα άκρα της ζεύξης, καθώς και τις ενέργειες που γίνονται από αυτούς τους κόμβους όταν στέλνουν και λαμβάνουν αυτά τα πακέτα. Οι μονάδες

δεδομένων που ανταλλάσσονται από ένα πρωτόκολλο επιπέδου ζεύξης ονομάζονται πλαίσια (frames) και συνεπώς οι ενέργειες που γίνονται από αυτά τα πρωτόκολλα κατά την αποστολή και λήψη τους είναι η ανίχνευση σφάλματος, η αναμετάδοση, ο έλεγχος ροής και η τυχαία πρόσβαση. Το πρωτόκολλο αυτό σε ένα ασύρματο multi-hop και αυτό-οργανούμενο 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 γνωστές μέθοδοι για τον έλεγχο σφαλμάτων (error control) είναι η εμπροσθόδοτη (χωρίς επαλήθευση) διόρθωση σφάλματος (Forward Error Correction-FEC) και ο έλεγχος σφαλμάτων με αυτόματη αίτηση επανάληψης (Automatic Repeat Request-ARQ). Η χρησιμότητα του ARQ σε multi-hop 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 Το Επίπεδο Δικτύου (Network Layer)

Οι τεχνικές δρομολόγησης των «παραδοσιακών» ad-hoc συνήθως δεν ταιριάζουν στις απαιτήσεις των ασύρματων δικτύων αισθητήρων. Μια από τις αρμοδιότητες του network layer είναι να παρέχει διασύνδεση με εξωτερικά δίκτυα (πχ. άλλα δίκτυα αισθητήρων ή το διαδίκτυο). Το επίπεδο αυτό των δικτύων αισθητήρων σχεδιάζεται με ορισμένες αρχές:

- Πρέπει να διαχειρίζεται αποδοτικά τα διαθέσιμα ποσά ενέργειας τους.
- Πρέπει να είναι δεδομενο-κεντρικό (data-centric).
- Η συνάθροιση των δεδομένων είναι χρήσιμη μόνο όταν δεν εμποδίζεται η συνεργασία μεταξύ των κόμβων αισθητήρων.
- Σε ένα ιδανικό δίκτυο αισθητήρων, οι διευθύνσεις των κόμβων είναι βασισμένες στην ιδιότητα (attribute-based) με αποτέλεσμα οι κόμβοι να μη γνωρίζουν την ακριβή τοποθεσία τους.

Τα energy–efficient δρομολόγια αναζητούνται είτε με βάση τη διαθέσιμη εναπομένουσα ενέργεια σε κάθε κόμβο (available power - PA) είτε με την ενέργεια 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:

- 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.

- Minimum energy (ME) route, επιλέγεται το δρομολόγιο το οποίο απαιτεί την ελάχιστη ενέργεια για τη μεταφορά των πακέτων μεταξύ των κόμβων.

- Minimum hop route (MH), επιλέγεται το δρομολόγιο το οποίο θα έχει τον ελάχιστο αριθμό hops μέχρι το κόμβο που θα λάβει τα δεδομένα.

- 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.

2.4.4 Το Επίπεδο Μεταφοράς (Transport Layer)

Το επίπεδο αυτό 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.

2.4.5 Το Επίπεδο Εφαρμογών (Application Layer)

Αν και έχουν ερευνηθεί πολλές περιοχές εφαρμογών για ασύρματα δίκτυα αισθητήρων, δεν συμβαίνει το ίδιο με την έρευνα των πρωτοκόλλων για το επίπεδο εφαρμογής, με αποτέλεσμα ο

αριθμός αυτών των πρωτοκόλλων να είναι περιορισμένος. Μερικά από τα πρωτόκολλα αυτού του επιπέδου που ερευνούνται είναι:

- Sensor Management Protocol (SMP)
- Task Assignment and Data advertisement Protocol (TADAP)
- Sensor Query and Data Dissemination Protocol (SQDDP)

2.5 Τεχνικές προκλήσεις και απαιτήσεις των Ασύρματων Δικτύων Αισθητήρων

2.5.1 Τεχνικές προκλήσεις των Ασύρματων Δικτύων Αισθητήρων

Ο σχεδιασμός των WSN στηρίζεται σε μια ή περισσότερες από τις παρακάτω τεχνικές προκλήσεις [09],[10]:

- 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.

2.5.2 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, αναφέρονται παρακάτω[09],[10],[11],[12],[13]:

- 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 προς 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. Ο Πίνακας 2.1 συνοψίζει τις τεχνικές προκλήσεις και απαιτήσεις των WSN και τους αντίστοιχους στόχους σχεδιασμού.

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

Πίνακας 2.1 Σύνοψη των τεχνικών προοπτικών και απαιτήσεων και των στόχων σχεδιασμού στα Ασύρματα Δίκτυα Αισθητήρων.

Κεφάλαιο 3
Το Υλικό (Hardware)

Το hardware που χρησιμοποιήθηκε στην παρούσα εργασία περιλαμβάνει τρείς ασύρματους κόμβους τύπου micaz, που σχεδιάζονται και κατασκευάζονται από την εταιρία Crossbow :

- μία πλακέτα αισθητήρων MDA 100CA,
- μία πλακέτα αισθητήρων MDA 300CA,
- μία πλακέτα προγραμματισμού των αισθητήρων MIB520

Επίσης χρησιμοποιήθηκε ένας εξωτερικός αισθητήρας υγρασίας εδάφους ECH2O EC-.

3.1 Η πλατφόρμα micaz

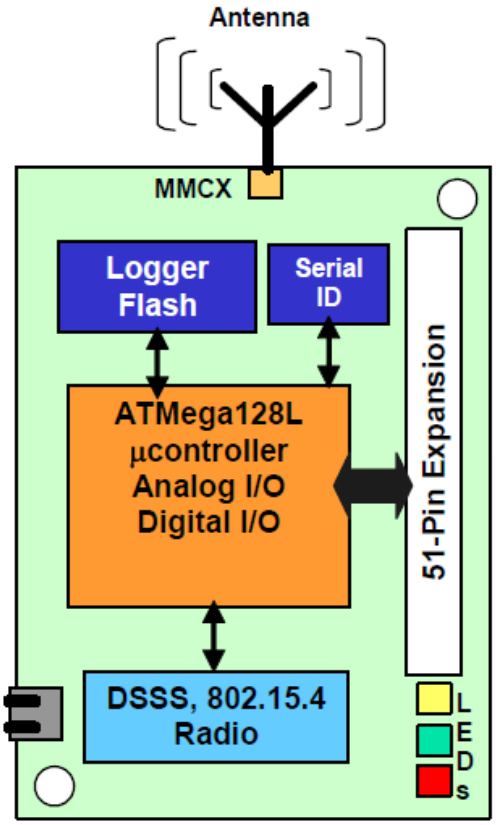
3.1.1 Τεχνικά χαρακτηριστικά του micaz

Το micaz αποτελεί μία από τις τελευταίες γενιές ασύρματων αισθητήρων που αναπτύχθηκαν για πρώτη φορά στο πανεπιστήμιο του Berkeley στην Καλιφόρνια και τώρα παράγεται από την εταιρία Crossbow. Στο εσωτερικό του φιλοξενεί τον μικροελεγκτή Atmega128L της Atmel, είναι συμβατό με το πρωτόκολλο IEEE 802.15.4, εκπέμπει στην μπάντα συχνοτήτων των 2.4GHz και είναι κατάλληλο για χαμηλής κατανάλωσης ασύρματα δίκτυα. Το micaz διαθέτει κάποια νέα χαρακτηριστικά τα οποία εκμεταλλεύονται στο έπακρο τη λειτουργικότητα της οικογένειας ασύρματων δικτυακών προϊόντων MICA της Crossbow. Τα χαρακτηριστικά αυτά περιλαμβάνουν[16]:

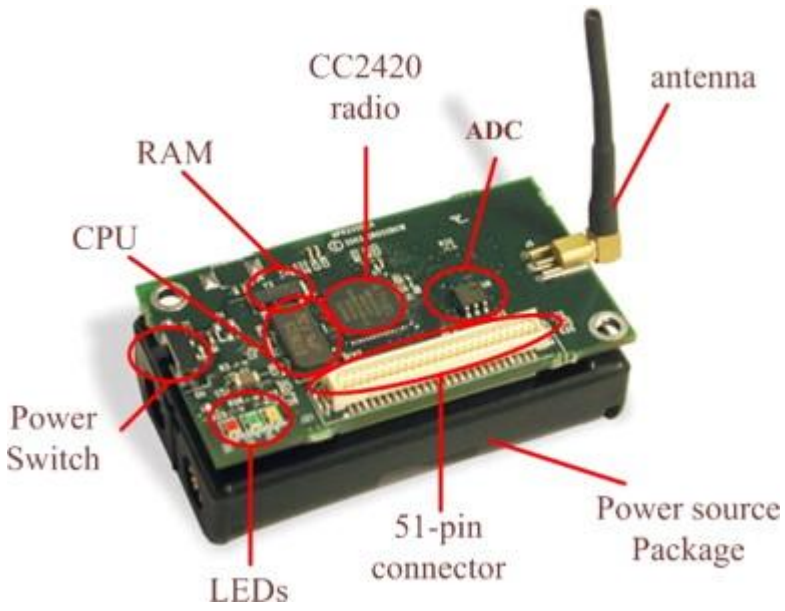
- RF πομποδέκτης συμβατός με το πρωτόκολλο IEEE 802.15.4/ZigBee.
- Παγκόσμια ζώνη εκπομπής 2.4-2.483GHz.
- Τεχνολογία DSSS – Direct Sequence Spread radio Spectrum (Απευθείας διάδοση της ακολουθίας του ραδιοφάσματος) το οποίο είναι ανθεκτικό στην RF παρεμβολή και παρέχει ασφάλεια στα δεδομένα.
- 250 kbps data rate (ρυθμός μετάδοσης δεδομένων).
- Λειτουργικό σύστημα TinyOS 1.1.7 και νεότερες εκδόσεις, περιλαμβάνοντας και το λογισμικό της Crossbow για τη δημιουργία mesh δικτύου.
- Plug and play λειτουργία για όλες τις πλακέτες αισθητήρων της Crossbow, πλακέτες συλλογής δεδομένων (data acquisition boards), gateways και λογισμικό.

Στην εικόνα 3.1 που ακολουθεί αποτυπώνεται γραφικά το εσωτερικό του micaz, όπου διακρίνονται ο μικροελεγκτής, ο RF πομποδέκτης, η logger flash(την οποία μπορούμε να χρησιμοποιήσουμε για over-the-air προγραμματισμό ή/και για αποθήκευση δεδομένων) και η 51-pin υποδοχή η οποία υποστηρίζει αναλογικές εισόδους, ψηφιακές εισόδους/εξόδους, διασυνδέσεις I2C, UART και SPI τα οποία κάνουν εύκολη τη σύνδεση με άλλα περιφερειακά ενώ στην εικόνα 3.2 φαίνεται το εξωτερικό σχήμα του micaz. Οι αισθητήρες που μπορεί να

φιλοξενήσει είναι θερμοκρασίας, βαρομετρικής πίεσης, φωτός, επιταχυνσιόμετρου δύο αξόνων και άλλες πλακέτες της Crossbow.



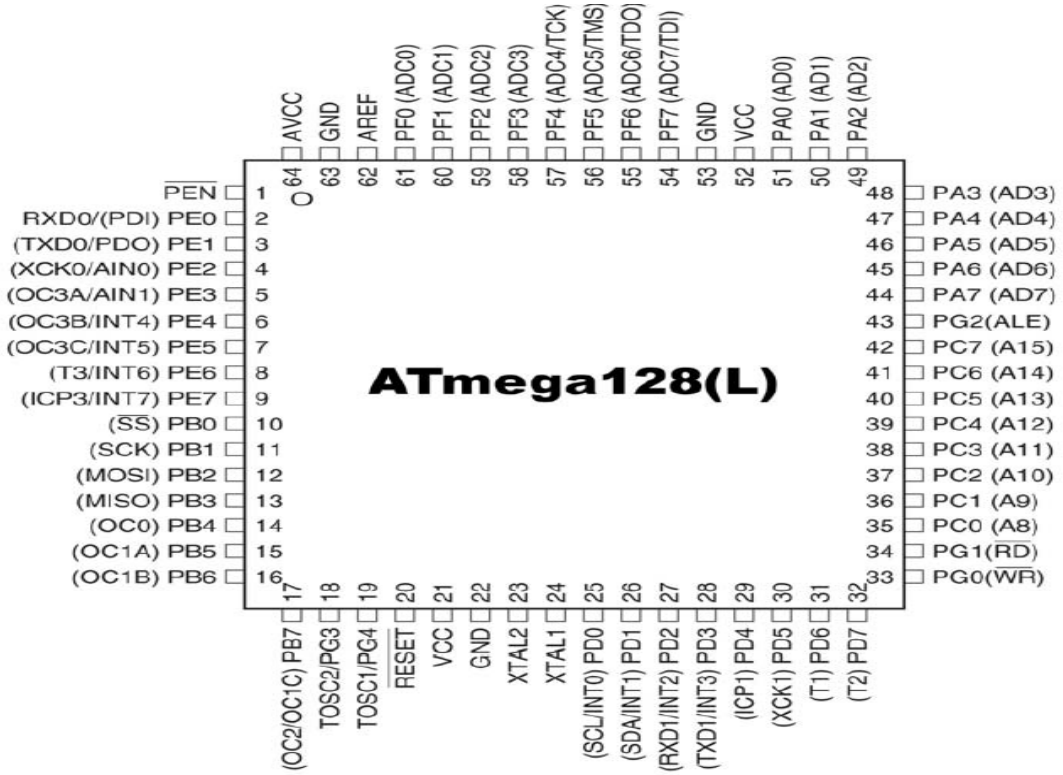
Εικόνα 3.1 Το εσωτερικό του micaz



Εικόνα 3.2 Το εξωτερικό σχήμα του micaz

3.1.2 Ο μικροελεγκτής ATmega128L

Ο μικροελεγκτής που φιλοξενείται στο εσωτερικό του micaz είναι ο 8-bit AVR ATmega128L[17] της Atmel, ο οποίος είναι υψηλής επίδοσης και χαμηλής κατανάλωσης. Έχει σχεδιαστεί με RISC αρχιτεκτονική, έχει συχνότητα λειτουργίας τα 8MHz, αποτελείται από 133 ισχυρές εντολές που οι περισσότερες από αυτές εκτελούνται κατά τη διάρκεια ενός κύκλου και από 32 8-bit καταχωρητές γενικού σκοπού μαζί με κάποιους καταχωρητές ελέγχου των περιφερειακών.



Εικόνα 3.3 Διάγραμμα ακροδεκτών του ATmega128L

Ο ATmega128L παρέχει 53 προγραμματιζόμενες γραμμές μέσω των οποίων ο μικροελεγκτής πραγματοποιεί την επικοινωνία με τα περιφερειακά του. Οι 53 αυτές γραμμές μπορούν να χρησιμοποιηθούν με εντελώς γενικό τρόπο, δηλαδή μπορούν να λειτουργούν είτε σαν γραμμές εισόδου είτε σαν γραμμές εξόδου. Ωστόσο, κάθε γραμμή συνδέεται on-chip με κάποια λειτουργία, όπως ο ADC, η σειριακή θύρα UART, έναν hardware timer ή μία εξωτερική διακοπή. Έγκειται πλέον στον χρήστη, είτε από την πλευρά του λογισμικού TinyOS είτε από την πλευρά μιας

εφαρμογής, να επιλέξει σωστά τον τρόπο που θα λειτουργήσει κάθε γραμμή, για παράδειγμα σαν γενική γραμμή εξόδου, σαν μέρος μιας ADC μετατροπής ή να ελέγχει ένα σύνολο ψηφιακών αισθητήρων μέσω της διασύνδεσης I2C .

Από το διάγραμμα ακροδεκτών του μικροελεγκτή βλέπουμε ότι οι διαθέσιμες γραμμές εισόδου-εξόδου είναι οι PA ως PG, όπου οι γραμμές A ως F έχουν 8 pins η κάθε μία, ενώ η γραμμή G έχει μόνο 5. Κάθε γραμμή αντιπροσωπεύεται από 3 καταχωρητές, οι οποίοι στο σύνολό τους, για κάθε pin κάθε γραμμής, παρέχουν πρόσβαση στη λειτουργία των γραμμών.

Οι καταχωρητές αυτοί είναι:

- Data Direcion Register (DDRx, όπου το x αντιστοιχεί σε A ως G), ο οποίος καθορίζει πότε ένα pin θα χρησιμοποιείται ως είσοδος ή έξοδος.

- Data Register και ο Port Input Pin register οι οποίοι παρέχουν πρόσβαση στην έξοδο και είσοδο του κάθε pin αντίστοιχα.

Επιπλέον, πάνω από 100 καταχωρητές μπορούν να χρησιμοποιηθούν ώστε να ενεργοποιούν την λειτουργία με την οποία συνδέεται το κάθε pin. Καθένας από αυτούς τους καταχωρητές παρέχεται στον προγραμματιστή σαν hardware καταχωρητής. Παρόλο που ο όρος καταχωρητής είναι γνωστός σαν καταχωρητής επεξεργασίας, δηλαδή μια μικρή ποσότητα on-chip μνήμης που χρησιμεύει στο να αποθηκεύει ενδιάμεσες τιμές κατά τη διάρκεια υπολογισμών, οι hardware καταχωρητές είναι πολύ πιο συνήθεις σε ενσωματωμένα συστήματα και ενώ συχνά μοιάζουν σαν άλλο ένα μέσο αποθήκευσης, έχουν τη δυνατότητα να ελέγχουν φυσικά την πρόσβαση σε διάφορες συσκευές.

Τα κύρια χαρακτηριστικά του ATmega128L είναι:

- 128Kbytes In-System προγραμματιζόμενη flash με δυνατότητες Read και Write

  Αποθηκεύει τον κώδικα της εφαρμογής και προγραμματίζεται μέσω του board MIB520 ή μέσω OTAP (Over the air programming). Όταν επαναπρογραμματίζεται, διαγράφεται όλη η μνήμη εκτός από το κομμάτι που περιέχει τον bootloader.

- 4Kbytes EEPROM

Η EEPROM χρησιμεύει για την αποθήκευση μόνιμων τιμών, όπως το group_id και το radio channel. Οι τιμές αυτές αποτελούν μέρος του ενεργού μηνύματος που ανταλλάσσουν οι αισθητήρες, περιέχονται στην επικεφαλίδα TOS_Header και στη δομή του ενεργού μηνύματος TOS_Msg, το οποίο θα αναλύσουμε μετέπειτα.

- 4Kbytes SRAM

Χρησιμοποιείται για αποθήκευση παραμέτρων της εφαρμογής, όπως μεταβλητές του TinyOS και των XSensor εφαρμογών. Περιέχει επίσης και τη στοίβα.

- Δύο 8-bit Timers/Counters και δύο 16-bit Timers/Counters

Οι δυο 16-bits Timers/Counters είναι ελεύθερα προσβάσιμοι για τον χρήστη, σε αντίθεση με τους 8-bits Timers/Counters που χρησιμοποιούνται αποκλειστικά από το TinyOS.

- SPI – Bus

Δεσμεύεται για τη σύνδεση επεξεργαστή – RF module και δεν είναι διαθέσιμη στον χρήστη. Χρησιμοποιείται από τον χρήστη μόνο στην περίπτωση επαναπρογραμματισμού.

- I2C Bus.

Αποτελεί καθιερωμένο σειριακό πρότυπο επικοινωνίας σε πολλούς αισθητήρες.

- 8-channel,10-bit ADC.

Έχει τάση αναφοράς την τάση της μπαταρίας.

- Δύο UARTS.

Έχουν δυνατότητα σύγχρονης και ασύγχρονης λειτουργίας. Η UART0 δεσμεύεται για την επικοινωνία με το mote που παίζει το ρόλο του base station μέσω του MIB520, ενώ η UART1 είναι διαθέσιμη για τον χρήστη. Τα control pins της UART1 είναι κοινά με αυτά της flash.

- GPIO-General Purpose I/O lines.

Γραμμές εισόδου-εξόδου γενικού σκοπού.

- Εξωτερικό Ρολόι Υψηλής ταχύτητας (External Clock High Speed) 7,3728 MHz.

Χρησιμεύει για την παραγωγή σταθερών baudrates της UART όταν ο κόμβος είναι συνδεδεμένος με το MIB520. Σε κάθε άλλη περίπτωση χρησιμοποιείται το εσωτερικό ρολόι των 8 MHz που μειώνει την ολική κατανάλωση ενέργειας. Το ρολόι υψηλής ταχύτητας απενεργοποιείται όταν το mote βρίσκεται σε κατάσταση sleep.

- Εξωτερικό Ρολόι Χαμηλής ταχύτητας (External Clock Low Speed) 32 kHz

Χρησιμεύει στον χρονισμό του TinyOS. Λειτουργεί συνεχώς έτσι ώστε να μπορει να "ξυπνάει" τον κόμβο από κατάσταση SLEEP.

- JTAG test interface συμβατό με το πρότυπο IEEE std. 1149.1

Χρησιμεύει για αποσφαλμάτωση on-chip και για τον προγραμματισμό της flash, της EEPROM, των fuses και των lock beats.

- Πηγές εξωτερικών και εσωτερικών διακοπών

- 6 προγραμματιζόμενοι sleep modes

Οι καταστάσεις επιλέγονται αυτόματα από το TinyOS μέσω του χρονοπρογραμματιστή (scheduler).

- Δυνατότητα για επιλογή συχνότητας λειτουργίας μέσω λογισμικού.

- 53 προγραμματιζόμενα pins μέσω των οποίων μπορούν να συνδεθούν αισθητήρες και άλλα περιφερειακά

- Τάση λειτουργίας 2.7 – 5.5Volts

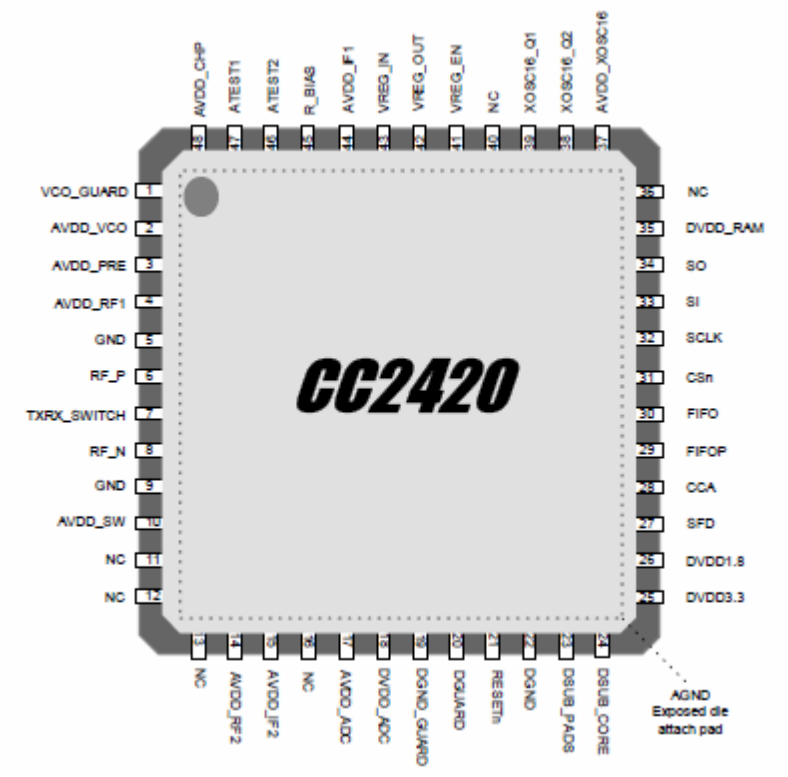
3.1.3 Ο πομποδέκτης CC2420 RF

Το CC2420[18] είναι ένα ολοκληρωμένο chip και λειτουργεί ως RF πομποδέκτης, συμβατός με το πρωτόκολλο 802.15.4/Zigbee που εκπέμπει στα 2.4GHz και είναι σχεδιασμένο για χαμηλής κατανάλωσης και χαμηλής τάσης ασύρματα δίκτυα. Το CC2420 βασίζεται στην τεχνολογία SmartRF της Chipcon στα 0,18μm CMOS. Επίσης, περιέχει ένα modem βασικής ζώνης με τεχνολογία DSSS το οποίο παρέχει κέρδος 9dB και έναν ικανοποιητικό ρυθμό μετάδοσης δεδομένων στα 250Kbps.

Το CC2420 παρέχει εκτεταμένη υποστήριξη υλικού για:

- Packet Handling (διαχείριση των πακέτων δεδομένων).
- Data Buffering (προσωρινή αποθήκευση δεδομένων).
- Burst Transmissions (μεταφορά δεδομένων κατά ριπές).
- Data Encryption (κρυπτογράφηση δεδομένων).
- Data Authentication (πιστοποίηση δεδομένων).
- Link Quality Indication (ένδειξη ποιότητας σύνδεσης).
- Packet Timing Information (συγχρονισμός πακέτων δεδομένων).

Τα χαρακτηριστικά αυτά μειώνουν το φόρτο εργασίας που ανατίθεται στον μικροελεγκτή και επιτρέπουν στο CC2420 να διασυνδέεται με χαμηλής ισχύος μικροελεγκτές. Ο μικροελεγκτής επικοινωνεί με το interface του CC2420 μέσω του SPI – Serial Peripheral Interface bus και σε μία τυπική εφαρμογή μπορεί να χρησιμοποιηθεί μαζί με τον μικροελεγκτή και μερικά εξω-τερικά παθητικά στοιχεία. Η εικόνα που ακολουθεί δείχνει το ολοκληρωμένο CC2420 με τα εξωτερικά του pins.



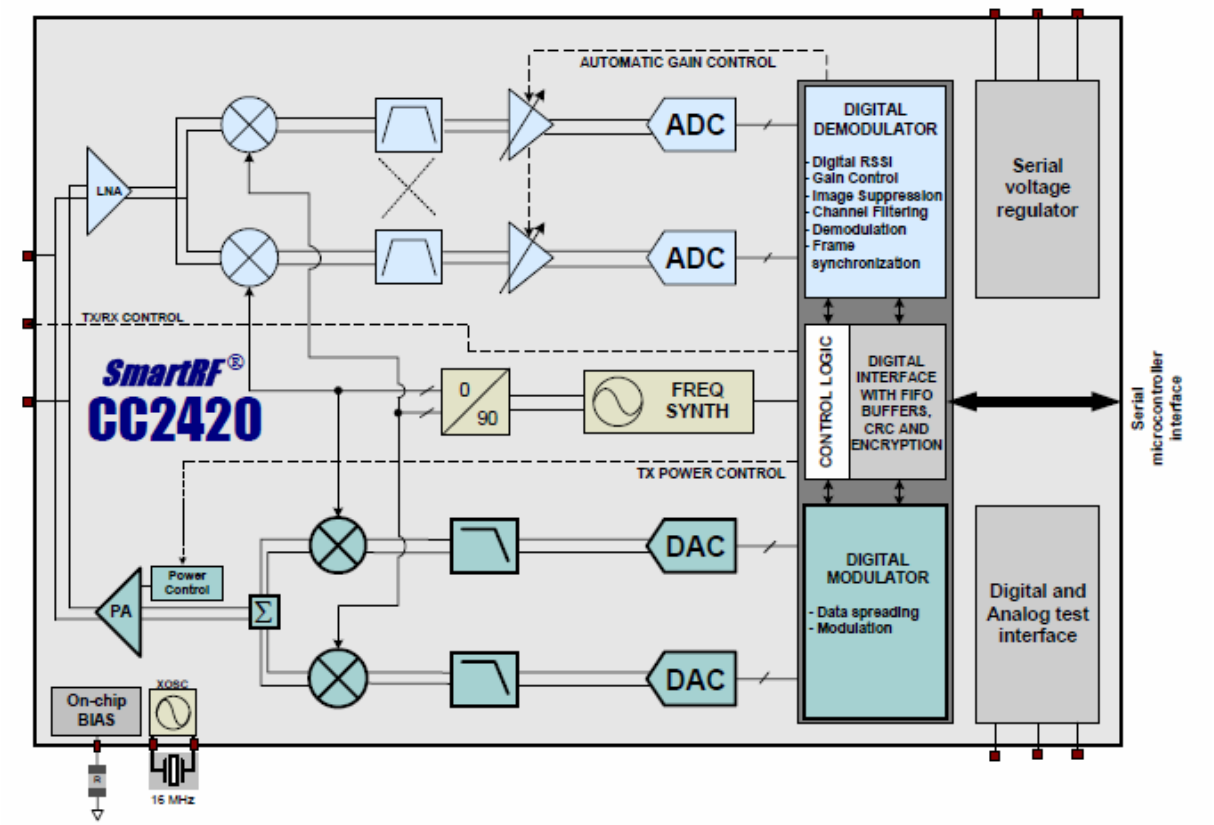
Εικόνα 3.4 Το ολοκληρωμένο CC2420

Τα πιο κύρια χαρακτηριστικά του ολοκληρωμένου είναι τα εξής:

- RF πομποδέκτης συμβατός με το πρωτόκολλο 802.15.4 με modem βασικής ζώνης και υποστήριξη του MAC επιπέδου.
- Modem βασικής ζώνης με τεχνολογία DSSS και 250Kbps ρυθμό μετάδοσης.
- Κατάλληλο για RFD και FFD λειτουργία.
- Χαμηλή κατανάλωση ρεύματος (RX: 18.8mA, TX: 17.4 mA).
- Χαμηλή τάση τροφοδοσίας (2.1-3.6Vt) με ολοκληρωμένο ρυθμιστή τάσης.
- Χαμηλή τάση τροφοδοσίας (1.6-2.0Vt) με εξωτερικό ρυθμιστή τάσης.
- Προγραμματιζόμενη ισχύς εξόδου.
- Δεν υπάρχει εξωτερικός RF διακόπτης (απαιτείται φίλτρο).

- 128(RX) + 128(TX) byte προσωρινή αποθήκευση δεδομένων.
- Hardware MAC encryption (AES-128)
- Έλεγχος της κατάστασης της μπαταρίας.
- Ευέλικτα και ισχυρά εργαλεία ανάπτυξης.

Παρακάτω δείχνουμε σε Block Διάγραμμα το εσωτερικό του CC2420 καθώς και τις τυπικές συνθήκες λειτουργίας του



Εικόνα 3.5 Block Διάγραμμα του 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

Εικόνα 3.6 Τυπικές συνθήκες λειτουργίας του CC2420

3.1.4 Τροφοδοσία και Κατανάλωση Ενέργειας

Η συσκευή λειτουργεί με δύο μπαταρίες τύπου AA, οι οποίες μπορούν να χρησιμοποιηθούν για λειτουργία στο εύρος τάσης 2.7V με 3.3V DC. Εάν η συσκευή είναι συνδεδεμένη στη θύρα USB για προγραμματισμό ή επικοινωνία με τον Η/Υ τότε μπορεί να τροφοδοτηθεί μέσω της θύρας αυτής. Σε αυτή την περίπτωση η τάση τροφοδοσίας είναι 3V και η χρήση της μπαταρίας πρέπει να αποφευχθεί διαφορετικά θα οδηγήσει σε καταστροφή της μονάδας . Η κατανάλωση ενέργειας ενός αισθητήρα δεν αφορά μόνο τον μικροελεγκτή ή τον πομποδέκτη, αλλά επίσης και τα

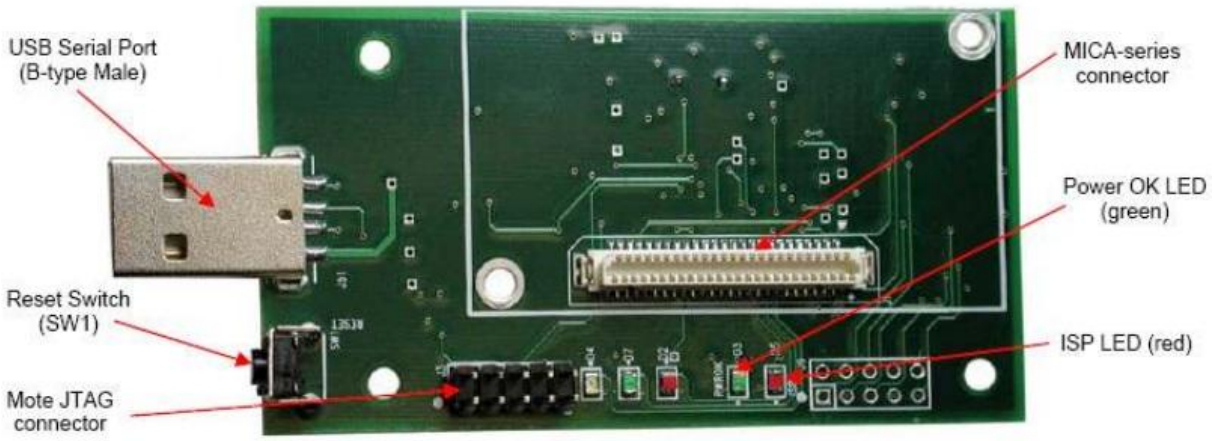
βοηθητικά στοιχεία από τα οποία αποτελείται. Στην εικόνα 3.7 που ακολουθεί φαίνεται η ποσότητα του ρεύματος που χρειάζεται για κάθε λειτουργία του micaz.

Operation	MicaZ
Minimum Voltage	2.7V
Mote Standby (RTC on)	27.0 μA
MCU Idle (DCO on)	3.2 mA
MCU Active	8.0 mA
MCU + Radio RX	23.3 mA
MCU + Radio TX (0dBm)	21.0 mA
MCU + Flash Read	9.4 mA
MCU + Flash Write	21.6 mA
MCU Wakeup	180 μs
Radio Wakeup	860 μs

Εικόνα 3.7 Οι απαιτήσεις ρεύματος του micaz

3.2 Η πλατφόρμα MIB520CB

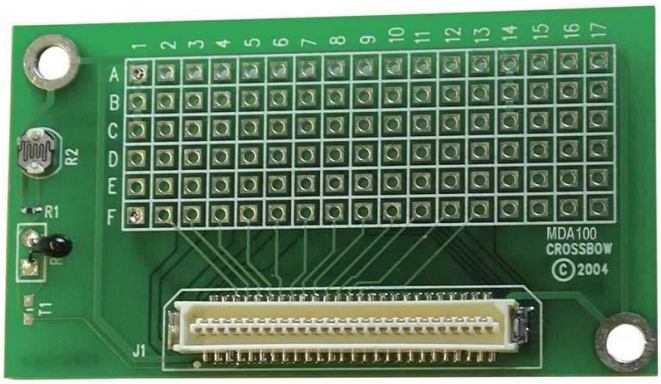
Η προγραμματιστική πλακέτα MIB520CB, μέσω της θύρας USB που ενσωματώνει, παρέχει την δυνατότητα σύνδεσης ενός ηλεκτρονικού υπολογιστή με κόμβους τύπου MICA και IRIS[19]. Έτσι είναι δυνατή η μεταφορά δεδομένων από και προς τον Η/Υ, καθώς και ο προγραμματισμός των κόμβων.



Εικόνα 3.8 Η πλατφόρμα MIB520CB

3.3 Η πλακέτα αισθητήρων MDA100CA

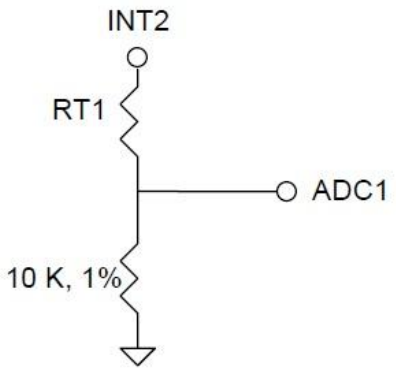
Η συγκεκριμένη πλακέτα μπορεί να χρησιμοποιηθεί σε συνδυασμό με τους κόμβους τύπου MICAz, IRIS και MICA2.Διαθέτει ένα θερμίστορ ακριβείας για την μέτρηση της θερμοκρασίας, έναν αισθητήρα φωτός, καθώς και μια περιοχή για την σύνδεση επιπλέον αισθητήρων.



Εικόνα 3.9 Η πλακέτα αισθητήρων MDA100CA

3.3.1 Ο αισθητήρας θερμοκρασίας του MDA100CA

Το θερμίστορ είναι ένας αισθητήρας με μεγάλη ακρίβεια και σταθερότητα. Ο αισθητήρας είναι συνδεδεμένος στο κανάλι ADC1 μέσω μιας αντίστασης. Στην εικόνα 3.10 φαίνεται το σχεδιάγραμμα αυτό[20].



Εικόνα 3.10 Το θερμίστορ του MDA100CA

Η μετατροπή των μετρήσεων που κάνει ο αισθητήρας σε βαθμούς Κέλβιν γίνεται από τον ακόλουθο τύπο:

$1/T(K) = a + b \times \ln(Rthr) + c \times [\ln(Rthr)]3$

Όπου έχουμε :

$Rthr = R1(ADC\_FS-ADC)/ADC1$

$a = 0.001010024$

$b = 0.000242127$

$c = 0.000000146$

$R1 = 10\ k\Omega$

$ADC\_FS = 1023$

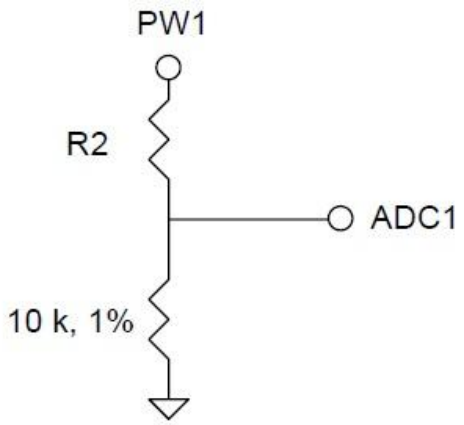
ADC1 = Η τιμή εξόδου από το κανάλι ADC από το οποίο πήραμε την μέτρηση.

Άμα θέλουμε μπορούμε να μετατρέψουμε την τιμή σε βαθμούς Κελσίου από τον τύπο :

[°C] = [K] – 273.15

3.3.2 Ο αισθητήρας φωτός του MDA100CA

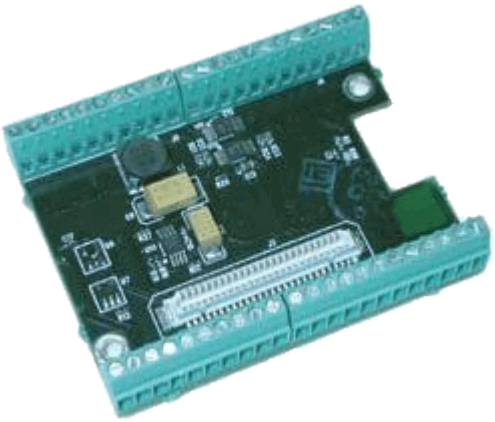
Ο αισθητήρας φωτός αποτελείται από φωτοκύτταρα[20]. Η μέγιστη ευαισθησία των φωτοκύτταρων είναι σε 690 nm μήκος κύματος του φωτός. Η αντίσταση είναι στα 2kΩ ενώ η αντίσταση όταν είναι στο σκοτάδι φτάνει τα 520 kΩ. Για να μπορέσουμε να χρησιμοποιήσουμε τον αισθητήρα φωτός πρέπει να ενεργοποιήσουμε τον PW1. Η έξοδος του αισθητήρα είναι συνδεδεμένη στο κανάλι ADC1. Στην εικόνα 3.11 δείχνουμε το σχεδιάγραμμα του αισθητήρα.



Εικόνα 3.11 Ο αισθητήρας φωτός του MDA100CA

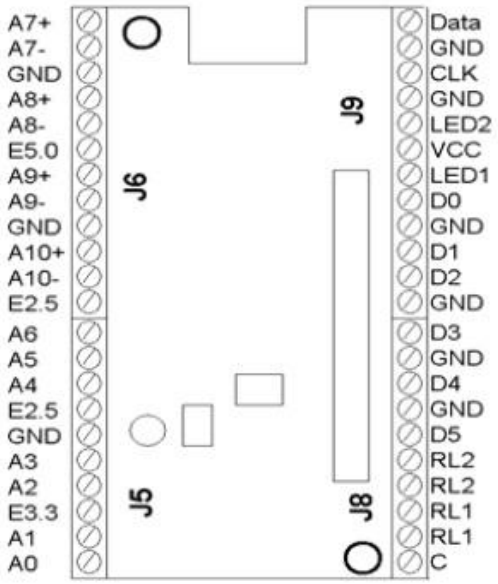
3.4 Η πλακέτα αισθητήρων MDA300CA

Η πλακέτα αισθητήρων MDA300CA[20] επιτρέπει την μέτρηση θερμοκρασίας και υγρασίας. Το πιο σημαντικό χαρακτηριστικό της συγκεκριμένης πλακέτας είναι ότι μας παρέχει μια πληθόρα επιλογών για σύνδεση εξωτερικών αισθητήρων καθώς μας επιτρέπει την σύνδεση αναλογικών και ψηφιακών καναλιών καθώς και σταθερή τάση τροφοδοσίας για τη σύνδεση των αισθητήρων. Όπως και η πλακέτα MDA100, η MDA300CA μπορεί να χρησιμοποιηθεί σε συνδυασμό με κόμβους IRIS, MICAz και MICA2.



Εικόνα 3.12 Η πλακέτα αισθητήρων MDA300CA

Όπως αναφέραμε πιο πάνω το MDA300CA έχει πληθώρα συνδέσεων. Παρακάτω δείχνουμε το σχεδιάγραμμα τους καθώς και μια σύντομη περιγραφή τους.



Εικόνα 3.13 Τα pin του MDA300CA

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

Εικόνα 3.14 Περιγραφή των pin του MDA300CA

3.5 Ο Αισθητήρας υγρασίας ECH2O-10H

Ο αισθητήρας υγρασίας που θα χρησιμοποιήσουμε είναι ο ECH2O-10H της εταιρίας Decagon. Είναι ένας αισθητήρας ακριβείας χαμηλής κατανάλωσης ο οποίος μετράει την διηλεκτρική σταθερά του εδάφους ώστε να βρει την ογκομετρική περιεκτικότητα σε νερό. Στην εικόνα 3.15 φαίνεται ο συγκεκριμένος αισθητήρας[21].

Εικόνα 3.15 Ο αισθητήρας ECH2O-10H

Οι τεχνικές προδιαγραφές του αισθητήρα είναι οι εξής :

Operation	ECH2O-10H
Χρόνος Μέτρησης	10 ms
Ακρίβεια	± .04 m3/m3
Ενεργειακές Απαιτήσεις	2.5VDC @ 2mA έως 5VDC @ 7mA
Περιβάλλον Λειτουργίας	0 έως 50°C

Εικόνα 3.16 Οι Τεχνικές προδιαγραφές του ECH2O-10H

3.6 Συνδεσμολογία του υλικού

Ο τρόπος σύνδεσης ενός κόμβου αισθητήρα με την προγραμματιστική πλακέτα MIB520 γίνεται μέσω της θύρας που διαθέτει των 51-pin. Στην εικόνα 3.17 βλέπουμε την εγκατάσταση ενός κόμβου micaz στην πλακέτα MIB520 η οποία και θα αποτελέσει τον σταθμό-βάσης μας που θα δέχεται τις μετρήσεις.

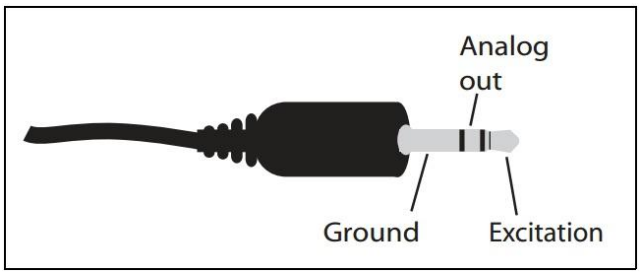
Εικόνα 3.17 Ο σταθμός-βάσης (MIB520 + Micaz)

Στις ακόλουθες δύο εικόνες δείχνουμε την σύνδεση των πλακετών αισθητήρων MDA100CA και MDA300CA αντίστοιχα μαζί με το Micaz πάνω στην προγραμματιστική πλακέτα MIB520 μέσω της οποίας θα γίνει η εγκατάσταση του λογισμικού σε κάθε κόμβο-αισθητήρα.

Εικόνα 3.18 Ο κόμβος-αισθητήρας με το MDA100CA

Εικόνα 3.19 Ο κόμβος-αισθητήρας με το MDA300CA

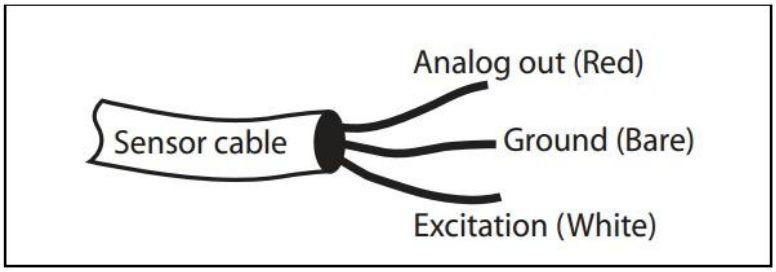
Η σύνδεση του αισθητήρα ECH2O-10H είναι αρκετά πιο περίπλοκη από την απλή τοποθέτηση των Micaz πάνω στην πλατφόρμα MIB520CA. Παρατηρούμε ότι τρόπος σύνδεσης του ECH2O-10H, για να πάρουμε τις μετρήσεις, γίνεται μέσω ενός καρφιού 3.5mm "Stereo Plug" το οποίο δείχνουμε στην εικόνα 3.20



Εικόνα 3.20 Το καλώδιο 3.5mm "Stereo Plug"

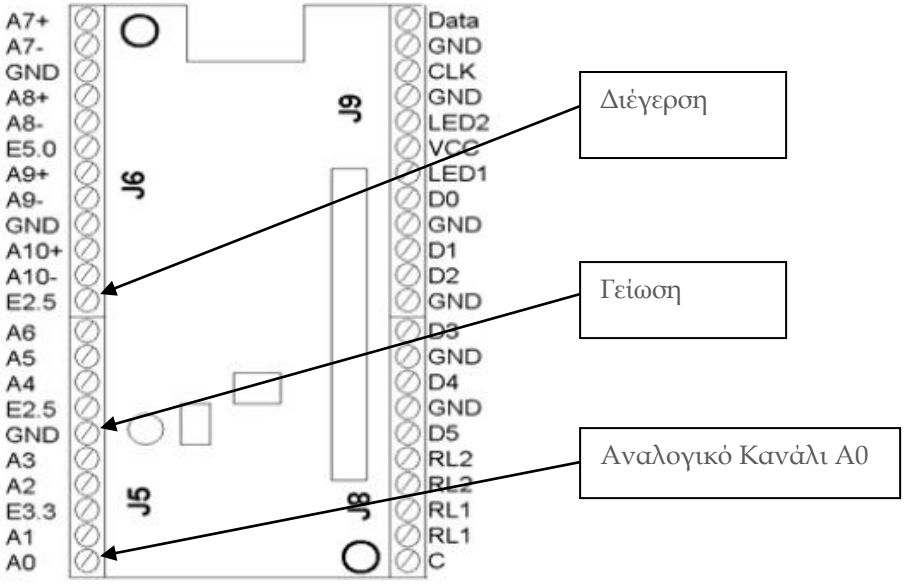
Το ECH2O-10H έχουμε την δυνατότητα να το συνδέσουμε σε καταγραφείς δεδομένων από την ίδια εταιρία την Decagon και σε αυτή τη περίπτωση δεν χρειάζεται να γίνει κάποια μετατροπή. Στην περίπτωσή μας όμως που θέλουμε να το συνδέσουμε στην πλακέτα αισθητήρων MDA300CA μας παρέχονται δύο λύσεις, Η μία είναι να κόψουμε το καλώδιο πριν από το καρφί να ξεγυμνώσουμε τα καλώδια και να τα συνδέσουμε απευθείας στην πλακέτα αισθητήρων. Αυτό έχει ως αποτέλεσμα να δημιουργήσουμε μια σταθερή σύνδεση με την πλακέτα αισθητήρων η οποία θα είναι πολύ δύσκολο να κοπεί. Φυσικά καταλαβαίνουμε ότι με αυτό τον τρόπο δεν θα μπορέσουμε να χρησιμοποιήσουμε στο μέλλον τον ίδιο αισθητήρα με κάποια άλλη πλακέτα αισθητήρων και επίσης θα είναι πολύ δύσκολη η επέκτασή του. Η άλλη λύση είναι να πάρουμε μια επέκταση για το καλώδιο των 3.5mm να το συνδέσουμε πάνω στο καρφί και μετά να

κόψουμε την επέκταση και να συνδέσουμε τα καλώδια πάνω στην πλακέτα των αισθητήρων. Στην εικόνα 3.21 βλέπουμε τις πληροφορίες για το κάθε καλώδιο μετά το την απογύμνωσή τους.



Εικόνα 3.21 Η ανατομία του καλωδίου

Λαμβάνοντας υπόψη την εικόνα 3.13 που μας δείχνει τα pin του MDA300CA, τις προδιαγραφές του ECH2O-10H καθώς και την εικόνα 3.21 παρατηρούμε ότι θα πρέπει να χρησιμοποιήσουμε μία εξωτερική σταθερή τάση για τη διέγερση του αισθητήρα της τάξης των 2.5 Volt. Επίσης θα πρέπει να συνδέσουμε την γείωση του καλωδίου με την γείωση που μας παρέχεται στην πλακέτα αισθητήρων. Τέλος θα πρέπει να χρησιμοποιήσουμε ένα αναλογικό κανάλι για να πάρουμε την μέτρηση. Επιλέξαμε, χωρίς ιδιαίτερο λόγο, την χρήση του αναλογικού καναλιού A0. Στην εικόνα 3.22 παρακάτω δείχνουμε την συνδεσμολογία την οποία προαναφέραμε.



Εικόνα 3.22 Η συνδεσμολογία του ECH2O-10H στην πλακέτα αισθητήρων MDA300CA

Κεφάλαιο 4
Το Λειτουργικό TinyOS

4.1 Εισαγωγή

Όπως αναφέραμε στα προηγούμενα κεφάλαια τα βασικά υλικά συστατικά τα οποία απαρτίζουν ένα ασύρματο δίκτυο αισθητήρων είναι περίπου ίδια. Αποτελείται από τους κόμβους-αισθητήρες οι οποίοι περιέχουν τον μικροελεγκτή, την μνήμη, τον πομποδέκτη, τον αισθητήρα και την μπαταρία. Αυτό μας κάνει να υποθέσουμε ότι για να είναι αποδοτικό το πρόγραμμα που τρέχει σε κάθε κόμβο-αισθητήρα θα πρέπει να είναι γραμμένο για τον κάθε κόμβο ξεχωριστά ανάλογα με τις απαιτήσεις τις εφαρμογής αλλά και τον αισθητήρα που έχουμε ενσωματώσει σε αυτόν. Επίσης θα πρέπει να μπορεί ο κάθε κόμβος ξεχωριστά με τον δικό του ρυθμό να κάνει τις μετρήσεις καθώς και την αποστολή τους.

4.2 Απαιτήσεις σχετικά με την σχεδίαση λειτουργικών συστημάτων για Ασύρματα Δίκτυα Αισθητήρων

Τα συμβατικά λειτουργικά αποτελούν λογισμικό συστήματος που περιλαμβάνει προγράμματα που διαχειρίζονται πόρους, ελέγχουν τις περιφερειακές συσκευές και παρέχουν αφηρημένη απεικόνιση του υλικού. Στόχος είναι η διαχείριση διεργασιών, μνήμης, χρόνου του επεξεργαστή, του συστήματος αρχείων και των συσκευών. Αυτό επιτυγχάνεται συνήθως με τον διαχωρισμό σε πολλά επίπεδα (Πυρήνας, βιβλιοθήκες συστήματος κτλ.). Τα WSN λόγω της ιδιομορφίας των εφαρμογών που εκτελούν και των περιορισμένων πόρων συστήματος, απαιτούν μια διαφορετική προσέγγιση στη σχεδίαση του λειτουργικού τους συστήματος. Σε ένα συμβατικό Λ.Σ. η εκτέλεση πολλαπλών διεργασιών γίνεται δυνατή μέσω της κατανομής μνήμης (στοίβα) για καθεμία ξεχωριστά. Το περιορισμένο μέγεθος μνήμης σε έναν ασύρματο κόμβο δεν επαρκεί για μια τέτοια διαδικασία. Επίσης η εναλλαγή στην εκτέλεση των διεργασιών δεν ευνοεί την εκτέλεση κάποιων real-time εφαρμογών που εμφανίζονται σε ένα WSN, με αποτέλεσμα να χάνονται κρίσιμα δεδομένα. Τέλος, ένα συμβατικό Λ.Σ. εξορισμού αξιοποιεί το σύνολο των δυνατοτήτων του υλικού για την καλύτερη εκτέλεση των προγραμμάτων, προσέγγιση απολύτως λογική για έναν υπολογιστή με τροφοδοσία από το δίκτυο αλλά υπερβολικά ενεργοβόρα για ένα σύστημα που τροφοδοτείται από δυο μικρές μπαταρίες[18].

Συνοπτικά οι προϋποθέσεις που πρέπει να πληρεί ένα λειτουργικό σύστημα για τα ασύρματα δίκτυα αισθητήρων είναι :

- Μικρή έκταση κώδικα.

  Δεδομένης της περιορισμένης μνήμης ενός κόμβου, ο πυρήνας του λειτουργικού πρέπει να υλοποιείται από τον ελάχιστο δυνατό κώδικα.

- Παροχή διεπαφής προγράμματος εφαρμογής

- Παροχή μηχανισμού διαχείρισης πόρων συστήματος και ενέργειας.

  Ο χρόνος απόκρισης του επεξεργαστή και η μνήμη πρέπει να κατανέμονται κατάλληλα, έτσι ώστε να διασφαλίζεται η ομαλή λειτουργία και η τήρηση αυστηρής προτεραιότητας στην εκτέλεση των διεργασιών. Η διαχείριση ενέργειας πρέπει να αποσκοπεί στην μεγιστοποίηση

της ζωής ενός κόμβου, κάνοντας σωστή χρήση των ενεργειακών καταστάσεων του επεξεργαστή (π.χ. τοποθέτηση σε idle ή sleep κατάσταση όταν αυτό είναι δυνατό).

- Δυνατότητα επαναπρογραμματισμού.

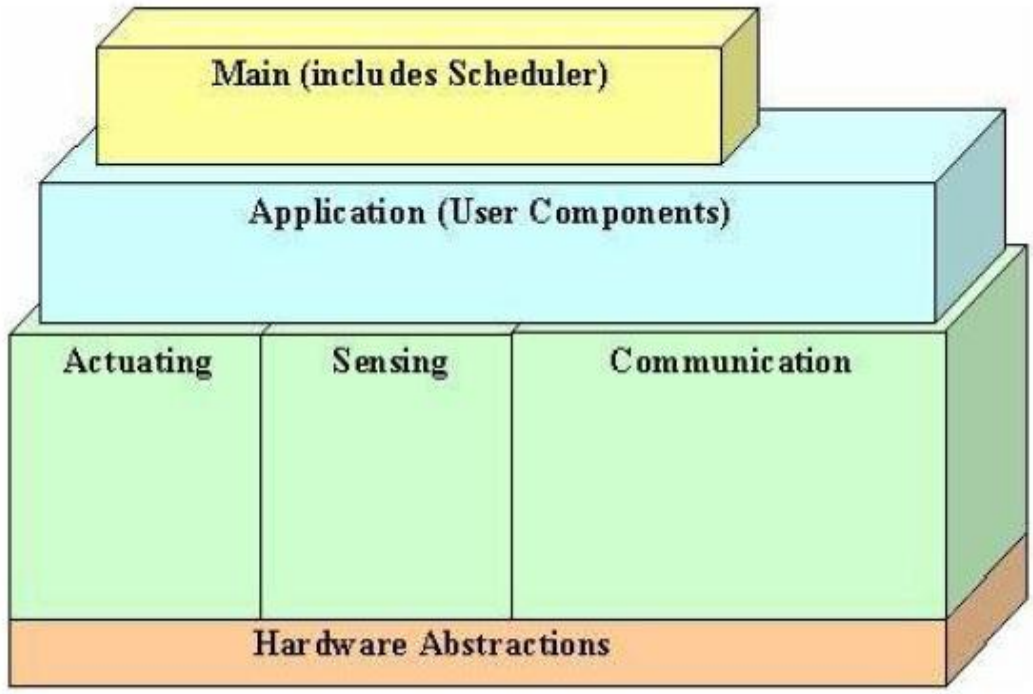
Σε ένα WSN δίκτυο, είναι συχνό φαινόμενο ο επαναπροσδιορισμός της λειτουργίας ενός κόμβου, όσον αφορά είτε τη τη συλλογή δεδομένων και τη συμπεριφορά του στο δίκτυο είτε την προσαρμογή της κατανάλωσης ενέργειας.

4.3 Η δημιουργία του TinyOS

Το TinyOS[23],[24] είναι από τα πρώτα λειτουργικά συστήματα που σχεδιάστηκαν ειδικά για WSN και είναι ανοικτού κώδικα (open source). Η ανάπτυξη του ξεκίνησε στα τέλη της προηγούμενης δεκαετίας από το πανεπιστήμιο Berkeley της Καλιφόρνια και η πρώτη έκδοση κυκλοφόρησε το 2000. Υποστηρίζει μέχρι στιγμής διαφόρων ειδών πλατφόρμες hardware, εμπορικές και μη, με επεξεργαστές από 8 bit αρχιτεκτονική και μόλις 2 KB RAM, μέχρι επεξεργαστές αρχιτεκτονικής 32-bit και μνήμης άνω των 32MB. Μερικές από αυτές είναι το Micaz της εταιρίας Crossbow, το iMote της Intel και το Eyes, αποτέλεσμα κοινοπραξίας Ευρωπαϊκών πανεπιστημίων και εταιριών. Η ανάπτυξη των εφαρμογών TinyOS γίνεται σε μια νέα γλώσσα, διάλεκτο της C, προσαρμοσμένη στις απαιτήσεις και τους περιορισμούς των WSN, την NesC (Network embedded systems C). Ο compiler της NesC καθώς και τα απαραίτητα εργαλεία Atmel AVR binutils είναι γραμμένα σε C. Το λειτουργικό περιέχει επίσης ενσωματωμένες κάποιες βοηθητικές εφαρμογές σε Java. Η εγκατάσταση του γίνεται σε ένα host PC (με λειτουργικό Linux ή Windows με την προσθήκη του Unix προσομοιωτή Cygwin) και μέσω της σειριακής ή της USB θύρας του γίνεται ο προγραμματισμός των κόμβων και η συλλογή των δεδομένων. Τελευταία έκδοση είναι η 2.12, που κυκλοφόρησε τον Αύγουστο του 2012. Στην ιστοσελίδα του TINYOS αναφέρεται ότι πάνω από 500 ερευνητικές ομάδες χρησιμοποιούν πλατφόρμες που τρέχουν TinyOS συνεισφέροντας στην ανάπτυξη κώδικα, ενώ άλλες κοινότητες ελέγχουν νέα πρωτόκολλα και αλγορίθμους μέσω προσομοιώσεων.

4.4 Η Αρχιτεκτονική του TinyOS

Η αρχιτεκτονική του TinyOS περιλαμβάνει ένα scheduler (χρονοπρογραμματιστή) και ένα σύνολο από διαβαθμισμένα components που αποτελούν το software. Ουσιαστικά απεικονίζεται μια στοίβα από components, η έννοια των οποίων υποκαθιστά εν μέρει την έννοια της διεργασίας σε συμβατικά Λ.Σ.. Τα περισσότερα components είναι modules λογισμικού, αυτά που ανήκουν στο κάτω μέρος της στοίβας όμως χρησιμοποιούνται κυρίως σαν απλά interfaces για το υλικό του συστήματος. Επίσης έννοιες όπως πυρήνας, process management, εικονική μνήμη, δυναμική κατανομή μνήμης και software signals, δεν ορίζονται. Από τη στιγμή που δεν υπάρχει πυρήνας η διαχείριση του hardware γίνεται άμεσα, υπάρχει μονό μια διαδικασία στο σύστημα, γραμμικός χώρος διευθύνσεων και η μνήμη κατανέμεται στατικά κατά τη διάρκεια της μεταγλώτισσης. Τέλος όταν λέμε ότι ένας κόμβος τρέχει TinyOS, εννοούμε ότι έχει εγκατεστημένη στη flash μνήμη του μια binary εκτελέσιμη εικόνα με τις βιβλιοθήκες του TinyOS, συνδεδεμένες με την εφαρμογή που πρέπει να εκτελεστεί. Η εικόνα αυτή αποτελεί την εφαρμογή TinyOS. Άλλωστε το να εγκαταστήσουμε το TinyOS σε ένα κόμβο δεν έχει νόημα καθώς από μόνο του δεν εκτελεί κάποια λειτουργία, ούτε έχει κάποιο user interface. Στην εικόνα 4.1 φαίνεται σχηματικά η αρχιτεκτονική του TinyOS.



Εικόνα 4.1 Η αρχιτεκτονική του TinyOS

4.5 Τα βασικά στοιχεία του TinyOs

4.5.1 Τα συστατικά (Components)

Η nesC είναι μια γλώσσα βασισμένη σε συστατικά (components). Τα συστατικά της nesC χρησιμοποιούν ένα καθαρά τοπικό όνομα. 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 όνομα. Ένα διαφορετικό συστατικό, C, το οποίο καλεί τη λειτουργία Β εισάγει το όνομα C.B ως ένα καθολικό όνομα. 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). Τα components μπορούν να χωριστούν σε κατηγορίες με βάση τη διαβάθμιση τους:

- Αφηρημένες απεικονίσεις υλικού (Hardware abstractions)

  Εδώ ανήκουν τα components χαμηλού επιπέδου. Σε αυτά γίνεται ουσιαστικά η αποτύπωση του hardware, όπως οι συσκευές εισόδου / εξόδου, ο πομποδέκτης και οι αισθητήρες. Ο κώδικας τους αποτελείται κυρίως από μακροεντολές.

- Συνθετικό υλικό (synthetic hardware)

  Το αμέσως επόμενο επίπεδο, όπου στα components αποτυπώνεται η συμπεριφορά προχωρημένων λειτουργιών του υλικού. Επικάθονται και διασυνδέονται με τις αφηρημένες

απεικονίσεις υλικού. Επειδή τα components συνθετικού υλικού αποτελούν συνδετικό κρίκο μεταξύ των hardware abstractions και του High Level Software, ο κώδικας τους είναι ένα μείγμα μακροεντολών, προγραμματισμού σε assembly και προγραμματισμού σε nesC.

- Υψηλού επιπέδου λογισμικό (High level software)

Όπως υπονοεί το όνομα είναι components υψηλού επιπέδου, που περιέχουν την εφαρμογή που έχει προγραμματίσει ο χρήστης.

Κάθε component μπορεί να περιέχει τρία είδη συναρτήσεων: command handlers, event handlers και tasks.Επίσης υπάρχει και ένας Χρονοπρογραμματιστής.

4.5.2 Εντολές (Commands)

Οι εντολές είναι αιτήσεις (requests) προς τα συστατικά χαμηλού επιπέδου. Η εκτέλεση τους είναι άμεση και χωρίς δυνατότητα φραγής, γι'αυτό και ολοκληρώνεται σε πολύ μικρό χρονικό διάστημα. Έχουν την δυνατότητα να καλούν άλλες εντολές χαμηλότερου επιπέδου, tasks αλλά δεν μπορούν να σηματοδοτούν events. Μαζί με τα events αποτελούν τον μηχανισμό επικοινωνίας μεταξύ των συστατικών.

4.5.3 Γεγονότα (Events)

Υπάρχουν δύο κατηγορίες events τα σύγχρονα και τα ασύγχρονα events. Τα σύγχρονα events συνήθως σηματοδοτούν την λήξη μιας λειτουργίας . Τα ασύγχρονα events προκύπτουν από hardware interrupts για παράδειγμα ένα event θα μπορούσε να είναι το τέλος της λειτουργίας boot του κόμβου . Κάτι πολύ σημαντικό που πρέπει να αναφέρουμε για τα events είναι ότι δε μπορούν να διακοπούν παρά μονό από ανώτερης σημασίας ασύγχρονο event . Μια εντολή για παράδειγμα, μπορεί να ζητήσει από έναν αισθητήρα την έναρξη συλλογής δεδομένων. Ένα event θα προκληθεί όταν ο αισθητήρας ολοκληρώσει τη συλλογή και είναι έτοιμος να τις στείλει στις εισόδους που είναι συνδεδεμένος. Ένα event μπορεί να εμφανιστεί και ασύγχρονα εξαιτίας μιας διακοπής (interrupt) του υλικού. Τα events όπως και οι εντολές δεν υπόκεινται σε φραγή εκτέλεσης. Με την χρήση εντολών και events, επιτυγχάνεται η εκτέλεση των λειτουργιών του λογισμικού σε δυο φάσεις (split phase operation), όπως ακριβώς συμβαίνει και με το υλικό. Μετά από την αίτηση εκτέλεσης μιας λειτουργίας, η εντολή επιστρέφει αμέσως και το αντίστοιχο event σηματοδοτεί το τέλος της λειτουργίας μετά τη μεσολάβηση κάποιου χρονικού διαστήματος.

Τέλος, τα events έχουν τη δυνατότητα να καλέσουν εντολές και tasks χαμηλότερου επιπέδου, καθώς και να σηματοδοτήσουν events ανωτέρου επιπέδου.

4.5.4 Διεργασίες (Tasks)

Τα events και τα commands είναι διεργασίες μικρές αλλά απόλυτα αναγκαίες . Τα υπόλοιπα υπολογιστικά κομμάτια μιας εφαρμογής που απαιτούν περισσότερο χρόνο εκτέλεσης αλλά είναι χαμηλότερης σημασίας υλοποιούνται με τα tasks .Τα tasks καλούνται από commands ή events και δεν εκτελούνται όσο υπάρχουν events και commands σε εκτέλεση από το λειτουργικό . Το σημαντικό είναι ότι ένα task δεν μπορεί να σταματήσει τη λειτουργία από ένα άλλο task άλλα ούτε από σύγχρονο event. Άρα ο σωστός τρόπος δημιουργίας ενός task από ένα σωστό προγραμματιστεί του tinyOs είναι να τα κρατάει μικρά σε μέγεθος έτσι ώστε να μην έρθει στη δυσάρεστη θέση να χάνει πακέτα επειδή εκτελώ το λειτουργικό μεγάλο task.

4.5.5 Χρονοπρογραμματιστής (Scheduler)

Ο scheduler οργανώνει την λειτουργία των components. Χρησιμοποιεί έναν απλό FIFO μηχανισμό και η φιλοσοφία λειτουργίας του γενικά είναι non-preemptive. Κάθε task που καλείται τοποθετείται στη στοίβα του scheduler. Αν δεν τρέχει κάποιο event, επιτρέπει την εκτέλεση των tasks που έχουν καταφτάσει, ένα κάθε φορά, διασφαλίζοντας την ολοκλήρωση τους. Ένα ασύγχρονο event έχει μεγαλύτερη σημασία από ένα task, οπότε αν υπάρχουν μόνο tasks στη στοίβα και ξαφνικά έρθει ένα event, μπορεί να διακοπεί η εκτέλεση του task και να αρχίσει η εκτέλεση του event. Με άλλα λόγια, ένα task μπορεί να γίνει pre-empted από ένα event, δεν μπορεί να γίνει όμως το αντίθετο. Η εκτέλεση των events είναι ατομική μεταξύ τους, όπως ατομική είναι και η εκτέλεση μεταξύ διαφορετικών tasks. Όταν η στοίβα αδειάσει ο scheduler δίνει εντολή στον επεξεργαστή να μπει σε κατάσταση SLEEP. Τα περιφερειακά όμως παραμένουν ενεργά, οπότε μια διακοπή μπορεί να επενεκκινήσει την ίδια διαδικασία. Ο scheduler υλοποιείται στο αρχείο sched.c που υπάρχει στον κατάλογο system του TinyOS και διαφέρει από έκδοση σε έκδοση.

4.6 Η γλώσσα προγραμματισμού NesC

Ο αρχικός προγραμματισμός του TinyOS έγινε σε C , αλλά χρειάστηκε μια καινούργια γλώσσα για να μπορέσει να υποστηρίξει την component – based λογική του και έτσι δημιουργήθηκε η nesC (network embedded system C). Η nesC είναι component – based , event-driven γλώσσα προγραμματισμού και σχεδιάστηκε ως προέκταση της γλώσσας C για την δημιουργία εφαρμογών , για την πλατφόρμα του TinyOS. Τα βασικά χαρακτηριστικά της nesC είναι τα εξής:

- 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.

- Τα components είναι συνδεδεμένα στατικά μεταξύ τους μέσω των διεπαφών. Αυτό αυξάνει την αποτελεσματικότητα κατά την εκτέλεση και επιτρέπει καλύτερη στατική ανάλυση του προγράμματος

- 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.

Παρακάτω θα αναλύσουμε τα βασικά στοιχεία που απαρτίζουν ένα πρόγραμμα γραμμένο σε nesC.

4.6.1 Διεπαφές (Interfaces)

Τα interfaces αποτελούν αναπόσπαστο κομμάτι ενός προγράμματος. Χρησιμεύουν στην απόδοση ενός αφηρημένου ορισμού στην αλληλεπίδραση μεταξύ των components. Τα interfaces μπορούν να παρέχονται ή να χρησιμοποιούνται από ένα component. Τα παρεχόμενα interfaces

αντιπροσωπεύουν τις λειτουργίες που παρέχει ένα component (provider), ενώ τα χρησιμοποιούμενα interfaces, τις λειτουργίες που απαιτεί ένας χρήστης (user) να του δοθούν για την εκτέλεση της λειτουργίας που του έχει ανατεθεί. Σε αναλογία με την Java, ένα interface απαγορεύεται να περιέχει κώδικα για εκτέλεση υπολογισμών. Το μόνο που μπορεί να περιέχει είναι απλές δηλώσεις συναρτήσεων (commands) που πρέπει να υλοποιούνται από το component που παρέχει το interface και δηλώσεις συναρτήσεων (events) που πρέπει να υλοποιηθούν από το component που χρησιμοποιεί το interface. Αυτός ο τρόπος χρήσης των interfaces τους δίνει την ιδιότητα της διπλής κατεύθυνσης και σχετίζεται με την split-phase operation. Πιο συγκεκριμένα, ένα component δεν μπορεί να καλέσει μια command αν δεν παρέχει μια υλοποίηση για το αντίστοιχο event που θα προκαλέσει στο τέλος της εκτέλεσής της. Ένα παράδειγμα interface είναι το interface Timer το οποίο από ότι λέει και το όνομά του είναι ένα ρολόι . Ας το δούμε όμως αναλυτικά:

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

Όπως προείπαμε το όνομα του interface είναι Timer και δηλώνονται δύο εντολές: η start() και η stop() και ένα γεγονός το fired(). Ένα component που παρέχει το interface πρέπει οπωσδήποτε να περιέχει υλοποιήσεις των commands start και stop ενώ ένα component που θα χρησιμοποιήσει το Timer, πρέπει να υλοποιεί το event fired(). Διαφορετικά θα μας βγάλει λάθος στην μεταγλώττιση.

4.6.2 Modules

Υπάρχουν δύο είδη συστατικών στη γλώσσα NesC. Τα modules και τα configurations. Τα module περιέχουν τον κώδικα της εφαρμογής που υλοποιεί ή χρησιμοποιεί ένα ή και περισσότερα interfaces. Μέσα στον κώδικα του module βρίσκεται ο πυρήνας μιας εφαρμογής TinyOS. Εδώ καθορίζεται η λειτουργία που θέλουμε να εκτελέσει το πρόγραμμα μας. Η δομή ενός module το οποίο ανάβει το κόκκινο LED κάθε φορά που τελειώνει ο χρόνος έχει ως εξής :

```
module BlinkM {
        provides {
                interface StdControl;
        }
        uses {
                interface Timer;
                interface Leds;
        }
}
implementation {
        command result_t StdControl.init() {
        call Leds.init();
        return SUCCESS;
}


 command result_t StdControl.start() {
          return call Timer.start(TIMER_REPEAT, 1000);
}
command result_t StdControl.stop() {
          return call Timer.stop();
}
event result_t Timer.fired() {
call Leds.redToggle();
return SUCCESS;
}
}
```

Παρατηρούμε ότι για το interface StdControl που παρέχει το module BlinkM, υπάρχουν και οι αντίστοιχες υλοποιήσεις των commands που αυτό ορίζει. Εφόσον υπάρχουν στο module οι commands init(), start() και stop(), αντιλαμβανόμαστε ότι θα είναι και στο interface StdControl δηλωμένες. Στο σώμα της StdControl.init(), υπάρχει η εντολή call Leds.init() η οποία σημαίνει ότι καλείται η command init(), που δηλώνεται στο interface Leds και εκτελεί μια λειτουργία. Γι'αυτό και λέμε ότι το BlinkM χρησιμοποιεί το interface Leds. Προφανώς η Leds.init(), υλοποιείται σε κάποιο άλλο module που παρέχει το interface Leds. Το BlinkM χρησιμοποιεί και άλλο ένα

interface, το Timer. Το Timer περιέχει τη συνάρτηση event fired(), γι'αυτό και στο BlinkM υπάρχει η αντίστοιχη υλοποίηση της.

4.6.3 Configuration

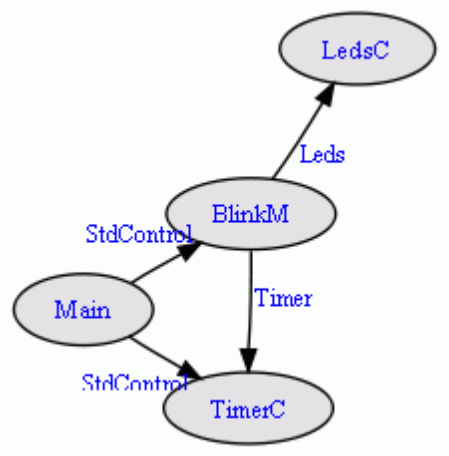
Το Configuration είναι το δεύτερο συστατικό της γλώσσας NesC. Παρατηρώντας τον παραπάνω κώδικα όταν δηλώσαμε τα interface που παρέχει και χρησιμοποιεί το module BlinkM (uses interface Leds;) δημιουργείτε το ερώτημα που ξέρει το module που να κοιτάξει να βρει αυτό το interface. Αυτή τη δουλειά αναλαμβάνει να κάνει το configuration component. Η διαδικασία σύνδεσης των components μεταξύ τους αποκαλείται wiring. Ένα configuration όσον αφορά τη βασική δομή μοιάζει με ένα module. Χωρίζεται σε πεδίο για specification και σε πεδίο για implementation. Συντακτικά τα configurations είναι πολύ απλά. Διαθέτουν μόλις τρεις τελεστές. Δεξί βέλος (->), αριστερό βέλος (<-) και ίσον (=). Οι δυο πρώτοι εξυπηρετούν το βασικό wiring, το οποίο είναι ουσιαστικά μια διαδικασία αντιστοίχησης interface ενός component σε interface ενός άλλου component. Η κατεύθυνση είναι πάντα από την πλευρά του χρήστη (user) προς αυτή του παροχέα (provider) Παρακάτω βλέπουμε ένα παράδειγμα μιας τέτοιας δήλωσης:

```
BlinkM.Leds -> LedsC.Leds;
LedsC.Leds <- BlinkM.Leds ;
```

Θα πρέπει να αναφέρουμε ότι πιο συνήθης δήλωση είναι η πρώτη. Επίσης θα πρέπει να τονισθεί ότι κάθε TinyOS εφαρμογή έχει τουλάχιστον ένα configuration component το οποίο πρέπει να συνδέσει το component της εφαρμογής μας με άλλα component του συστήματος . Κατ' αυτόν τον τρόπο ένα module που χρησιμοποιεί ένα interface, βρίσκει το module στο οποίο υλοποιείται το interface μέσω του wiring στο configuration. Ας δούμε για παράδειγμα το configuration για το module BlinkM.

```
configuration Blink {
}
implementation {
components Main, BlinkM, SingleTimer, LedsC;
Main.StdControl -> BlinkM.StdControl;
Main.StdControl -> SingleTimer.StdControl;
BlinkM.Timer -> SingleTimer.Timer;
BlinkM.Leds -> LedsC;
}
```

Στην πρώτη γραμμή δηλώνουμε το όνομα του configuration. Το πεδίο του specification που ακολουθεί είναι άδειο στην συγκεκριμένη εφαρμογή. Αμέσως μετά έχουμε το πεδίο υλοποίησης το οποίο ξεκινάει με την λέξη implementation. Στην πρώτη γραμμή ορίζουμε πια components θα χρησιμοποιήσει το configuration για την διασύνδεση. Στην συγκεκριμένη εφαρμογή έχουμε τα Main, BlinkM, SingleTimer, LedsC. Στις επόμενες σειρές βλέπουμε την χρήση του τελεστή ->. Εδώ γίνεται η καλωδίωση των συστατικών μας. Το component Main χρησιμοποιεί το interface StdControl που υλοποιείται από το component BlinkM. Θα πρέπει να αναφέρουμε ότι το component Main συνδέεται μέσω μιας διαδρομής με τον scheduler του TinyOS, γι'αυτό είναι και το πρώτο που καλείται σε κάθε εφαρμογή. Για την ακρίβεια, η συνάρτηση Main.StdControl.init() είναι η πρώτη που εκτελείται και την ακολουθεί η Main.StdControl.start(). Το interface StdControl χρησιμοποιείται για την αρχικοποίηση των components που το παρέχουν. Έτσι τα components BlinkM και Timer που παρέχουν το StdControl στο Main, θα αρχικοποιηθούν με την έναρξη της εφαρμογής. Στη συνέχεια ορίζεται ότι το BlinkM χρησιμοποιεί το interface Timer που παρέχεται από το SingleTimer και το interface Leds που παρέχεται από το LedsC. Στην εικόνα 4.2 βλέπουμε μια γραφική αναπαράσταση για την διασύνδεση.



Εικόνα 4.2 Γραφική αναπαράσταση του wiring

Άλλη μια παρατήρηση που πρέπει να κάνουμε αφορά την σύνταξη της εντολής BlinkM.Leds -> LedsC; . Παρατηρούμε ότι στο δεξιό κομμάτι του εντολής δεν έχουμε την κλασική έκφραση που θα περιμέναμε του στυλ BlinkM.Leds -> LedsC.Leds;. Όταν δεν αναφέρουμε την χρήση του interface τότε υπονοείται η χρήση του interface από το αριστερό κομμάτι. Δηλαδή οι εκφράσεις BlinkM.Leds -> LedsC; Και BlinkM.Leds -> LedsC.Leds; είναι ταυτόσημες.

4.6.4 Οι Διεργασίες (Tasks) στην NesC

Προηγουμένως κάναμε αναφορά στην χρήση των tasks. Παρακάτω θα δούμε πως μέσο της nesC μπορούμε να τα χρησιμοποιήσουμε .Ένα task δε δέχεται ποτέ παραμέτρους και επιστρέφει void. Στη συνέχεια το task χωρίζεται ως εξής :

```
task void taskName() {
//Εδώ γράφουμε τον κώδικα που καθορίζει τη λειτουργία του task για να τελειώσει τη λειτουργία //του το task ακολουθεί η παρακάτω εντολή.
signal taskNameDone();
}
```

Ένα task χρησιμοποιεί τη λέξη post για να καλέσει ένα command η event . Η post τοποθετεί στη στοιβάδα του χρονοπρογραμματιστή την task η οποία θα εκτελεστεί την ώρα που πρέπει . Επιστρέφει success εάν το συγκεκριμένο task δεν υπάρχει στο task query και fail αν υπάρχει στη στοίβα και δεν έχει ακόμα εκτελεστεί .

4.6.5 Async συναρτήσεις και Atomic δηλώσεις

Επειδή τα task είναι non – preemptive και αυτό θα μπορούσε να προκαλέσει πολλά προβλήματα τη λύση την φέρνει η nesC δίνοντας μας τη δυνατότητα να δημιουργήσουμε συναρτήσεις ασύγχρονες οι οποίες αν χρειαστεί σταματάνε τα task για να εκτελεστούνε. Οι συναρτήσεις που σχετίζονται με διακοπές ονομάζονται hardware interrupt handlers και δηλώνονται με τη λέξη async μπροστά. Για παράδειγμα, το interface ADC που περιγράφει τον A/D Converter και αναφέρεται στην δειγματοληψία που είναι καθαρά μία ασύγχρονη διαδικασία, ορίζεται ως :

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

Εδώ όμως πρέπει να επισημάνουμε ότι η ασύγχρονη εκτέλεση μπορεί να προκαλέσει προβλήματα σε τμήματα κώδικα όπου υπάρχουν εντολές που καταχωρούν σε μεταβλητές ληφθέντα δεδομένα. Ας δούμε ένα χαρακτηριστικό παράδειγμα για την πρόκληση προβλήματος με την χρήση της ασύγχρονης εκτέλεσης. Έστω ότι εκτελείται ο κώδικας :

```
async event dataReady(uint16_t data){
DataReads=data;
post Store();
return SUCCESS ;
}
```

Το event dataReady() σηματοδοτεί ότι στον A/D υπάρχουν δεδομένα (data), τα οποία λαμβάνονται και καταχωρούνται στην μεταβλητή DataReads. Στη συνέχεια καλείται το task

Store(), μέσω της εντολής post, που αποθηκεύει την τιμή στην EEPROM. Αν για κάποιο λόγο ένα καινούριο event dataReady() εμφανιστεί πριν γίνει η καταχώρηση στην DataReads, η τιμή που είχε μετρηθεί πρώτα δεν θα αποθηκευτεί ποτέ στην EEPROM καθώς θα καλυφθεί από την καινούρια.

Τα atomic statements φτιάχτηκαν για να δίνουν στο χρήστη τη δυνατότητα προστασίας μεταβλητών που είναι προσβάσιμες από ασύγχρονες διεργασίες. Η χρήση της δήλωσης atomic πριν από κάθε εντολή που θέλουμε να τρέξει απρόσκοπτα μας δίνει αυτή τη δυνατότητα. Έτσι, αν γράψουμε atomic DataReads=data, η εντολή θα εκτελεστεί ότι και αν συμβεί. Τα atomic statements πρέπει να χρησιμοποιούνται προσεκτικά διότι κατά την εκτέλεσή τους απενεργοποιούνται οι διακοπές (interrupts), οπότε η άσκοπη χρήση τους προκαλεί χάσιμο κύκλων του επεξεργαστή.

Κεφάλαιο 5
Το Πρωτόκολλο Επικοινωνίας XMesh

5.1 Εισαγωγή

Το XMesh[25] είναι ένα πρωτόκολλο δικτύου το οποίο σχεδιάστηκε από την εταιρία Crossbow για multi – hop, ad – hoc, mesh WSN. Πρόκειται ουσιαστικά για μια βιβλιοθήκη λογισμικού η οποία χρησιμοποιεί το λειτουργικό σύστημα TinyOS που τρέχει στους κόμβους MICA2, MICAz, MICA2DOT και IRIS της ίδιας εταιρίας. Στόχος είναι η επικοινωνία των κόμβων μεταξύ τους, ακόμα κι αν ο ένας δεν βρίσκεται εντός της ακτίνας επικοινωνίας του άλλου, γεγονός που επιτυγχάνεται με την τεχνική multi – hopping, δηλαδή την προώθηση μηνυμάτων σε ενδιάμεσους κόμβους – σταθμούς, μέχρι αυτά να φτάσουν στον τελικό προορισμό τους. Μεγάλο μέρος του πρωτοκόλλου βασίζεται στα πρότυπα ZigBee και 802.15.4 .

5.2 Μονάδες Δικτύου

Οι μονάδες που απαρτίζουν ένα multi – hop mesh δίκτυο είναι τεσσάρων ειδών :

- Τελικοί κόμβοι (Endpoints): Κόμβοι οι οποίοι βρίσκονται στις άκρες του δικτύου. Στέλνουν μόνο δικά τους δεδομένα και δεν προωθούν πακέτα που μπορεί να φτάσουν από γειτονικούς κόμβους.

- Routers: Δημιουργούν τα δικά τους μηνύματα και ταυτόχρονα λειτουργούν ως ενδιάμεσοι προωθώντας πακέτα που λαμβάνουν από γειτονικούς κόμβους.

- Gateways: Συλλέγουν δεδομένα από το δίκτυο και παρέχουν διασύνδεση με τον κεντρικό σταθμό του δικτύου. Είναι ουσιαστικά η πύλη μέσω της οποίας ρυθμίζονται οι παράμετροι του δικτύου και επιτηρείται η λειτουργία του.

- Λογισμικό Συστήματος (System Software): Παρέχει το πρωτόκολλο δικτύου που επιτρέπει την αυτοοργάνωση και αυτορύθμιση του δικτύου.

5.3 Χαρακτηριστικά

Όπως αντιλαμβανόμαστε τα χαρακτηριστικά του XMesh είναι προσανατολισμένα στο να είναι αποδοτικά για τα ασύρματα δίκτυα αισθητήρων. Παρακάτω αναφέρουμε μερικά από αυτά.

5.3.1 TrueMesh

Η τεχνολογία TrueMesh αναφέρεται στην ικανότητα των κόμβων να κάνουν δυναμική αναζήτηση νέων διαδρομών για την παράδοση πακέτων, ειδικά όταν τμήματα του δικτύου βγαίνουν εκτός λειτουργίας είτε λόγω παρεμβολών, είτε λόγω της περίπτωσης όπου κάποιος κόμβος έχει εξαντλήσει την ενέργεια του ή απλά βρίσκεται για τη συγκεκριμένη χρονική στιγμή σε κατάσταση sleep. Οι κόμβοι που διασκορπίζονται στο δίκτυο, αναγνωρίζουν ο ένας τον άλλο και και δημιουργούν ένα δένδρο διαδρομών που βασίζεται στην ισχύ των σημάτων που λαμβάνουν.

5.3.2 Quality of Service (QoS) υπηρεσίες

Παρέχονται δυο επιλογές όσον αφορά την αξιοπιστία της επικοινωνίας:

- Best Effort: Σε αυτή την περίπτωση γίνεται η καλύτερη δυνατή προσπάθεια για την παράδοση του πακέτου σε τοπικό επίπεδο. Ο κόμβος δηλαδή περιμένει επιβεβαίωση λήψης από τον γείτονα και αν δεν την παραλάβει ξαναπροσπαθεί (Link Level Acknowledgement).

- Εγγυημένη παράδοση: Εδώ για ένα μήνυμα που στέλνεται upstream ή downstream, ένα σήμα ACK αποστέλλεται πάντα πίσω στον κόμβο που έστειλε το αρχικό μήνυμα (End to End Acknowledgement).

5.3.3 Πολλαπλά είδη μετάδοσης

Το XMesh παρέχει τρεις τρόπους επικοινωνίας μεταξύ των κόμβων:

- Upstream: Παράδοση πακέτων από έναν κόμβο στον κεντρικό σταθμό.

- Downstream: Παράδοση πακέτων από τον κεντρικό σταθμό σε έναν ή περισσότερους κόμβους.

- Single – Hop: Παράδοση πακέτων μόνο σε γειτονικούς κόμβους.

5.3.4 Health Diagnostics

Υπάρχει η δυνατότητα οι κόμβοι να στέλνουν περιοδικά πληροφορίες σχετικά με την κατάσταση τους. Οι πληροφορίες αφορούν στοιχεία όπως η κίνηση δικτύου, η τάση της μπαταρίας, ο κόμβος γονέας και οι γείτονες του κόμβου και ισχύς του σήματος που στέλνουν (δείκτης RSSI[2]).

5.3.5 Ενεργειακές Καταστάσεις

Ένας κόμβος μπορεί να ρυθμιστεί σε τρεις διαφορετικές ενεργειακές καταστάσεις λειτουργίας .

- High Power (HP)

- Low Power (LP)

- Extended Low Power (ELP)

5.3.6 OTAP(Over the Air Programming)

Δυνατότητα επαναπρογραμματισμού των κόμβων από μακριά μέσω downstream μηνυμάτων που στέλνονται από τον κεντρικό σταθμό. Λειτουργεί μόνο για κόμβους σε HP κατάσταση.

5.3.7 Συγχρονισμός στον χρόνο (Time Synchronization)

Στην κατάσταση LP υποστηρίζεται συγχρονισμός του δικτύου με βάση μια γενική χρονική σταθερά (±1 ms). Με αυτό τον τρόπο συγχρονίζονται όχι μόνο τα μηνύματα αλλά και οι μετρήσεις των αισθητήρων. Αυτή η επιλογή όμως δεν είναι διαθέσιμη για τους κόμβους IRIS

5.3.8 Watch Dog

Το XMesh διαθέτει το component WDTM.nc το οποίο δίνει την δυνατότητα σε έναν κόμβο να κάνει ένα watchdog reset σε περίπτωση που χάσει πέντε RUM. Η ενεργοποίηση του watchdog timer αυξάνει το ρεύμα σε SLEEP κατάσταση κατά 15μA.

5.4 Διαχείριση Ενέργειας

Για το TinyOS η διαχείριση ενέργειας του επεξεργαστή γίνεται απευθείας από το λειτουργικό σύστημα. Ο προγραμματιστής δεν έχει άμεση πρόσβαση σε εντολές που μεταβάλλουν την ενεργειακή κατάσταση λειτουργίας του μικροελεγκτή. Αντί αυτού το ίδιο το TinyOS, μέσω μιας σειράς ελέγχων που εκτελούνται από τον scheduler και το module HPLPowerManagement[20], αποφασίζει πότε θα τεθεί ο επεξεργαστής σε κατάσταση SLEEP ή IDLE και ποια περιφερειακά θα μείνουν ενεργά, ανάλογα πάντα με τις επιλογές λειτουργίας που διαθέτει το μοντέλο του

επεξεργαστή. Από τη στιγμή που το θέμα της κατανάλωσης του επεξεργαστή είναι λυμένο, το βάρος πέφτει στη διαχείριση ενέργειας του RF πομποδέκτη, μέσω του πρωτοκόλλου επικοινωνίας. Το XMesh καθότι σχεδιασμένο με βάση το TinyOS, ακολουθεί παρόμοια στρατηγική διαχείρισης ενέργειας. Ο προγραμματιστής καλείται να επιλέξει ανάμεσα σε τρία είδη λειτουργίας που έχουν ήδη υλοποιήσει οι κατασκευαστές.

5.4.1 XMesh HP

Σε αυτή την κατάσταση ο πομποδέκτης είναι μόνιμα ενεργός. Η κατανάλωση κυμαίνεται από 15 ως 30 mA, ανάλογα με το μοντέλο. Ο κόμβος έχει δυνατότητα λήψης και αποστολής πακέτων ανά πάσα χρονική στιγμή και λειτουργεί σαν router. Τα πακέτα RouteUpdate και Health στέλνονται με υψηλούς ρυθμούς ανά 36", με αποτέλεσμα τη μείωση του χρόνου σχηματισμού του δικτύου και της εισόδου ενός νέου κόμβου σε αυτό.

5.4.2 XMesh LP

Η επικοινωνία των κόμβων μπορεί να γίνει με δυο τρόπους .Με συγχρονισμό (time synchronization) ή ασύγχρονα. Η διαδικασία επικοινωνίας των κόμβων όσον αφορά τη λήψη μηνυμάτων βασίζεται στην τεχνική preamble sampling ενώ όσον αφορά την αποστολή, χρησιμοποιείται ο μηχανισμός του πρωτοκόλλου CSMA – CA του 802.15.4 Η προεπιλογή για την wake – up sequence είναι 125 ms, που σημαίνει ότι ο κόμβος ξυπνά για να ακούσει 8 φορές ανά δευτερόλεπτο. Ο χρόνος αναμονής T μετά την αποστολή της wake – up sequence είναι 15 ms. Η συνολική διάρκεια του preamble σήματος είναι δηλαδή

$$\text{Preamble} = \text{wake – up sequence} + T = 140 \text{ ms}$$

Τα RouteUpdate πακέτα στέλνονται πολύ πιο αραιά απ'ότι στην κατάσταση HP και το bandwidth δικτύου είναι χαμηλότερο. Αυτό έχει ως αποτέλεσμα ο σχηματισμός του δικτύου να είναι αργότερος, κάτι τέτοιο όμως δεν έχει μεγάλη σημασία για δίκτυα που πρόκειται να λειτουργήσουν για πολύ καιρό χωρίς πολλές αλλαγές στη σύνθεση και διάταξή τους. Η βασική

κατανάλωση είναι 80 μA, ενώ για δίκτυο 50 κόμβων με αποστολή πακέτων ανά 3 λεπτά, ο μέσος όρος εκτιμάται στα 400 μA.

5.4.3 XMesh ELP

Ενδείκνυται μόνο για κόμβους που βρίσκονται στις άκρες του δικτύου (endpoints). Η διαδικασία λειτουργίας που ακολουθείται είναι η εξής : Μετά το άνοιγμα του ένας κόμβος παραμένει ON μέχρι να εισέλθει κανονικά στο δίκτυο. Αφού βρει τους γειτονικούς κόμβους αποθηκεύει τις διευθύνσεις τους, επιλέγει ένα ως γονέα με βάση την ισχύ του σήματος και αμέσως μετά θέτει τον πομποδέκτη σε κατάσταση SLEEP. Όταν ξυπνήσει για να μεταδώσει μετρήσεις θα δοκιμάσει πρώτα να στείλει στον γονέα που είχε επιλέξει αρχικά. Αν αποτύχει θα δοκιμάσει με κάποιον άλλο γειτονικό κόμβο που είχε αποθηκεύσει. Σε αυτή την κατάσταση ο κόμβος ξοδεύει ελάχιστα ποσά ενέργειας και θεωρητικά μπορεί να λειτουργεί για χρόνια. Ένα σημαντικό μειονέκτημα όμως είναι ότι οι σχεδιαστές δεν έχουν προβλέψει την ύπαρξη preamble σήματος, με αποτέλεσμα ένας κόμβος σε κατάσταση ELP να μπορεί να επικοινωνήσει μόνο με κόμβους σε κατάσταση HP και όχι με αυτούς που έχουν ρυθμιστεί σε LP.

Στην εικόνα 5.1 δείχνουμε τις παραμέτρους που παίρνει κάθε κατάσταση λειτουργίας που αναλύσαμε πιο πάνω.

Parameter	XMesh-HP	XMesh-LP	XMesh-ELP
Route Update Interval	36 sec.	360 sec.	36 sec if built using HP 360 sec if built using LP
Data Message Rate	10 sec, typ.	180 sec., typ.	N/A
Mesh formation time	2-3 times Route Update Interval for mesh with average of 2.5 hops		
Average current usage	20-30 mA	<250 μA [1] < 400 μA [2]	50 μA

Εικόνα 5.1 Παράμετροι για την κάθε λειτουργίας

5.5 Σχηματισμός Multi-Hop Δικτύου

Για το σχηματισμό του δικτύου εκτελούνται δυο παράλληλες διεργασίες. Μία είναι η Εκτίμηση Σήματος (Link Estimation) και η άλλη η Επιλογή Γονέα (Parent Selection).

5.5.1 Εκτίμηση Σήματος (Link Estimation)

Ένας κόμβος ακούγοντας το κανάλι παίρνει πληροφορίες για την κίνηση του δικτύου στη γειτονιά του. Αυτές τις πληροφορίες τις χρησιμοποιεί για τη δημιουργία ενός πίνακα όπου καταχωρεί τις διευθύνσεις των γειτόνων (neighbourhood table). Ταυτόχρονα ελέγχει πόσο καλά ακούει ένα γείτονα παρακολουθώντας τις τιμές μιας μεταβλητής στην κεφαλίδα των πακέτων. Η μεταβλητή αυτή περιέχει κάθε φορά ένα διαφορετικό αύξοντα αριθμό (sequence number). Με βάση αυτές τις τιμές γίνεται η εκτίμηση μέσω ενός αλγορίθμου EWMA (Exponentially Weighted Moving Average)[25]. Πιο συγκεκριμένα από τα sequence numbers που λαμβάνονται γίνεται γνωστό πόσα πακέτα λήφθηκαν κανονικά (received) και πόσα όχι (missed). Στη συνέχεια ο υπολογισμός της εκτίμησης λήψης (RE) γίνεται εφαρμόζοντας τον αλγόριθμο EWMA στο ποσοστό των ληφθέντων πακέτων, οπότε και προκύπτει ότι

$$\text{Received\_percentage} = 255*\text{received}/(\text{received}+\text{missed})$$
$$\text{RE} = (1-a)*\text{RE} + a*\text{Received\_percentage}$$

όπου a είναι ο παράγοντας EWMA με τιμές από 0 ως 1, ενώ το 255 χρησιμοποιείται για κανονικοποίηση έτσι ώστε το RE να κυμαίνεται στο [0,255] . Το μέγεθος του neighborhood table ορίζεται 16 για τους κόμβους του δικτύου και 40 για τον κόμβο του κεντρικού σταθμού. Αν κάποιος κόμβος βρει παραπάνω από 16 γείτονες θα διαγράψει αυτούς με την χαμηλότερη ποιότητα εκπομπής από τον πίνακα.

5.5.2 Επιλογή Γονέα (Parent Selection)

Αφού ο κόμβος ανιχνεύσει τους γείτονες και σχηματίσει το neighbor table, επιλέγει μέσα από αυτόν ένα γονέα με κριτήριο το ελάχιστο κόστος επικοινωνίας. Ένας κόμβος μπορεί να είναι υποψήφιος γονέας αν πληρεί τις παρακάτω προϋποθέσεις :

- Να έχει ήδη εισέλθει στο δίκτυο .

- Να μην υπήρξε παιδί (child node) του συγκεκριμένου κόμβου για τα τρία τελευταία RUI (Route Update Intervals), έτσι ώστε να αποφεύγονται οι κύκλοι στον σχηματισμό δικτύου.

- Να μην βρίσκεται σε κατάσταση ELP.

5.5.3 Route Update Messages (RUM)

Το τελικό στάδιο σχηματισμού του δικτύου επιτυγχάνεται με την περιοδική εκπομπή από όλους τους κόμβους, μηνυμάτων τα οποία περιέχουν πληροφορίες για τις διαθέσιμες διαδρομές στο δίκτυο. Τα μηνύματα αυτά ονομάζονται Route Update Messages (RUM) και το διάστημα που μεσολαβεί μεταξύ δυο διαδοχικών εκπομπών τους Route Update Interval (RUI). Η τιμή του RUI και η δομή των RUMs είναι κοινή για όλους τους κόμβους του δικτύου καθώς τα έχουμε παρουσιάσει στην εικόνα 5.1 Οι πληροφορίες που μεταφέρονται από τα RUM είναι :

- Ταυτότητα Γονέα (Parent ID) : Αν ο κόμβος δεν έχει μπει ακόμα στο δίκτυο η τιμή αυτής της μεταβλητής είναι 0xFFFF.

- Κόστος : Ενημερώνει τους γειτονικούς κόμβους ποιο είναι το κόστος αποστολής ενός πακέτου στον κεντρικό σταθμό.

- Hop Count : Ο αριθμός των hops που απέχει ο κόμβος από τον κεντρικό σταθμό.

Οι πληροφορίες αυτές καθορίζονται από μεταβλητές όπως οι :

RE : Αναφέρθηκε στην παράγραφο 5.5.1

SE (send estimation): Εκτίμηση εκπομπής. Όταν ένας κόμβος μεταδίδει ένα RUM, ενσωματώνει σε αυτό και την τιμή RE. Οι κόμβοι που λαμβάνουν το μήνυμα, με βάση την RE υπολογίζουν την SE.

LC (Link Cost) : Το κόστος αποστολής πακέτου στον κεντρικό σταθμό από ένα συγκεκριμένο γείτονα. Αν οι τιμές των RE και SE είναι γνωστές, το LC υπολογίζεται από τη σχέση

$$LC = (1<<18)/(SE \cdot RE)$$

NC (Neighbor Cost) : Κόστος αποστολής πακέτου σε γείτονα.

OC : Κόστος αποστολής πακέτου από τον κόμβο στον κεντρικό σταθμό. Υπολογίζεται από τη σχέση : $OC = LC + NC$
και αποτελεί το βασικότερο κριτήριο για την επιλογή του γονέα.

5.6 Αποστολή και Λήψη Πακέτων

5.6.1 XMesh Messaging API

Όπως αναφέραμε και στην αρχή του κεφαλαίου στο XMesh τα μηνύματα μπορούν να μεταδίδονται UPSTREAM (από κόμβο προς τη βάση) ή DOWNSTREAM (από τη βάση προς κόμβο). Η DOWNSTREAM διαδρομή είναι πάντα ίδια με την προηγούμενη UPSTREAM, γι'άυτο και για να επιτευχθεί DOWNSTREAM επικοινωνία με έναν κόμβο αυτός θα πρέπει να έχει στείλει πιο πριν τουλάχιστον ένα μήνυμα στη βάση. Τα μηνύματα μπορούν να σταλούν με δυο επιλογές QoS (Quality of Service):

- Link Level Acknowledgement

Αφορά την επικοινωνία μεταξύ δυο γειτονικών κόμβων. Ο αποστολέας ξαναστέλνει το μήνυμα αν δεν λάβει σήμα ACK από τον γειτονικό λήπτη. Η επιβεβαίωση όμως είναι μόνο τοπική. Έτσι για ένα multi – hop μήνυμα δεν είναι εγγυημένη η άφιξη του στον τελικό προορισμό. Χρησιμοποιώντας την LLA επιλογή εξοικονομείται ενέργεια και είναι χρήσιμη για εφαρμογές που δεν απαιτούν 100% παράδοση των δεδομένων.

- End to End Acknowledgement

Σε αυτή την περίπτωση εκτός από το τοπικό ACK, στέλνεται και άλλο ένα από τον τελικό παραλήπτη του multi – hop μηνύματος. Σε αντίθεση με πριν, αν το EtEAck δεν φτάσει, η επιλογή για το αν το μήνυμα θα ξαναμεταδοθεί δεν γίνεται αυτόματα αλλά αφήνεται στον χρήστη. Προφανώς η εγγυημένη μετάδοση έχει ως αντίτιμο την αύξηση στην κατανάλωση ενέργειας και bandwidth.

Το XMesh παρέχει ένα σύνολο από interfaces τα οποία ο χρήστης μπορεί να συνδέσει με τα components του TinyOS, μέσω της διαδικασίας wiring, έτσι ώστε να καλεί τις εντολές τους στην εφαρμογή του. Τα παραθέτουμε ονομαστικά και με μια περιληπτική περιγραφή για το καθένα.

5.6.1.1 MhopSend Interface

Διαθέτει τις commands getBuffer(msg, length),send(dest, mode, msg, length) και το event sendDone(msg, success). Με την getBuffer(msg, length) παρέχεται ένας δείκτης (msg) σε ένα τυποποιημένο buffer στο οποίο θα αποθηκευτεί το μήνυμα και ταυτόχρονα αρχικοποιούνται κάποιες τιμές στις κεφαλίδες. Η send χρησιμοποιείται για την αποστολή του μηνύματος. Τα ορίσματα της περιγράφονται στην εικόνα 5.4. Βασική προϋπόθεση είναι πριν την κλήση της send να έχει ήδη κληθεί η getBuffer έτσι ώστε να έχει οριστεί το buffer που θα περιέχει το μήνυμα.

Parameter	Description
dest	Destination Mote_Id for the message. (Usually BASE_STATION_ADDRESS for upstream messages)
mode	MODE_UPSTREAM_ACK: upstream with return acknowledge, MODE_UPSTREAM: upstream no return acknowledge, MODE_DOWNSTREAM_ACK: downstream with return acknowledge MODE_DOWNSTREAM: downstream no return acknowledge MODE_ONE_HOP_BROADCAST: one hop broadcast.
pMsg	pointer to the buffer to be sent
length	The length of the data buffer sent using this component. **This must be less than or equal to the maximum length provided by getBuffer()**
return	Whether the send request was successful: SUCCESS means a sendDone() event will be signaled later, FAIL means one will not. This does not mean that the destination mote received the message, only that the message was transmitted.

Εικόνα 5.4 Ορίσματα της εντολής MhopSend.Send

Το event sendDone ενεργοποιείται όταν η αποστολή μηνύματος μέσω της send ολοκληρωθεί. Αν ληφθεί Link Level Ack επιστρέφει την τιμή SUCCESS. Αν όχι η αποστολή επιλαμβάνεται μέχρι και οχτώ φορές. Σε περίπτωση αποτυχίας των πρώτων έξι, την έβδομη γίνεται επιλογή νέου γονέα. Αν αποτύχει και αυτή, την όγδοη το μήνυμα στέλνεται με διεύθυνση γονέα 0xFFFF, που σημαίνει ότι μπορεί να το παραλάβει οποιασδήποτε κόμβος βρίσκεται κοντά. Μετά απ'αυτό επιστρέφει την τιμή FAIL καθώς το μήνυμα κατά μεγάλο ποσοστό δεν παραδόθηκε πουθενά.

5.6.1.2 Receive και ReceiveAck Interface

Δυο παρόμοια interface που επιτρέπουν σε κόμβους endpoints να λαμβάνουν μηνύματα δεδομένων ή ACK αντίστοιχα. Παρέχουν τα events receive(msg, payload, payLoadLen) και receiveAck (msg, payload, payLoadLen). Η παράμετρος msg είναι δείκτης σε buffer όπου αποθηκεύεται το μήνυμα, η payLoad δείκτης στο τμήμα του buffer όπου βρίσκονται τα δεδομένα και στην payLoadLen αποθηκεύεται το μέγεθος σε bytes των δεδομένων. Στην περίπτωση ACK μηνύματος βέβαια, η payLoad δεν περιέχει δεδομένα αλλά την διεύθυνση του αποστολέα και τον τύπο του μηνύματος.

5.6.1.3 Intercept και Snoop Interface

Το interface Intercept διαθέτει το event intercept(msg, payload, payLoadLen). Για τις παραμέτρους ισχύει ότι και για το Receive. Ενεργοποιείται όταν ένα multi – hop μήνυμα φτάσει στον κόμβο. Αν μετά την εκτέλεση του επιστρέψει την τιμή SUCCESS, το μήνυμα θα προωθηθεί στον επόμενο κόμβο της διαδρομής. Αν επιστρέψει την τιμή FAIL, το μήνυμα δεν θα μεταδοθεί. Με αυτόν το τρόπο δίνεται η δυνατότητα στον χρήστη να παρέμβει στην προώθηση των multi – hop μηνυμάτων, όταν αυτά καταφτάνουν σε ενδιάμεσους κόμβους, με κριτήριο το περιεχόμενο τους. Το Snoop λειτουργεί με τον ίδιο ακριβώς τρόπο, με την διαφορά ότι μπορεί να διαχειριστεί όχι μόνο τα μηνυματα που προοριζονται για τον κομβο, αλλα και οσα εμφανιζονται γενικως στο καναλι.

5.6.1.4 PromisciousSniff Interface

Το PromisciousSniff interface δίνει τη δυνατότητα σε έναν κόμβο λαμβάνει οποιοδήποτε μήνυμα εμφανίζεται στο κανάλι ανεξαρτήτως χαρακτηριστικών. Είναι χρήσιμο μόνο για δοκιμαστικές εφαρμογές που απλά επιτηρούν τη λειτουργία ενός δικτύου.

5.6.1.5 RouteControl Interface

Τέλος το RouteControl interface αποτελεί ένα πολύ χρήσιμο εργαλείο για τη συλλογή πληροφοριών που αφορούν τη δομή και τη λειτουργία του δικτύου στην περιοχή του κόμβου που το χρησιμοποιεί. Διαθέτει έξι commands τα οποία τα αναφέρουμε παρακάτω:

- getParent(): Επιστρέφει την διεύθυνση του κόμβου γονέα.

- getDepth(): Επιστρέφει τον αριθμό των hops που απέχει ο κόμβος από τη βάση.

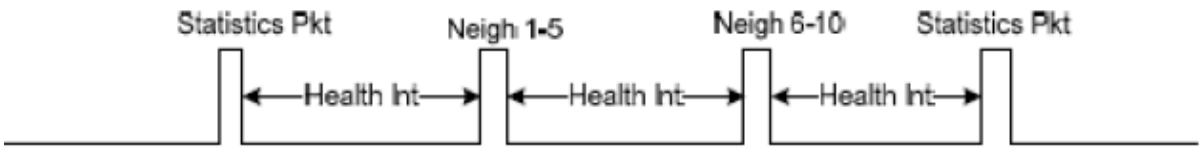
- getSender(msg): Επιστρέφει την διεύθυνση του κόμβου που έστειλε το μήνυμα.

- getOccupancy(): Επιστρέφει το πλήθος των μηνυμάτων που είναι σε αναμονή για προώθηση από τον κόμβο. Ενδεικτικό του πόσο απασχολημένος είναι ο κόμβος.

- getQuality(type): Για type = 1 επιστρέφει την τιμή SE και για type = 2 επιστρέφει την RE.

5.6.2 Health πακέτα

Υπάρχουν δυο τύποι health πακέτων: τα statistics health και τα neighbor health. Τα πρώτα περιλαμβάνουν στατιστικά δεδομένα σχετικά με τα πακέτα που στέλνει ο κόμβος (πλήθος προωθημένων, πλήθος απεσταλμένων, αριθμός επανεκπομπών κ.α.) καθώς και την τάση της μπαταρίας που τον τροφοδοτεί. Τα neighbor health περιέχουν πληροφορίες για την «γειτονιά» του κόμβου. Κάθε neighbor health πακέτο, μπορεί να περιέχει μέχρι και πέντε id γειτονικών κόμβων μαζί με δεδομένα που αφορούν την ποιότητα σύνδεσης με αυτούς (LQI) και την απόσταση από τον κεντρικό σταθμό (hop count). Η αποστολή health πακέτων γίνεται περιοδικά με περίοδο που ορίζεται από τον χρήστη. Η αποστολή statistics health πακέτων εναλλάσσεται με την αποστολή neighbor health πακέτων. Αν ο κόμβος έχει παραπάνω από πέντε γείτονες στέλνονται δυο συνεχόμενα neighbor πακέτα, όπως φαίνεται στην εικόνα 5.5 .



Εικόνα 5.5 Αποστολή πακέτων health

Κεφάλαιο 6
Απαιτήσεις και Σχεδιασμός Συστήματος

Σε αυτό το κεφάλαιο θα αναφέρουμε τα επίπεδα στα οποία χωρίσαμε τον σχεδιασμό του συστήματός μας και θα αναφερθούμε αναλυτικά στο κάθε ένα ξεχωριστά. Χωρίσαμε τον σχεδιασμό σε 3 επίπεδα. Στο πρώτο επίπεδο σχεδιάσαμε το λογισμικό που θα εγκατασταθεί τόσο στις μονάδες που απαρτίζουν το ασύρματο δίκτυο όσο και τη μονάδα-βάση που θα είναι συνδεδεμένη στον υπολογιστή. Στο δεύτερο επίπεδο σχεδιάσαμε τη βάση δεδομένων του ασύρματου δικτύου μας και στο τρίτο επίπεδο σχεδιάσαμε τον ιστότοπο διαχείρισης του ασύρματου δικτύου αισθητήρων.

6.1 Επίπεδο 1: Απαιτήσεις και Σχεδιασμός του λογισμικού των μονάδων του ασύρματου δικτύου αισθητήρων

Το ασύρματο δίκτυο αισθητήρων θα αποτελείται από κόμβους οι οποίοι έχουν αναλάβει να εκτελούν τις μετρήσεις που χρειαζόμαστε και από έναν σταθμό-βάσης στον οποίο στέλνουνε τις μετρήσεις οι μονάδες. Ο σταθμός βάσης στην συνέχεια θα μεταφέρει τις πληροφορίες στην βάση δεδομένων. Σε αυτό το σημείο γίνεται αντιληπτό ότι θα πρέπει να σχεδιαστούν δύο λογισμικά. Ένα λογισμικό που θα εκτελείτε στους κόμβους του δικτύου και ένα λογισμικό που θα εκτελείται στον σταθμό-βάσης.

6.1.1 Λογισμικό Κόμβων

Το λογισμικό που θα εκτελείται στους κόμβους του δικτύου θα πρέπει να πληροί κάποιες προϋποθέσεις:

- Σταθερή λειτουργική συμπεριφορά κόμβων

  Οι κόμβοι θα πρέπει να έχουν την προβλεπόμενη λειτουργική συμπεριφορά σε διάφορες περιπτώσεις. Σε αντίθετη περίπτωση η διαχείριση τους και η διόρθωση του συστήματος σε περίπτωση εντοπισμού λάθους θα είναι εξαιρετικά μια δύσκολη και επίπονη διαδικασία.

- Κατανάλωση ενέργειας

  Η διάρκεια ζωής του δικτύου είναι ένα από τα πιο σημαντικά σημεία της ανάπτυξης του λογισμικού. Εφόσον οι αισθητήρες παίρνουν ενέργεια από μπαταρίες η κατανάλωση ενέργειας θα πρέπει να είναι όσο το δυνατόν πιο χαμηλή.

- Επικοινωνία

Ένας κόμβος θα πρέπει να μπορεί να επικοινωνεί με τους γειτονικούς κόμβους είτε για την αποστολή των δεδομένων σε αυτούς και αυτοί με την σειρά τους στον σταθμό-βάση είτε για την αποστολή μηνυμάτων καλής λειτουργίας.

- Επεκτασιμότητα

Η επεκτασιμότητα του δικτύου είναι απαραίτητη καθώς σε κάθε δίκτυο θα πρέπει να δίνεται η δυνατότητα προσθήκης επιπλέον αισθητήρων. Θα πρέπει οπότε ο κάθε κόμβος να μπορεί να ανιχνεύει την παρουσία καινούριων κόμβων.

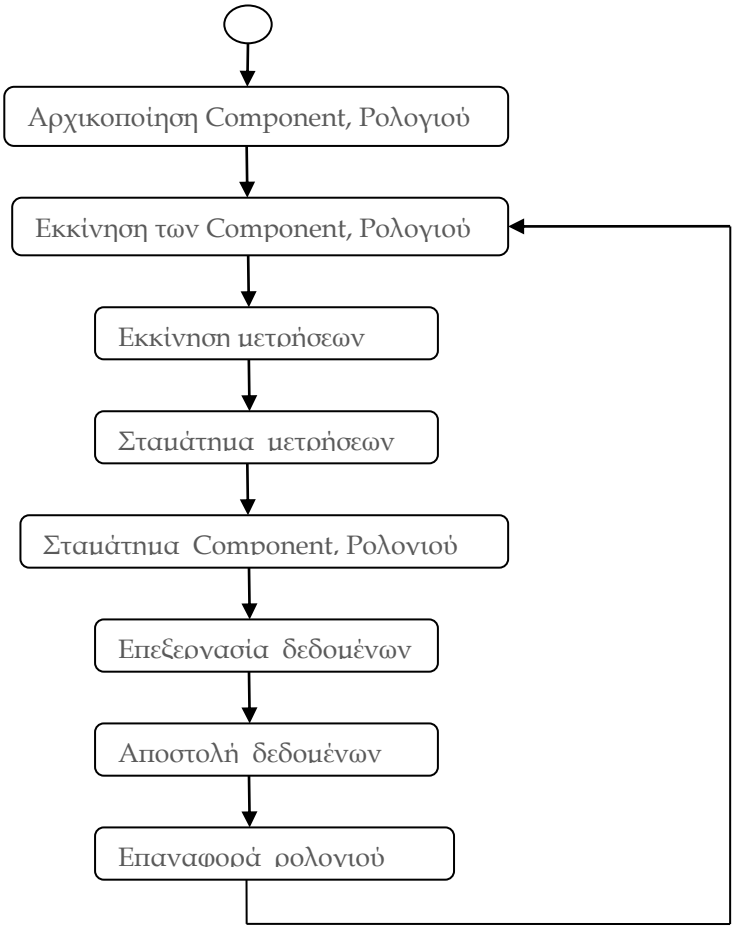
- Είδος Μετρήσεων

Οι κόμβοι θα πρέπει να είναι σε θέση να κάνουν διαφορετικές μετρήσεις ανάλογα με την πλακέτα αισθητήρων που έχουμε επιλέξει και του μεγέθους που θέλουμε να μετρήσουμε

- Ρυθμός μετρήσεων

Οι κόμβοι θα πρέπει να είναι σε θέση να αλλάζουν τον ρυθμό μετρήσεων από τους αισθητήρες έτσι ώστε στην περίπτωση που χρειαστεί κάποια αλλαγή να μπορεί να γίνει εύκολα.

Στην εικόνα 6.1 φαίνεται το διάγραμμα δραστηριοτήτων του κόμβου-αισθητήρα



Εικόνα 6.1 Διάγραμμα δραστηριοτήτων του κόμβου-αισθητήρων

Συγκεκριμένα χρησιμοποιώντας την πλακέτα αισθητήρων MDA100 θα πρέπει να γίνονται μετρήσεις θερμοκρασίας και φωτός. Επίσης χρησιμοποιώντας την πλακέτα αισθητήρων MDA300 σε συνδυασμό με τον αισθητήρα EC-10 θα πρέπει να γίνονται μετρήσεις που να αφορούν την υγρασία εδάφους. Το διάγραμμα δραστηριοτήτων ανάμεσα στο MDA100 και στο MDA300 διαφέρει μόνο στα Components που χρησιμοποιούμε . Η επιχειρησιακή λογική είναι ίδια.

6.1.2 Λογισμικό σταθμού-βάσης

Το λογισμικό θα χρησιμοποιείται από ένα mote το οποίο θα είναι συνδεδεμένο στον υπολογιστή και θα έχει τον ρόλο του σταθμού-βάσης. Οι απαιτήσεις του είναι :

- Συνεχή λειτουργία

Ο σταθμός-βάσης θα πρέπει να λειτουργεί συνέχεια και να είναι σε κατάσταση αναμονής ώστε να μπορεί να λαμβάνει τα μηνύματα που στέλνονται από τον κάθε κόμβο ξεχωριστά και μάλιστα και σε διαφορετικές στιγμές.

- Αποτύπωση πληροφορίας

Ο σταθμός-βάσης θα πρέπει να είναι σε θέση να αποτυπώσει τα δεδομένα που παίρνει από τους κόμβους αισθητήρων στην οθόνη.

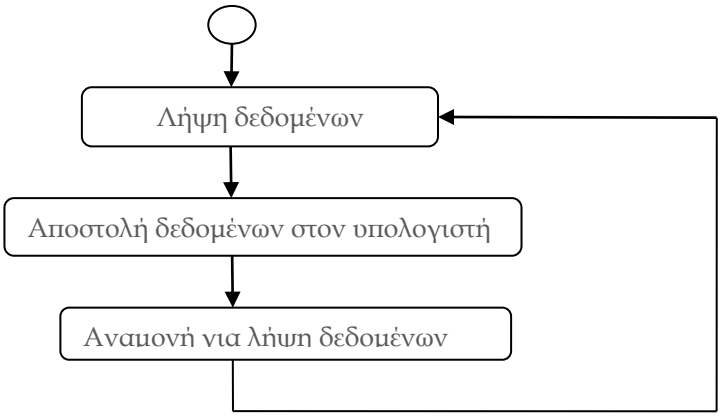
- Αποθήκευση πληροφορίας

Ο σταθμός-βάση θα πρέπει να είναι σε θέση να αποθηκεύσει τα δεδομένα που παίρνει από τους κόμβους αισθητήρων σε μία βάση δεδομένων για την περεταίρω επεξεργασία τους.

- Σύνδεση του σταθμού-βάσης στη θύρα USB του υπολογιστή

Ο σταθμός-βάσης θα πρέπει να μπορεί να συνδεθεί επιτυχώς στη θύρα USB του υπολογιστή.

Στην εικόνα 6.2 δείχνουμε το διάγραμμα δραστηριοτήτων του σταθμού-βάσης



Εικόνα 6.2 Διάγραμμα δραστηριοτήτων του σταθμού-βάσης

6.2 Επίπεδο 2: Απαιτήσεις και Σχεδιασμός της Βάσης Δεδομένων

Το δεύτερο επίπεδο αφορά τον σχεδιασμό της βάσης δεδομένων η οποία αποτελεί ένα πολύ σημαντικό κομμάτι της υλοποίησης του δικτύου μας. Στη βάση δεδομένων θα αποθηκεύονται όλα τα δεδομένα που συλλέγονται από τους αισθητήρες. Επίσης η βάση δεδομένων θα είναι και η πηγή από την οποία θα πάρουμε τα δεδομένα για να τα επεξεργαστούμε περεταίρω.

6.2.1 Επικοινωνία σταθμού-βάσης με βάση δεδομένων

Για την επικοινωνία του σταθμού-βάσης με την βάση δεδομένων χρησιμοποιείτε ένα εργαλείο σειριακής προώθησης το οποίο επιτρέπει τη μεταφορά πακέτων από τη θύρα USB στην οποία έχει συνδεθεί ο σταθμός-βάσης με τον υπολογιστή. Αυτός θα είναι και ο τρόπος μεταφοράς των μετρήσεων στον υπολογιστή και μετέπειτα στη βάση δεδομένων.

6.2.2 Απαιτήσεις βάσης δεδομένων

Στη βάση δεδομένων θα πρέπει να αποθηκεύονται τα δεδομένα τα οποία στέλνονται στον σταθμό-βάσης και τα οποία περιέχουν πληροφορίες για τις μετρήσεις του αισθητήρα MDA100, τις μετρήσεις του αισθητήρα MDA300 καθώς και πακέτα και των δύο αισθητήρων τα οποία περιέχουν πληροφορίες όσον αφορά την κατάστασή τους (health packets). Πιο συγκεκριμένα θα πρέπει ο κάθε κόμβος-αισθητήρας να έχει ένα αναγνωριστικό καθώς και να μπορεί να ομαδοποιηθεί μαζί με άλλους κόμβους-αισθητήρες είτε λόγω γεωγραφικής θέσης είτε λόγω είδους μέτρησης. Επίσης θα πρέπει να παρέχει πληροφορίες για την ώρα που έγινε η αποστολή της εκάστοτε μέτρησης. Στη συνέχεια, ο κάθε κόμβος-αισθητήρας θα στέλνει ένα πακέτο το οποίο θα περιλαμβάνει την κατάσταση του δικτύου, αξιοποιώντας τις δυνατότητες του XMesh.

6.2.3 Πίνακες βάσης δεδομένων

Ο πίνακας στον οποίο γίνεται η αποθήκευση των μετρήσεων του αισθητήρα MDA100 ορίστικε ως mda100_mesurments. Παρακάτω παραθέτουμε τον πίνακα 6.1 ο οποίος περιέχει μια περιγραφή των πεδίων του.

Όνομα στήλης	**Τύπος πεδίου**	**Μήκος**	**Περιγραφή**
result_time	Timestamp without time zone	8 bytes	Χρόνος αποθήκευσης του πακέτου στη βάση
nodeid	integer	4 bytes	Μοναδικός αριθμός αισθητήρα
parent	integer	4 bytes	Μοναδικός αριθμός πατέρα - αισθητήρα
voltage	integer	4 bytes	Τιμή τάσης
temp	integer	4 bytes	Τιμή θερμοκρασίας
light	integer	4 bytes	Τιμή έντασης φωτός

Πίνακας 6.1: Ο πίνακας mda100_mesurments

Ο πίνακας στον οποίο γίνεται η αποθήκευση των μετρήσεων του αισθητήρα MDA300 ορίστηκε ως mda300_mesurments. Παρακάτω παραθέτουμε τον πίνακα 6.2 ο οποίος περιέχει μια περιγραφή των πεδίων του.

Όνομα στήλης	**Τύπος πεδίου**	**Μήκος**	**Περιγραφή**
result_time	Timestamp without time zone	8 bytes	Χρόνος αποθήκευσης του πακέτου στη βάση
nodeid	integer	4 bytes	Μοναδικός αριθμός αισθητήρα
parent	integer	4 bytes	Μοναδικός αριθμός πατέρα -αισθητήρα
voltage	integer	4 bytes	Τιμή τάσης
adc0	integer	4 bytes	Τιμή υγρασίας εδάφους από τον αισθητήρα EC-10

Πίνακας 6.2: Ο πίνακας mda300_mesurments

Ο πίνακας στον οποίο γίνεται η αποθήκευση των πακέτων που αναφέρονται στην κατάσταση των αισθητήρων MDA100 και MDA300 ορίστηκε ως node_health. Παρακάτω παραθέτουμε τον πίνακα 6.3 ο οποίος περιέχει μια περιγραφή των πεδίων του.

Όνομα στήλης	**Τύπος πεδίου**	**Μήκος**	**Περιγραφή**
result_time	Timestamp without time zone	8 bytes	Χρόνος αποθήκευσης του πακέτου στη βάση δεδομένων
nodeid	integer	4 bytes	Μοναδικός αριθμός αισθητήρα
health_pkts	integer	4 bytes	Συνολικά πακέτα υγείας
node_pkts	integer	4 bytes	Πακέτα υγείας του κάθε αισθητήρα
forwarded	integer	4 bytes	Αριθμός. προωθημένων πακέτων
dropped	integer	4 bytes	Αριθμός χαμένων πακέτων
retries	integer	4 bytes	Αριθμός επαναπροσπαθειών αποστολής πακέτων
battery	integer	4 bytes	Υπόλοιπο μπαταρίας
power_sum	integer	4 bytes	Πάντα 0. Υπάρχει για μελλοντική χρήση
board_id	integer	4 bytes	Μοναδικός αριθμός board
parent	integer	4 bytes	Μοναδικός αριθμός πατέρα -αισθητήρα
quality_tx	integer	4 bytes	Ποιότητα κόμβου – πατέρα λαμβάνοντας υπόψη τα packet collisions. Τιμές από 0 – 15 σε αντιστοιχία 0 – 100%
quality_rx	integer	4 bytes	Ποιότητα πατέρα - κόμβου λαμβάνοντας υπόψη τα packet collisions. Τιμές από 0 – 15 σε αντιστοιχία 0 – 100%

path_cost	integer	4 bytes	Εκτίμηση του αριθμού των αναγκαίων μεταδόσεων για αποστολή πακέτου από τον κόμβο στο base κόμβο. Path Cost = 4 * Estimated Transmission Number (ETX) Path Cost=210/(quality_tx * quality_rx) Η πιο χαμηλή πιθανή τιμή είναι 4: απαιτείται τουλάχιστον μία μετάδοση για αποστολή πακέτου από το node στο base station.
parent_rssi	integer	4 bytes	Received Signal Strength Indicator πατρικού κόμβου

Πίνακας 6.3: Ο πίνακας node_health

6.3 Επίπεδο 3: Απαιτήσεις και Σχεδιασμός του ιστότοπου

Η πρόσβαση στις πληροφορίες που συλλέγονται από το ασύρματο δίκτυο αισθητήρων θα γίνεται μέσω ενός ιστότοπου. Πιο συγκεκριμένα ο ιστότοπος θα πρέπει να πληροί τα εξής κριτήρια :

- Θα πρέπει να είναι απλός και κατανοητός στον χρήστη.
- Θα πρέπει να μπορεί να μας παρουσιάσει σε πραγματικό χρόνο τις μετρήσεις από τον κάθε αισθητήρα ξεχωριστά ανάλογα με το είδος της μέτρησης.

- Θα πρέπει να μπορεί να μας παρουσιάσει το ιστορικό των μετρήσεων από τον κάθε αισθητήρα ξεχωριστά ανάλογα με το είδος της μέτρησης.

- Θα έχει γραφική αναπαράσταση των αποτελεσμάτων για πιο εύκολη κατανόηση από τον κάθε αισθητήρα ξεχωριστά και ανάλογα με το είδος της μέτρησης.

- Θα πρέπει να μπορεί να μας παρουσιάσει τα πακέτα με την κατάσταση των αισθητήρων (health packets).

Κεφάλαιο 7
Υλοποίηση Συστήματος

Σε αυτό το κεφάλαιο αναφέρουμε τον τρόπο υλοποίησης των επιπέδων στα οποία χωρίσαμε τον σχεδιασμό του συστήματός μας και θα αναφερθούμε αναλυτικά στο κάθε ένα ξεχωριστά. Η υλοποίηση ακολουθεί τον τρόπο σχεδιασμού.. Στο πρώτο επίπεδο υλοποιήθηκε το λογισμικό που θα εγκατασταθεί τόσο στις μονάδες που απαρτίζουν το ασύρματο δίκτυο όσο και τη μονάδα-βάση που θα είναι συνδεδεμένη στον υπολογιστή. Στο δεύτερο επίπεδο υλοποιήθηκε η βάση δεδομένων του ασύρματου δικτύου μας και στο τρίτο επίπεδο υλοποιήθηκε ο ιστότοπος που επιτρέπει τη διαχείριση του ασύρματου δικτύου αισθητήρων.

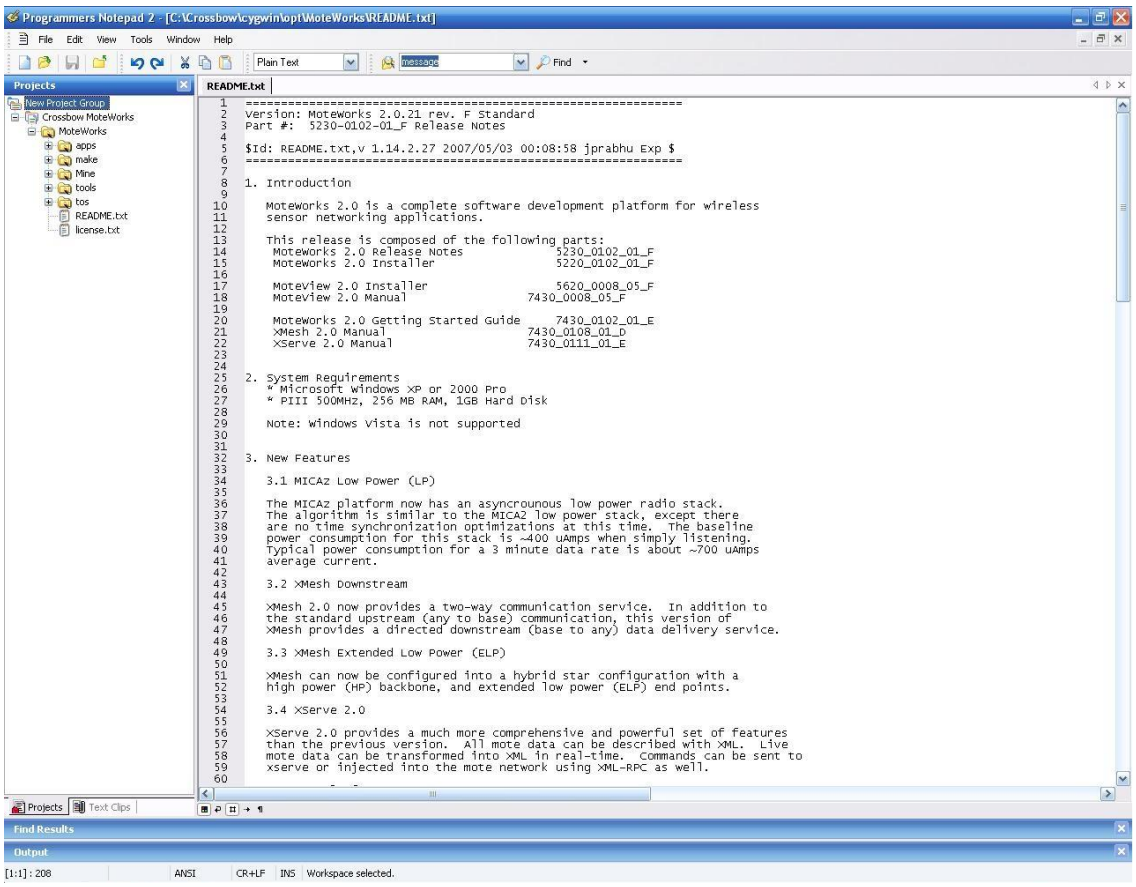
7.1 Υλοποίηση επιπέδου 1

7.1.1 Εργαλεία υλοποίησης επιπέδου 1

Όσον αφορά τον προγραμματισμό τόσο των κόμβων-αισθητήρων όσο και του σταθμού-βάσης χρησιμοποιήθηκε το εργαλείο Moteworks 2.0 το οποίο παρέχεται από την εταιρία crossbow από την οποία προμηθευτήκαμε τους κόμβους και τα αισθητήρια. Το συγκεκριμένο εργαλείο επιτρέπει την ανάπτυξη εφαρμογών για κόμβους-αισθητήρων και είναι ειδικά βελτιστοποιημένο για δίκτυα τα οποία λειτουργούν με χαμηλή κατανάλωση ενέργειας κάνοντας χρήση μπαταριών. Βασίζεται στο open source λειτουργικό σύστημα TinyOS και μας δίνει την δυνατότητα να χρησιμοποιηθεί η γλώσσα προγραμματισμού nesC για τον προγραμματισμό των κόμβων μας. Επίσης θα χρησιμοποιήθεί το πρωτόκολλο επικοινωνίας XMesh είτε για την επικοινωνία των κόμβων-αισθητήρων μεταξύ τους είτε για την επικοινωνία των κόμβων αισθητήρων με τον σταθμό-βάσης.

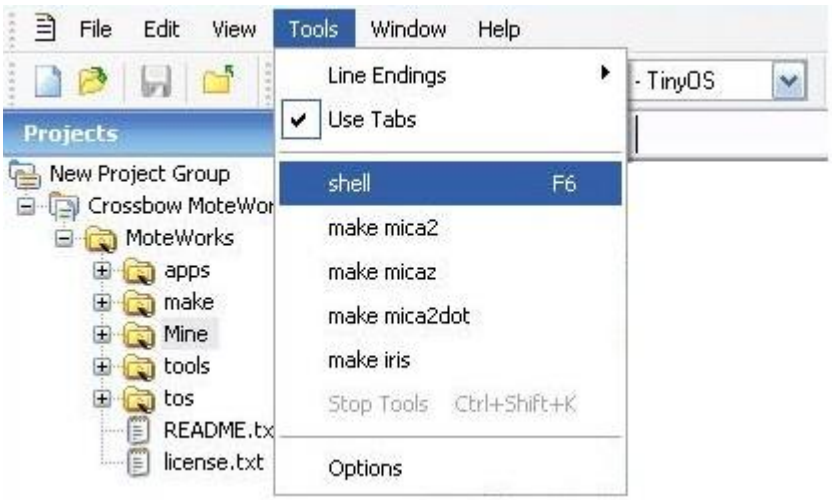
Ο σταθμός-βάσης όπως αναφέραμε στο κεφάλαιο 3, ο οποίος θα χρησιμοποιηθεί μαζί με τον υπολογιστή, αποτελείται από την προγραμματιστική πλακέτα MIB520CB και από ένα ασύρματο κόμβο Micaz και συνδέεται μέσω της θύρας USB. Η προγραμματιστική πλακέτα προσφέρει δύο θύρες, μία για τον προγραμματισμό των μονάδων και μια για την μεταφορά των δεδομένων στον υπολογιστή μέσω USB. Για να μπορέσει να γίνει εφικτή αυτή η επικοινωνία θα πρέπει να μπορεί να χρησιμοποιηθεί η θύρα USB σαν θύρα COM. Για να το επιτύχθεί αυτό χρησιμοποιήθηκαν οι V(irtual)C(om)P(ort) drivers τους οποίους μας παρέχει η FTDI chips.. Αυτοί οι drivers έχουν την ιδιότητα να μεταμορφώνουν την θύρα USB στην οποία έχει συνδεθεί η προγραμματιστική πλακέτα σε θύρα COM και η επικοινωνία μέσω των λειτουργικών γίνεται κανονικά σαν να είχαμε πρόσβαση σε κανονική θύρα COM. Αυτό θα έχει σαν αποτέλεσμα όποτε συνδέουμε την προγραμματιστική πλακέτα ο υπολογιστής μας να αναγνωρίζει τις δύο προαναφερόμενες θύρες COM. Στην δική μας περίπτωση αναγνωρίζει τις θύρες COM4 και COM5.

Για τον προγραμματισμό των κόμβων όπως αναφέρθηκε και προηγουμένως χρησιμοποιήθηκε το λογισμικό MoteWorks 2.0 το οποίο το έχει συνδυασθεί με το πρόγραμμα Programmers Notepad 2, ένα πρόγραμμα επεξεργασίας κειμένου το οποίο διαθέτει επισήμανση σύνταξης . Το λογισμικό αυτό μπορεί να το προμηθευτεί κανείς από την ιστοσελίδα της crossbow. Στην εικόνα 7.1 παρουσιάζουμε το συγκεκριμένο περιβάλλον.



Εικόνα 7.1: Το περιβάλλον του MoteWorks 2.0

Από την στιγμή που έχει συνδεθεί η συσκευή Micaz στον σταθμό-βάσης χρησιμοποιείτε το περιβάλλον του MoteWorks και γίνεται επιλογή του shell από το μενού Tools όπως φαίνεται στην εικόνα 7.2 και εμφανίζεται το παράθυρο της εικόνας 7.3.



Εικόνα 7.2: Το μενού Tools

Εικόνα 7.3: Το μενού Shell

Στο παράθυρο το οποίο εμφανίζεται μετά την συγκεκριμένη επιλογή εισάγεται η εντολή make micaz,<n> install mib520,com<m>.

Η εντολή make αναλαμβάνει να κάνει compile τον κώδικα που έχει γραφτεί στην γλώσσα προγραμματισμού NesC. Όπως φαίνεται η εντολή έχει δύο παραμέτρους. Η παράμετρος <n> χρησιμεύει για να δωθεί το id το οποίο θέλουμε να έχει το κάθε node. Η επόμενη παράμετρος είναι η com<m>. Για τον προγραμματισμό και μόνο των κόμβων μέσω της προγραμματιστικής πλακέτας ο κατασκευαστής απαιτεί να δηλωθεί η μικρότερη από τις δύο θύρες COM που έχει ο υπολογιστής μας. Στη δική μας περίπτωση είναι η θύρα COM4. Για την μετάδοση των πακέτων που λαμβάνει ο σταθμός-βάσης από τους κόμβους-αισθητήρες στον υπολογιστή θα χρησιμοποιηθεί η μεγαλύτερη από τις δύο θύρες δηλαδή η θύρα COM5.

Όταν εκτελείτε η συγκεκριμένη εντολή παρατηρούμε ότι το κόκκινο LED του σταθμού-βάσης ανάβει. Αυτό γίνεται γιατί εκείνη την ώρα μεταφέρεται το πρόγραμμα στο κόμβο. Εάν όλα έχουν πάει καλά τότε το κόκκινο LED σταματάει να είναι αναμμένο και ανάβει το πράσινο LED που είναι δίπλα του.

7.1.2 Λογισμικό κόμβων-αισθητήρων

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο των απαιτήσεων και του σχεδιασμού του λογισμικού οι κόμβοι-αισθητήρες θα πρέπει να είναι ικανοί να παίρνουν μετρήσεις ανά τακτά χρονικά διαστήματα από διάφορα φυσικά φαινόμενα και να μπορούν να τα στέλνουν στο σταθμό-βάσης. Παρακάτω ακολουθεί ο κώδικας για την υλοποίηση του προαναφερόμενης λειτουργίας.

7.1.2.1 Λογισμικό κόμβου-αισθητήρα MDA100

Το λογισμικό των εφαρμογών αποτελείτε από δύο αρχεία, το configuration και το αρχείο υλοποίησης της λειτουργικότητας της εφαρμογής μας. Δημιουργούμε στο MoteWorks έναν φάκελο με το όνομα MDA100 και μέσα σε αυτόν δημιουργούμε τα αρχεία MDA100.nc και MDA100M.nc .

Το αρχείο MDA100.nc αποτελεί το configuration αρχείο της εφαρμογής μας. Σε αυτό δηλώνονται όλα τα components που θα χρησιμοποιηθούν και επίσης πραγματοποιείται η σύνδεση των components με την εφαρμογή μας. Στον πίνακα 7.1 που παρουσιάζεται παρακάτω αναφέρουμε τα components που θα χρειαστούν για την ανάπτυξη της εφαρμογής:

Component	**Περιγραφή**
Main	Βασικό component το οποίο προσφέρει interface για την αρχικοποίηση του mote
TimerC	Μας δίνει την δυνατότητα να χρησιμοποιήσουμε χρονοδιακόπτη στην εφαρμογή μας
Voltage	Μας δίνει την δυνατότητα ελέγχου της μπαταρίας
Phototemp	Μας δίνει την δυνατότητα να χρησιμοποιήσουμε τον αισθητήρα θερμοκρασίας στην εφαρμογή μας
ADCC	Μας δίνει την δυνατότητα να χρησιμοποιήσουμε τον αισθητήρα φωτός στην εφαρμογή μας
GenericCommPromiscuous	Το βασικό συστατικό για τον έλεγχο των μονάδων
MULTIHOPROUTER	Είναι το βασικό συστατικό του πρωτοκόλλου XMesh
LedsC	Μας δίνει την δυνατότητα ελέγχου των leds του micaz

Bcast	Μας δίνει την δυνατότητα αποστολής πακέτων είτε σε κόμβους-αισθητήρες είτε στον σταθμό-βάσης

Πίνακας 7.1: Τα components του MDA100.nc

Στην εικόνα 7.4 που ακολουθεί παρακάτω δείχνουμε το πρόγραμμα του MDA100.nc και επεξηγούμε κάποια στοιχεία του

```
includes sensorboardApp;

configuration MDA100 {
}
implementation
{
  components Main,
             TimerC,
             Voltage, PhotoTemp, ADCC,
             GenericCommPromiscuous as Comm,
           MULTIHOPROUTER, MDA100M,LEDS_COMPONENT, XCommandC, Bcast;

  Main.StdControl -> MDA100M;
  Main.StdControl -> MULTIHOPROUTER.StdControl;
  Main.StdControl -> Comm;
  Main.StdControl -> TimerC;

  LEDS_WIRING(XMDA100M)

  // Wiring for Battery Ref
  MDA100M.BattControl -> Voltage;
  MDA100M.ADCBATT -> Voltage;

  MDA100M.TempControl -> PhotoTemp.TempStdControl;
  MDA100M.Temperature -> PhotoTemp.ExternalTempADC;

  MDA100M.PhotoControl -> PhotoTemp.PhotoStdControl;
  MDA100M.Light -> PhotoTemp.ExternalPhotoADC;

  MDA100M.Timer -> TimerC.Timer[unique("Timer")];

  // Wiring for broadcast commands.
  MDA100M.XCommand -> XCommandC;
  MDA100M.XEEControl -> XCommandC;

  // Wiring for RF mesh networking.
  MDA100M.RouteControl -> MULTIHOPROUTER;
  MDA100M.Send -> MULTIHOPROUTER.MhopSend[AM_XMULTIHOP_MSG];
  MULTIHOPROUTER.ReceiveMsg[AM_XMULTIHOP_MSG] ->Comm.ReceiveMsg[AM_XMULTIHOP_MSG];
  MDA100M.HealthMsgGet -> MULTIHOPROUTER;
  MDA100M.health_packet -> MULTIHOPROUTER.health_packet;
}
```

Εικόνα 7.4: Τα components του MDA100.nc

Αυτό που παρατηρούμε στην αρχή είναι η δήλωση “includes sensorboardApp;” . Εκεί βρίσκεται η δομή των μηνυμάτων που θα αποστέλλει ο κόμβος-αισθητήρας μας στον σταθμό-βάσης. Όπως

έχουμε αναφερθεί στα προηγούμενα κεφάλαια θα αποστέλλεται ένα μήνυμα που θα περιέχει τις μετρήσεις και ένα μήνυμα το οποίο θα περιέχει την κατάσταση του κόμβου-αισθητήρα το λεγόμενο health packet.

Επίσης παρατηρούμε την δήλωση "GenericCommPromiscuous as Comm". Αυτή η δήλωση μας επιτρέπει να δηλωθεί το GenericCommPromiscuous με το ψευδώνυμο Comm ώστε να γίνεται πιο εύχρηστη η χρήση της στον υπόλοιπο κώδικα.

Στη συνέχεια θα αναλύσουμε τον τρόπο λειτουργίας της εφαρμογής μας ο οποίος όπως έχουμε αναφέρει προδιαγράφεται στο αρχείο MDA100M.nc .

Αρχικά θα πρέπει να πραγματοποιηθεί ο συγχρονισμός του δικτύου. Όταν θέτουμε σε λειτουργία τον κόμβο στέλνεται ένα αίτημα συγχρονισμού στον σταθμό-βάσης και μπαίνει σε κατάσταση αναμονής. Όταν ο σταθμός-βάσης λαμβάνει σήμα από όλους τους κόμβους του δικτύου στέλνει το σήμα συγχρονισμού στους κόμβους οι οποίοι όταν το λάβουν ξεκινάνε το ρολόι τους και τις όποιες λειτουργίες έχει αναλάβει ο καθένας. Παρακάτω δείχνουμε αυτή τη διαδικασία συγχρονισμού.

```
event result_t XCommand.received(XCommandOp *opcode) {
switch (opcode->cmd) {
case XCOMMAND_WAKEUP:
if (sleeping) {
initialize();
call Timer.start(TIMER_REPEAT, timer_rate);
sleeping = FALSE;
} break;
```

Αμέσως μετά ξεκινάνε οι μετρήσεις από τον αισθητήρα. Αναφέραμε στον σχεδιασμό ότι με τον συγκεκριμένο αισθητήρα θα πρέπει να μετράμε θερμοκρασία και φως. Το μπλοκ κώδικα που κάνει την δειγματοληψία της θερμοκρασίας και του φωτός δίνονται παρακάτω:

```
async event result_t Temperature.dataReady(uint16_t data) {
atomic pack.xData.datap1.thermistor = data ;
atomic state = TEMP_DONE;
call Light.getData();
```

```
return SUCCESS;
}

async event result_t Light.dataReady(uint16_t data) {
atomic pack.xData.datap1.photo = data ;
post photostop();
post tempstop();
call ADC2.getData();
TOSH_uwait(100);
return SUCCESS;
}
```

Όταν ολοκληρωθούν οι μετρήσεις τότε ο κόμβος-αισθητήρας στέλνει τα δεδομένα στον σταθμό βάσης με την παρακάτω εντολή:

```
call SendBCAST.send(TOS_BCAST_ADDR, sizeof(XDataMsg),msg_ptr)
```

Η παραπάνω εντολή χρησιμοποιεί την παράμετρο TOS_BCAST_ADDR για να προσδιορίσει τον τρόπο αποστολής των πακέτων σε οποιοδήποτε κόμβο-αισθητήρα ή σταθμό-βάσης είναι μέσα στην εμβέλεια του.

Στην συνέχεια και για να εκτελέσουμε το λογισμικό στον κόμβο-αισθητήρα MDA100 χρησιμοποιούμε μέσω του MoteWorks την εντολή make micaz install,1 mib520,com4 ώστε να δηλώσουμε τον κόμβο-αισθητήρα με το id 1.

7.1.2.2 Λογισμικό κόμβου-αισθητήρα MDA300

Για το λογισμικό της εφαρμογής του κόμβου-αισθητήρα με το MDA300 ακολουθήθηκε η ίδια λογική όπως και στον κόμβο-αισθητήρα με το MDA100. Δημιουργήθηκε στο MoteWorks ένας φάκελος με το όνομα MDA300 και μέσα σε αυτόν δημιουργήθηκαν τα αρχεία MDA300.nc και MDA300M.nc .

Το αρχείο MDA300.nc αποτελεί το configuration αρχείο της εφαρμογής μας. Είναι το αρχείο που περιέχει τις συνδέσεις της εφαρμογής μας με τα components που θα χρησιμοποιηθούν. Στον πίνακα 7.2 που παρουσιάζεται παρακάτω αναφέρουμε τα components που θα χρειαστούν για την ανάπτυξη της εφαρμογής:

Component	**Περιγραφή**
Main	Βασικό component το οποίο προσφέρει interface για την αρχικοποίηση του mote
TimerC	Μας δίνει τη δυνατότητα να χρησιμοποιήσουμε χρονοδιακόπτη στην εφαρμογή μας
Voltage	Μας δίνει τη δυνατότητα ελέγχου της μπαταρίας
Phototemp	Μας δίνει τη δυνατότητα να χρησιμοποιήσουμε τον αισθητήρα θερμοκρασίας στην εφαρμογή μας
SamplerC	Μας παρέχει τη δυνατότητα ελέγχου από το αναλογικό κανάλι A0 που έχουμε συνδέσει το εξωτερικό αισθητήριο
GenericCommPromiscuous	Το βασικό συστατικό για τον έλεγχο των μονάδων
MULTIHOPROUTER	Είναι το βασικό συστατικό του πρωτοκόλλου XMesh
LedsC	Μας δίνει τη δυνατότητα ελέγχου των leds του micaz
Bcast	Μας δίνει τη δυνατότητα αποστολής πακέτων είτε σε κόμβους-αισθητήρες είτε στον σταθμό-βάση

Πίνακας 7.2: Τα components του MDA300.nc

Στην εικόνα 7.5 που ακολουθεί παρακάτω δείχνουμε το πρόγραμμα του MDA300.nc . Όπως αναφέραμε στο κεφάλαιο του σχεδιασμού η επιχειρησιακή λογική του προγράμματος είναι ίδια με την επιχειρησιακή λογική του MDA100 αλλά αλλάζουν οι μετρήσεις που πρέπει να κάνει ο κάθε κόμβος ανάλογα με τον αισθητήρα που έχει ενσωματωθεί. Στην περίπτωση μας το MDA300 έχει συνδεθεί τον αισθητήρα EC-10 στο Adc0 και για τον έλεγχό του χρησιμοποιείτε το ανάλογο component.

```
configuration MDA300 {
}

implementation {
    components Main,
    GenericCommPromiscuous as Comm,
    MULTIHOPROUTER,MDA300M,LEDS_COMPONENT,XCommandC,Bcast,SamplerC,TimerC;

    Main.StdControl -> MDA300M;
    Main.StdControl -> MULTIHOPROUTER.StdControl;
    Main.StdControl -> Comm;
    Main.StdControl -> TimerC;
    LEDS_WIRING(XMDA300M)
    MDA300M.Timer -> TimerC.Timer[unique("Timer")];

    //Sampler Communication

    MDA300M.SamplerControl -> SamplerC.SamplerControl;
    MDA300M.Sample -> SamplerC.Sample;
    MDA300M.PlugPlay -> SamplerC.PlugPlay;

    //relays

    MDA300M.relay_normally_closed -> SamplerC.relay_normally_closed;
    MDA300M.relay_normally_open -> SamplerC.relay_normally_open;

    // Wiring for broadcast commands.
    MDA300M.XCommand -> XCommandC;
    MDA300M.XEEControl -> XCommandC;

    // Wiring for RF mesh networking.
    MDA300M.RouteControl -> MULTIHOPROUTER;
    MDA300M.Send -> MULTIHOPROUTER.MhopSend[AM_XMULTIHOP_MSG];
    MULTIHOPROUTER.ReceiveMsg[AM_XMULTIHOP_MSG] ->Comm.ReceiveMsg[AM_XMULTIHOP_MSG];
    MDA300M.HealthMsgGet -> MULTIHOPROUTER.HealthMsgGet;
   XMDA300M.health_packet -> MULTIHOPROUTER.health_packet;
}
```

Εικόνα 7.5: Τα components του MDA300.nc

Το μπλοκ κώδικα που κάνει τη δειγματοληψία του εξωτερικού αισθητήρα EC-10 ο οποίος είναι συνδεδεμένος στο κανάλι A0 και εφόσον χρειάζεται 2,5V για να μπορέσει να λειτουργήσει είναι το εξής:

record[0] = call Sample.getSample(0,ANALOG,ANALOG_SAMPLING_TIME | EXCITATION_25 | DELAY_BEFORE_MEASUREMENT);

Η παραπάνω εντολή δέχεται για παράμετρο το κανάλι στο οποίο κάνουμε την δειγματοληψία, στην περίπτωσή μας το 0, τον τύπο του καναλιού δηλαδή εάν είναι το κανάλι Analog ή Digital καθώς και τον χρόνο δειγματοληψίας. Άλλες παράμετροι οι οποίοι είναι σημαντικοί είναι ο σκανδαλισμός του αισθητήρα για να μπορέσει να λειτουργήσει, στην περίπτωσή μας ακολουθώντας τις οδηγίες του κατασκευαστή χρειαζόμαστε 2,5V επιλέγουμε το EXCITATION_25 (μας παρέχονται και οι επιλογές EXCITATION_33 και EXCITATION_50) καθώς και η παράμετρος DELAY_BEFORE_MEASUREMENT ώστε να υπάρχει η απαραίτητη καθυστέρηση που χρειάζεται από το σκανδαλισμό του αισθητήρα μέχρι την μέτρηση.

Στη συνέχεια και για να εκτελσθεί το λογισμικό στον κόμβο-αισθητήρα MDA300 χρησιμοποιείτε μέσω του MoteWorks η εντολή "make micaz install,2 mib520,com4" ώστε να δηλωθεί ο κόμβος-αισθητήρας με το id 2.

7.1.3 Λογισμικό σταθμού-βάσης

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο των απαιτήσεων και του σχεδιασμού του λογισμικού ο σταθμός-βάση θα πρέπει να μπορεί να δέχεται δεδομένα από τους κόμβους και να τα προωθεί στην βάση δεδομένων. Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι θα χρησιμοποιεί το πρωτόκολλο XMesh για την επικοινωνία με τους κόμβους αισθητήρες.

Παρακάτω δίνεται το κομμάτι του κώδικα για το XMeshBaseStation.nc

```
includes sensorboardApp;

configuration XMeshBaseStation {
}
implementation
{
    components Main,
    MULTIHOPROUTER,XMeshBaseStationM,
    LEDS_COMPONENT,
    XCommandC;


    Main.StdControl -> XMeshBaseStationM;

    LEDS_WIRING(XMeshBaseStationM)
    XMeshBaseStationM.XCommand -> XCommandC;


    // Wiring for RF mesh networking.
    XMeshBaseStationM.RouteControl -> MULTIHOPROUTER;
}
```

Εικόνα 7.6: Τα components του XMeshBaseStation.nc

Παρατηρούμε ότι χρησιμοποιείται το component MULTIHOPROUTER και XcommandC για την αποστολή των δεδομένων και την επικοινωνία του δικτύου μας καθώς και τα Leds για να φαίνεται πότε λαμβάνει χώρα η επικοινωνία με τους κόμβους-αισθητήρες.

Στην επόμενη εικόνα βλέπουμε το Module αρχείο για το XMeshBaseStation

```
module XMeshBaseStationM {
    provides {
    interface StdControl;
    }
    uses {
    interface RouteControl;
    interface XCommand;
    interface Leds;
    }
}

implementation {
  command result_t StdControl.init() {
    call Leds.init();
    atomic{ TOS_LOCAL_ADDRESS = 0; }
    return SUCCESS;
  }
  command result_t StdControl.start(){
    return SUCCESS;
  }
  command result_t StdControl.stop() {
    return SUCCESS;
  }
  event result_t XCommand.received(XCommandOp *opcode) {

    switch (opcode->cmd) {

    case XCOMMAND_SLEEP:
        break;

    case XCOMMAND_WAKEUP:
      break;

    default:
      break;
    }
    return SUCCESS;
  }
}
```

Εικόνα 7.7: Το module του XMeshBaseStation.nc

Στην συνέχεια και για να εκτελσθεί το λογισμικό στον σταθμό-βάσης χρησιμοποιείτε μέσω του MoteWorks η εντολή "make micaz install,0 mib520,com4" ώστε να δηλωθεί ο σταθμός-βάσης με το id 0.

Επίσης αναφέραμε ότι θα πρέπει να μπορεί ο σταθμός-βάση να στέλνει τα δεδομένα στον υπολογιστή με τον οποίο είναι συνδεδεμένος είτε για να τα εμφανίσει στην οθόνη του υπολογιστή είτε για να τα αποθηκεύσει σε μία βάση δεδομένων για την περαιτέρω επεξεργασία τους. Τα προγράμματα που χρησιμοποιούνται για να «ακούνε» στην θύρα USB του υπολογιστή

αναφέρονται ως προωθητές σειριακής θύρας (Serial Port Forwarding). Στην περίπτωσή μας επιλέχτηκε να χρησιμοποιηθεί το πρόγραμμα Xserve.

Το Xserve χρησιμεύει ως η κύρια πύλη μεταξύ των ασύρματων δικτύων και των εφαρμογών που υλοποιούνται σε αυτά. Μας παρέχει υπηρεσίες δρομολόγησης δεδομένων από και προς το δίκτυο σε πραγματικό χρόνο. Οι υπηρεσίες που μας παρέχει είναι προσαρμόσιμες χρησιμοποιώντας XML αρχεία ρυθμίσεων και plugin modules. Οι χρήστες μπορούν να αλληλεπιδρούν με το Xserve με τη χρήση τερματικού και οι εφαρμογές μπορούν να έχουν πρόσβαση στο δίκτυο είτε άμεσα είτε μέσω ενός δυναμικού XML RPC command interface.

Για την εκκίνηση του Xserve και αφού έχουν προγραμματισθεί οι κόμβοι μας χρειάστηκε ένα τερματικό. Χρησιμοποιήθηκε το Cygwin που παρέχεται μαζί με το λογισμικό MoteWorks. Για να μπορέσουμε να δούμε τα δεδομένα που έφτασαν στον σταθμό-βάση πρέπει να χρησιμοποιηθεί η εντολή: Xserve –device=COM5.

Για την καταγραφή όμως των δεδομένων τα οποία λαμβάνει ο σταθμός-βάσης σε μια βάση δεδομένων θα πρέπει να χρησιμοποιηθούν κάποιες παράμετροι επιπλέον. Οι παράμετροι αυτοί φαίνονται στην παρακάτω εντολή και ακολουθεί μια σύντομη επεξήγηση:

Xserve –device=COM5 –db –dbname=mydb –dbuser=tele –dbpasswd=tiny

Με την συγκεκριμένη εντολή καθορίζεται στο Xserve ότι τα δεδομένα θα πρέπει να αποθηκευθούν στην βάση δεδομένων με το όνομα mydb και στην οποία θα έχει πρόσβαση με τα στοιχεία username: tele και password: tiny.

Από την στιγμή που το Xserve λάβει την παραπάνω εντολή εξετάζει τα XML αρχεία, τα οποία αποτελούν μέρος του συστήματος και έχουν παραμετροποιηθεί κατάλληλα, ώστε να κάνει τις απαιτούμενες ενέργειες για την αποθήκευση των δεδομένων. Στα XML αρχεία υπάρχουν τα DataSink όσον αφορά τα δεδομένα του MDA100 και τα δεδομένα του MDA300 καθώς και για τα health packets:

Για το MDA100:

```
<XDataSink name="Open Log Datasink">
      <XDSParam name="tablename_1" value="mda100_mesurments"/>
                  <XDSParam name="insertsql_1" value="INSERT into mda100_
mesurments      (result_time,nodeid,parent,voltage,temp,light)      values
```

```
(now(),%i,%i,%i,%i,%i)"/>                    <XDSParam
name="insertfields_1"          value="nodeid,parent,voltage,temp,light"/>
</XDataSink>
```

Για το MDA300:

```
<XDataSink name="Open Log Datasink">
      <XDSParam name="tablename_1" value="mda300_ mesurments "/>
                  <XDSParam name="insertsql_1" value="INSERT into mda300_
mesurments          (result_time,nodeid,parent,voltage,adc0)          values
(now(),%i,%i,%i,%i)"/>                    <XDSParam   name="insertfields_1"
value="nodeid,parent,voltage,adc0"/>       </XDataSink>
```

Για τα health packet:

```
<XDataSink name="Open Log Datasink">
      <XDSParam name="tablename_1" value="node_health"/>
      <XDSParam   name="insertsql_1"   value="INSERT   into   node_health
(result_time, nodeid, health_pkts, node_pkts, forwarded, dropped, retries,
battery, power_sum, board_id, parent, quality_tx, quality_rx, path_cost,
parent_rssi) values (now() ,%i, %i, %i, %i, %i, %i, %i, %i, %i, %i,
%i, %i, %i)"/>
      <XDSParam    name="insertfields_1"    value="nodeid,    health_pkts,
node_pkts, forwarded, dropped, retries, battery, power_sum, board_id,
parent, quality_rx, quality_tx, path_cost, parent_rssi"/>

</XDataSink>
```

7.2 Υλοποίηση επιπέδου 2

Για την υλοποίηση της βάσης δεδομένων χρησιμοποιήσαμε το Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) PostgreSQL. Το συγκεκριμένο ΣΔΒΔ είναι ανοιχτού λογισμικού και επιτρέπει τη δημιουργία αντικειμενοστεφών σχεσιακών βάσεων δεδομένων. Υπάρχει στην αγορά για περισσότερα από 15 χρόνια και χαρακτηρίζεται για την αξιοπιστία καθώς και για την ακεραιότητα των δεδομένων. Τρέχει σε όλα τα γνωστά λειτουργικά συστήματα

συμπεριλαμβανομένων των Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), και Windows. Επίσης υποστηρίζει την χρήση των primary keys, foreign keys, joins, views, triggers, και stored procedures .

Για τη δημιουργία της βάσης δεδομένων καθώς και για την διαχείρισή της χρησιμοποιήθηκε το λογισμικό PgAdmin III. Είναι ανοιχτού κώδικα και είναι το πιο δημοφιλές εργαλείο διαχείρισης και ανάπτυξης για την PostgreSQL. Μπορεί να χρησιμοποιηθεί σε πλατφόρμες Linux, FreeBSD, Solaris, Mac OSX και Windows. Το PgAdmin είναι σχεδιασμένο ώστε να ανταποκρίνεται στις ανάγκες όλων των χρηστών καθώς είναι ιδανικό για την εκτέλεση απλών SQL queries μέχρι και την ανάπτυξη σύνθετων βάσεων δεδομένων. Το γραφικό περιβάλλον του υποστηρίζει όλα τα χαρακτηριστικά της PostgreSQL ενώ περιέχει και ξεχωριστό γραφικό περιβάλλον για την εκτέλεση των SQL queries.

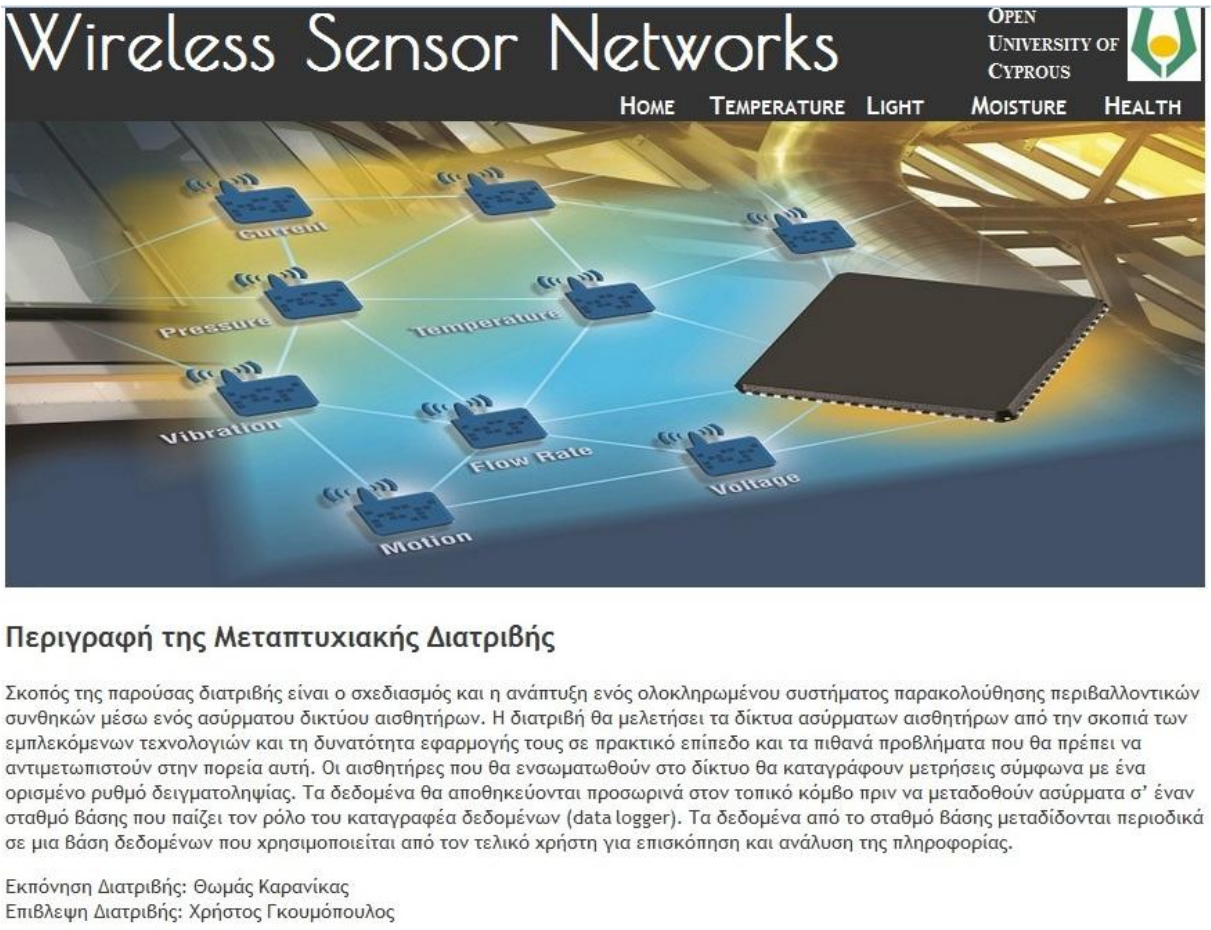
Τα SQL queries που χρησιμοποιήθηκαν για την δημιουργία της βάσης δεδομένων ήταν τα παρακάτω καθώς και για την εισαγωγή των δεδομένων είναι τα εξής:

```
CREATE TABLE mda100_ mesurments ( result_time timestamp without time zone, nodeid integer, parent integer,voltage integer, temp integer, light integer

INSERT into mda100_mesurments (result_time, nodeid, parent, voltage, temp, light) values (now(),%i,%i,%i,%i,%i)

CREATE TABLE mda300_ mesurments ( result_time timestamp without time zone, nodeid integer, parent integer,adc0 integer, voltage integer)

INSERT into mda300_mesurments (result_time,nodeid,parent,voltage,adc0) values (now(),%i,%i,%i,%i)

CREATE TABLE node_health ( result_time timestamp without time zone,epoch integer, nodeid integer, health_pkts integer, node_pkts integer, forwarded integer, dropped integer, retries integer, battery integer, power_sum integer,board_id integer, parent integer, quality_tx integer, quality_rx integer, path_cost integer, parent_rssi integer)

INSERT into node_health (result_time, nodeid, health_pkts, node_pkts, forwarded, dropped, retries, battery, power_sum, board_id, parent,
```

```
quality_tx, quality_rx, path_cost, parent_rssi) values (now() ,%i, %i, %i,
%i, %i, %i, %i, %i, %i, %i, %i, %i, %i, %i, %i)
```

7.3 Υλοποίηση επιπέδου 3

Για την υλοποίηση του ιστότοπου παρουσίασης των αποτελεσμάτων χρησιμοποιήθηκε η γλώσσα προγραμματισμού php και η γλώσσα HTML σε συνδυασμό με τη χρήση του CSS για την οπτική μορφοποίηση του ιστότοπου. Επίσης με την χρήση SQL queries παίρνονται τα επιθυμητά αποτελέσματα από τη βάση δεδομένων ώστε να παρουσιαστούν στην τελική μορφή στον χρήστη. Στην εικόνα 7.8 που ακολουθεί δείχνουμε τον βασική σελίδα του ιστότοπου.



Εικόνα 7.8: Η βασική σελίδα του ιστότοπου

Όπως φαίνεται στην εικόνα αποφασίσαμε να έχουμε απεικόνιση των μετρήσεων σε ξεχωριστή ιστοσελίδα ανά είδος μέτρησης. Έτσι δημιουργήθηκε ένα μενού στο οποίο αναθέσαμε μια ιστοσελίδα για θερμοκρασία, μία για τις μετρήσεις της έντασης του φωτός και μία για την υγρασία εδάφους. Δημιουργήθηκε επίσης μία ιστοσελίδα στην οποία αποτυπώνονται τα πιο σημαντικά χαρακτηριστικά του health packet το οποίο υποστηρίζει το πρωτόκολλο XMesh και έχει να κάνει με την κατάσταση του δικτύου. Σε κάθε ιστοσελίδα δίνεται η δυνατότητα στο χρήστη μέσω των αντίστοιχων κουμπιών να παρακολουθεί την τελευταία μέτρηση που έχει κάνει ο συγκεκριμένος αισθητήρας, τον μέσο όρο καθώς και ένα ιστορικό των μετρήσεων που έχει κάνει. Στην εικόνα 7.9 δείχνουμε την επιλογή του μενού θερμοκρασίας.

Εικόνα 7.9: Παρουσίαση μετρήσεων θερμοκρασίας

Επίσης δημιουργήθηκε μια ιστοσελίδα στην οποία δείχνουμε την κατάσταση του δικτύου μας. Αυτή παρουσιάζεται στην εικόνα 7.10 και εξηγείται αναλυτικά.

Health

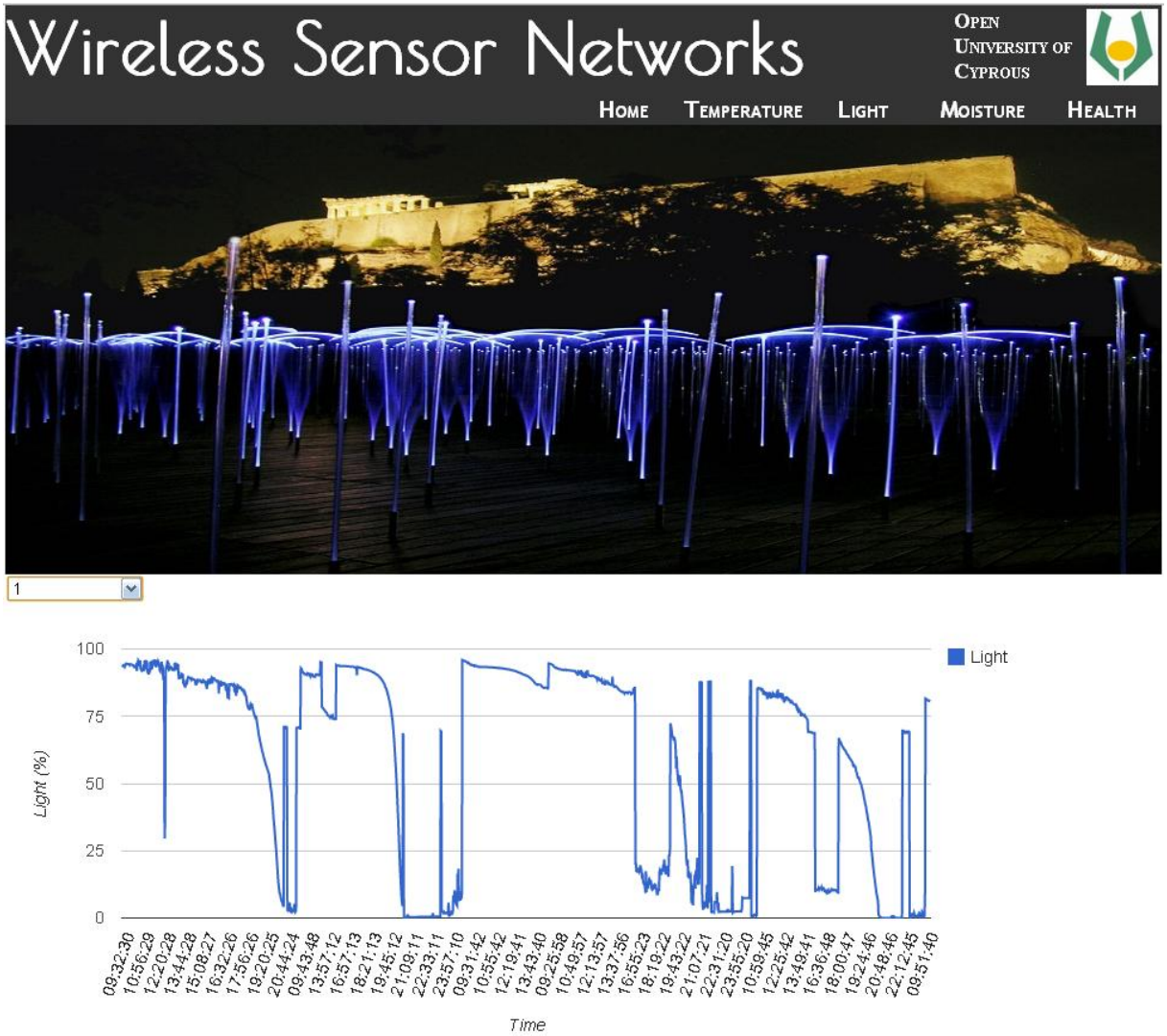
Σε αυτήν τη σελίδα μπορούμε να δούμε τις πληροφορίες για την κατάσταση του δικτύου μας .

Node id	Health Packets	Battery	Forwarded	Dropped	Retries
2	128	2.6	0	0	7
1	333	2.6	0	0	8

Εικόνα 7.10: Παρουσίαση Health packet

Στην εικόνα 7.10 ο χρήστης έχει την δυνατότητα να παρακολουθεί πόσα Health packet έστειλε ο κάθε κόμβος-αισθητήρας και πόση μπαταρία έχει μέχρι την εξάντλησή της. Επίσης φαίνονται τα πακέτα τα οποία προωθήθηκαν από τον κάθε κόμβο-αισθητήρα στον σταθμό-βάσης και είχαν προέλθει από άλλους κόμβους-αισθητήρες, πόσα πακέτα χάθηκαν καθώς και πόσα φορές ξαναπροσπάθησε ο κάθε κόμβος-αισθητήρας να αποστείλει πακέτα.

Όπως αναφέραμε και στο σχεδιασμό των απαιτήσεων θα έπρεπε να δημιουργήθεί και μια γραφική παράσταση για την παρουσίαση των αποτελεσμάτων. Χρησιμοποιήθηκε το εργαλείο Google Chart Tools καθώς και για την σύνδεση με την βάση δεδομένων την βιβλιοθήκη jquery της Javascript. Στην εικόνα 7.11 φαίνονται τα αποτελέσματα από την μέτρηση της υγρασίας εδάφους.



Εικόνα 7.11: Παρουσίαση Γραφικής Παράστασης

Κεφάλαιο 8
Επίλογος

Σε αυτό το κεφάλαιο θα αναφερθούν τα γενικά συμπεράσματα που προέκυψαν από την ενασχόλησή μας με το πεδίο των ασύρματων δικτύων αισθητήρων. Η ενασχόληση αφορά κυρίως τον σχεδιασμό και την υλοποίηση ενός συστήματος που βασίζεται σε ένα δίκτυο αισθητήρων για την συλλογή, μετάδοση, αποθήκευση και παρουσίαση μετρήσεων θερμοκρασίας, φωτός και υγρασίας εδάφους. Επίσης θα παρουσιαστούν τα συμπεράσματα από την χρήση του συστήματος. Άλλο ένα σημείο το οποίο θα δειχθεί είναι τα προβλήματα τα οποία προέκυψαν κατά την υλοποίηση της καθώς και τον τρόπο με τον οποίο ξεπεράστηκαν αυτά τα προβλήματα. Τέλος θα αναφερθούν οι μελλοντικές κατευθύνσεις οι οποίες θα μπορούσαν να βελτιώσουν το σύστημα που αναπτύχθηκε.

8.1 Γενικά Συμπεράσματα

Ο στόχος της μεταπτυχιακής διατριβής ήταν ο σχεδιασμός και η ανάπτυξη ενός ολοκληρωμένου συστήματος παρακολούθησης περιβαλλοντικών συνθηκών μέσω ενός ασύρματου δικτύου αισθητήρων και παρουσίασής τους στον τελικό χρήστη. Η πολυπλοκότητα που είχε ο σχεδιασμός και η ανάπτυξη του συστήματος ήταν μεγάλη καθότι υλοποιήθηκε χρησιμοποιώντας ένα σύνολο από τεχνολογίες. Σε επίπεδο υλικού χρειάστηκε η ενασχόληση με τα motes, η σύνδεση του εξωτερικού αισθητήρα υγρασίας με το mote καθώς και η σύνδεση των motes με τον υπολογιστή. Σε επίπεδο λογισμικού χρειάστηκε η ενασχόληση με τις γλώσσες προγραμματισμού NesC, Java, SQL, PHP, η γλώσσα CSS ενώ χρειάστηκε και η ενασχόληση με το περιβάλλον MoteWorks και PostgreSQL. Από τα παραπάνω, μπορεί κανείς να συμπεράνει, ότι χρειάστηκε ένα ευρύ πεδίο γνώσεων διότι έπρεπε να συνδυαστούν ξεχωριστές τεχνολογίες. Αυτό αποτέλεσε την μεγαλύτερη πρόκληση κατά την υλοποίηση του συστήματος λόγω του ότι θα έπρεπε να συνδυαστούν όλες οι προαναφερόμενες τεχνολογίες για να δοθεί το τελικό αποτέλεσμα.

Για το σύστημα χρησιμοποιήθηκαν οι ασύρματοι κόμβοι Micaz της εταιρίας Crossbow και το πρωτόκολλο επικοινωνίας XMesh. Επιτεύχθηκε η μετάδοση των πακέτων με τεχνολογία multi-hop από τον ένα κόμβο-αισθητήρα σε άλλο κόμβο-αισθητήρα και εκείνος με την σειρά του στον σταθμό-βάσης. Η υλοποίηση τέτοιου είδους ασύρματων δικτύων είναι σημαντική γιατί μπορεί να μας παρέχει διάφορες πληροφορίες με μικρό κόστος είτε αυτό αναφέρεται στο ανθρώπινο δυναμικό, καθότι οι αισθητήρες τοποθετούνται και μετά λειτουργούν αυτόνομα, είτε στο ενεργειακό μέρος καθότι χρειάζονται μικρή ποσότητα ενέργειας για την λειτουργία τους. Λαμβάνοντας υπόψη το γεγονός ότι οι αισθητήρες τοποθετούνται και μετά δεν χρειάζονται την ανθρώπινη παρουσία για την λειτουργία τους μας ανοίγει νέους δρόμους για την χρησιμοποίησή τους. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε τέτοιου είδους δίκτυα για την παρακολούθηση της ηφαιστειογενούς δραστηριότητας σε ενεργά ηφαίστεια όπου υπάρχει μεγάλος κίνδυνος ανθρώπινου τραυματισμού. Επίσης μπορεί να χρησιμοποιηθεί για συστήματα έξυπνης γεωργίας ελέγχοντας την υγρασία του εδάφους με απώτερο σκοπό την εξοικονόμηση νερού και άλλων πόρων.

8.2 Συμπεράσματα από τη Χρήση του Συστήματος

Αρχικά έγινε επιβεβαίωση της ορθής λειτουργίας των αισθητήρων. Στην περίπτωση του κόμβου-αισθητήρα με το MDA100 για την μέτρηση της θερμοκρασίας ήταν σχετικά απλή η επιβεβαίωση της ορθότητας της μέτρησης χρησιμοποιώντας ένα κοινό θερμόμετρο χώρου. Για την μέτρηση του φωτός σκεπάστηκε πλήρως ο αισθητήρας με ένα αδιαπέραστο από το φώς ύφασμα ώστε να παρθεί η χαμηλότερη ένδειξη. Για την υψηλότερη ένδειξη χρησιμοποιήθηκε ένας φακός κατευθείαν πάνω στο αισθητήριο. Στην συνέχεια πάρθηκαν κάποιες ενδιάμεσες τιμές χρησιμοποιώντας ένα σύστημα φωτός με ψηφιακό ροοστάτη. Στην περίπτωση του κόμβου-αισθητήρα με το MDA300 για την ορθότητα των μετρήσεων τοποθετήθηκε ο αισθητήρας σε ένα δοχείο με άνυδρο χώμα ώστε να παρθεί η ελάχιστη ένδειξη και στην συνέχεια βυθίστηκε ο αισθητήρας μέσα σε ένα δοχείο με νερό ώστε να βρεθεί η υψηλότερη ένδειξη. Με τη χρήση του συστήματος έγινε προσπάθεια να βγούνε συμπεράσματα για τον επιθυμητό ρυθμό δειγματοληψίας. Έχοντας τον ρυθμό δειγματοληψίας στα 1800 ms δηλαδή στα 1.8 δευτερόλεπτα παρατηρήθηκε ότι οι μετρήσεις, είτε στον κόμβο-αισθητήρα με το MDA100 είτε στον κόμβο-αισθητήρα με το MDA300, διήρκησαν περίπου 49 ώρες συνεχόμενης λειτουργίας. Στην συνέχεια αυξήθηκε ο ρυθμός δειγματοληψίας στα 3 δευτερόλεπτα και παρατηρήθηκε μια αύξηση στην διάρκεια των μετρήσεων αγγίζοντας περίπου τις 60 ώρες συνολικής καταγραφής. Όπως είναι φυσιολογικό ο ρυθμός δειγματοληψίας και φυσικά η αποστολή των δεδομένων στον σταθμό-βάσης παίζει σημαντικό ρόλο για την διάρκεια των μετρήσεων.

Στην συνέχεια έγινε προσπάθεια να βρεθεί η μέγιστη εφικτή απόσταση επικοινωνίας των ασύρματων-κόμβων με τον σταθμό-βάσης. Σε ελεύθερο χώρο χωρίς τοιχοποιία ενδιάμεσα η απόσταση από τον σταθμό-βάσης και τον κόμβο-αισθητήρα που έστελνε τα δικά του δεδομένα αλλά έκανε και προώθηση τα δεδομένα από τον άλλο κόμβο-αισθητήρα ήταν περίπου στα δέκα (10) μέτρα. Η απόσταση αυτή όμως μειώνονταν στα επτά (7) μέτρα περίπου όταν παρεμβαλλόταν τοιχοποιία. Οι ίδιες αποστάσεις ίσχυαν και στην αποστολή από τον ένα κόμβο-αισθητήρα στον άλλον.

8.3 Προβλήματα και Τρόποι Αντιμετώπισής.

Ένα από τα πιο σημαντικά προβλήματα με τα οποία βρεθήκαμε αντιμέτωποι από την αρχή της υλοποίησης ήταν η προβληματική λειτουργία του λογισμικού MoteWorks με το λειτουργικό σύστημα Windows 7 64bit. Γενικότερα το λογισμικό MoteWorks δεν συμβαδίζει με αρχιτεκτονικές 64bit. Έγινε προσπάθεια να λυθεί το συγκεκριμένο πρόβλημα δημιουργώντας ένα εικονικό σύστημα χρησιμοποιώντας το λογισμικό XP mode το οποίο παρέχεται από την Microsoft και το οποίο δίνει πλήρη αναγνώριση των συσκευών αλλά και πάλι το πρόβλημα συνέχιζε και υπήρχε. Το επόμενο βήμα ήταν η δημιουργία ενός εικονικού συστήματος με το λογισμικό VMware και λειτουργικού συστήματος 32bit. Με την συγκεκριμένη επιλογή βρεθήκαμε αντιμέτωποι με ένα σχετικά σπάνιο πρόβλημα που αφορούσε στο γεγονός ότι ο υπολογιστής που επιλέχθηκε για την υλοποίηση της εφαρμογής μας ήταν εξοπλισμένος με usb3.0 θύρες της Intel Corp. Το VMware δεν αναγνώριζε αυτές τις θύρες και δεν ήταν δυνατή η σύνδεση των συσκευών μέσω της θύρας. Το πρόβλημα ξεπεράστηκε χρησιμοποιώντας άλλον υπολογιστή με λειτουργικό σύστημα Windows XP 32bit.

8.4 Μελλοντική Εργασία.

Μία μεγάλη πρόκληση για το μέλλον είναι η βελτίωση της χρονικής διάρκειας ζωής του δικτύου μας. Αυτό μπορεί να επιτευχθεί με την βελτιστοποίηση των αλγόριθμων δρομολόγησης που χρησιμοποιούνται για την προώθηση των πακέτων. Ένας άλλος τρόπος είναι η δημιουργία αισθητήρων με ένα συνδυασμό από επαναφορτιζόμενες μπαταρίες και χρήση φωτοβολταϊκών συστημάτων όπου θα είχαμε θεαματική αύξηση του χρόνου ζωής του δικτύου μας

Ένας άλλος τομέας ο οποίος θα βελτίωνε το δίκτυό μας θα ήταν η προσθήκη ενός συστήματος κρυπτογράφησης των δεδομένων έτσι ώστε το σύστημά να είναι προστατευμένο από κακόβουλες ενέργειες.

Στο τομέα του υλικού μία βελτίωση θα ήταν η ύπαρξη θήκης για τους κόμβους-αισθητήρες οι οποίες θα πληρούν τις προϋποθέσεις IP-67 που αναφέρονται στο επίπεδο διαπεραστικότητας της θήκης από νερό και σκόνη.

Βιβλιογραφία

[01] Liu Yong-Min. «The Architecture and Characteristics of Wireless Sensor Network». Computer Technology and Development,. ICCTD '09. International Conference on, 561-565, 2009

[02] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor network: a survey", Computer Networks, 38(4), 393–422, March 2002

[03] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next century challenges: scalable coordination in sensor networks", ACM MobiCom'99, Washingtion, USA, pp. 263-270, 1999.

[04] 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

[05] J. Kimionis, A. Bletsas, A.G.Dimitriou, and G.N. Karystinos "Inventory Time Reduction inGen2 with Single-Antenna Separation of FM0 RFID signals IEEE International Conference on RFID Technologies and Applications (RFID-TA) 2011, Sitges, Barcelona Spain, Sep. 2011.

[06] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, T. Porcheron, Monitoring behavior in home using a smart fall sensor, IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Bio-logy, pp. 607-610, October 2000.

[07] 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.

[08] C. Herring, S. Kaplan, Component-based software systems for smart environments, IEEE Personal Communications, pp. 60-61 October 2000.

[09] J. Agre and L. Clare, "An integrated architecture for cooperative sensing networks", Computer Mag., 33(5), 106–108, 2000.

[10] 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.

[11] 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.

[12] 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.

[13] Chandrakasan et al. “Design considerations for distributed microsensor systems”, IEEE 1999 Custom Integrated Circuits Conf., 279–286, San Diego, May 1999

[14] 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.

[15] W. Heinzelman, “Application-specific protocol architecture for wireless networks”, Ph.D. dissertation, Massachusetts Institute of Technology, June 2000.

[16] Micaz Datasheet, crossbow, http://bullseye.xbow.com:81/ Products/ Product_pdf_files/ Wireless_pdf/ MICAz_Datasheet.pdf

[17] Atmel Datasheet http://www.atmel.com/Images/doc2467.pdf

[18] CC2420 datasheet, http://www.ti.com/lit/ds/symlink/cc2420.pdf

[19] MIB520CB datasheet, http://bullseye.xbow.com:81/ Products/ Product_pdf_files/ Wireless_pdf/ MIB520_Datasheet.pdf

[20] MTS/MDA Sensor and Data Acquisition Board User’s Manual, January 2006

[21] Manual Soil Moisture Sensor Ech1o. decagon, 2009.

[22] Wireless Sensor Networks: Technology, Protocols and Applications – K. Sohraby, D. Minoli, T. Znati.

[23] TinyOS, http://www.tinyos.net/

[24] 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

[25] XMesh User's .

Παράρτημα Α
Κώδικας Motes

Α.1 Κώδικας Κόμβου-Αισθητήρα MDA100

Α.1.1 MDA100.nc

```
includes sensorboardApp;
      configuration MDA100 {
      }
implementation
      {
components Main, TimerC, Voltage, PhotoTemp, ADCC,
GenericCommPromiscuous as Comm, MULTIHOPROUTER, MDA100M,
LEDS_COMPONENT,XCommandC, Bcast;

    Main.StdControl -> MDA100M;

    Main.StdControl -> MULTIHOPROUTER.StdControl;

    Main.StdControl -> Comm;
```

```
    Main.StdControl -> TimerC;

    LEDS_WIRING(MDA100M)

    MDA100M.BattControl -> Voltage;

    MDA100M.ADCBATT -> Voltage;

    MDA100M.TempControl -> PhotoTemp.TempStdControl;

    MDA100M.Temperature -> PhotoTemp.ExternalTempADC;

    MDA100M.PhotoControl -> PhotoTemp.PhotoStdControl;

    MDA100M.Light -> PhotoTemp.ExternalPhotoADC;

    MDA100M.Timer -> TimerC.Timer[unique("Timer")];

    MDA100M.XCommand -> XCommandC;

    MDA100M.XEEControl -> XCommandC;

    MDA100M.RouteControl -> MULTIHOPROUTER;

    MDA100M.Send -> MULTIHOPROUTER.MhopSend[AM_XMULTIHOP_MSG];

    MULTIHOPROUTER.ReceiveMsg[AM_XMULTIHOP_MSG]->
      Comm.ReceiveMsg[AM_XMULTIHOP_MSG];

    MDA100M.HealthMsgGet -> MULTIHOPROUTER;

    MDA100M.health_packet -> MULTIHOPROUTER.health_packet;

}
```

A.1.2 MDA100M.nc

```
includes XCommand;
includes sensorboard;

module MDA100M {
provides {interface StdControl;  }
uses {interface Leds; interface MhopSend as Send; interface RouteControl;
interface XCommand;    interface XEEControl; interface ADC as ADCBATT;
interface StdControl as BattControl;  interface StdControl as TempControl;
interface ADC as Temperature;   interface StdControl as PhotoControl;
interface ADC as Light;   interface Timer;}

command void health_packet(bool enable, uint16_t intv);
command HealthMsg* HealthMsgGet();  }}

implementation {  enum { START, BUSY, BATT_DONE, TEMP_DONE, LIGHT_DONE};

  TOS_Msg msg_buf;     TOS_MsgPtr msg_ptr;    HealthMsg *h_msg;    norace
bool sending_packet,sensinginsession;    norace uint8_t state;    XDataMsg
pack;     bool       sleeping;

  static void initialize()

   static void start()     {

      atomic state = START;
      call BattControl.start();
      call TempControl.start();
      call PhotoControl.start();     }

    task void stop()    {

      call StdControl.stop();    }

    task void battstop()    {

      call BattControl.stop();    }
```

```
    task void tempstop()    {

      call TempControl.stop();    }

    task void photostop()    {

      call PhotoControl.stop();    }

   task void send_radio_msg(){

    uint16_t  len;
      XDataMsg *data;
uint8_t i;
if(sending_packet) return;
call Leds.yellowOn();
atomic sending_packet=TRUE;
data = (XDataMsg*)call Send.getBuffer(msg_ptr, &len);
      ((uint8_t*) data)[1] = ((uint8_t*)&pack)[1];
data->xMeshHeader.board_id = SENSOR_BOARD_ID;
data->xMeshHeader.packet_id = 1;
data->xMeshHeader.parent    = call RouteControl.getParent();
data->xMeshHeader.packet_id = data->xMeshHeader.packet_id | 0x80;

    if (TOS_LOCAL_ADDRESS != 0) {
      call Leds.yellowOn();
call PowerMgrDisable();
TOSH_uwait(1000);
if   (call   SendUART.send(TOS_UART_ADDR,   sizeof(XDataMsg),msg_ptr)   !=
SUCCESS) {
      atomic sending_packet = FALSE;
      call Leds.greenToggle();
      call PowerMgrEnable();      }       }
      else        {
if       (call       Send.send(BASE_STATION_ADDRESS,MODE_UPSTREAM,msg_ptr,
sizeof(XDataMsg)) != SUCCESS) {
      atomic sending_packet = FALSE;
          call Leds.yellowOn();
      call Leds.greenOff();       }       }
return;  }
```

```
  command result_t StdControl.init() {
      atomic {    msg_ptr = &msg_buf;    }
call BattControl.init();
call Leds.init();
call TempControl.init();
call PhotoControl.init();
initialize();
      return SUCCESS;  }

 /*Start the component. Start the clock.*/

  command result_t StdControl.start(){
call StdControl.stop();
h_msg = call HealthMsgGet();
h_msg->rsvd_app_type = SENSOR_BOARD_ID;
call Timer.start(TIMER_REPEAT, timer_rate);
call health_packet(TRUE,TOS_HEALTH_UPDATE);
return SUCCESS;    }

 /*Stop the component*/

  command result_t StdControl.stop() {
call BattControl.stop();
call TempControl.stop();
call PhotoControl.stop();
return SUCCESS;     }

/*Measure Temperature, Light */

event result_t Timer.fired() {
if ( !sending_packet) {
      start();
      if (!sensinginsession){
      call ADCBATT.getData();
atomic sensinginsession = TRUE;  }    }
return SUCCESS;    }

  /* Battery Ref  or thermistor data ready */
```

```
  async event result_t ADCBATT.dataReady(uint16_t data) {
if (!sensinginsession) return FAIL;
atomic sensinginsession = FALSE;
atomic pack.xData.datap1.vref = data ;
post battstop();
atomic state = BATT_DONE;
call Temperature.getData();
return SUCCESS;  }

   /*Send temperature data packet*/

  async event result_t Temperature.dataReady(uint16_t data) {
atomic pack.xData.datap1.thermistor = data ;
post tempstop();
atomic state = TEMP_DONE;
call Light.getData();
return SUCCESS;  }

/* Send data packet */

  event result_t SendUART.sendDone(TOS_MsgPtr msg, result_t success)    {
atomic msg_ptr = msg;
msg_ptr->addr = TOS_BCAST_ADDR;
      if (call Send.send( BASE_STATION_ADDRESS, MODE_UPSTREAM, msg_ptr,
sizeof(XDataMsg) ) != SUCCESS) {
atomic sending_packet = FALSE;
call Leds.yellowOff();      }
if (TOS_LOCAL_ADDRESS != 0)
call PowerMgrEnable();}
return SUCCESS;  }

event result_t Send.sendDone(TOS_MsgPtr msg, result_t success)      {
atomic {
msg_ptr = msg;
sending_packet = FALSE; }
call Leds.yellowOff();
return SUCCESS;  }
```

```
   event result_t XCommand.received(XCommandOp *opcode) {
switch (opcode->cmd) {
case XCOMMAND_SLEEP:sleeping = TRUE;
call Timer.stop();
call Leds.set(0);
call StdControl.stop();
break;
case XCOMMAND_WAKEUP:
if (sleeping) {
initialize();
call Timer.start(TIMER_REPEAT, timer_rate);
sleeping = FALSE;       }
break;
default:
break;}
return SUCCESS;  }

event result_t XEEControl.restoreDone(result_t result)   {
      if(result) {
      call Timer.stop();
      call Timer.start(TIMER_REPEAT, timer_rate);          }
       return SUCCESS;   }  }
```

A.1.3 Sensorapp.h

```
typedef struct XMeshHeader{
uint8_t  board_id;
uint8_t  packet_id;
uint16_t parent;
      }__attribute__ ((packed)) XMeshHeader;

typedef struct PData1 {
uint16_t vref;
uint16_t thermistor;
uint16_t photo;
      } __attribute__ ((packed)) PData1;
```

```
typedef struct XDataMsg {
XMeshHeader xMeshHeader;
union {
PData1    datap1;
}xData;
      } __attribute__ ((packed)) XDataMsg;

enum {
AM_XSXMSG = 0,};

enum {
AM_XMULTIHOP_MSG = 51,};

uint32_t XSENSOR_SAMPLE_RATE = 2000;
uint32_t   timer_rate;
```

A.1.4 Makefile

```
include Makefile.component
include ../../MakeXbowlocal
GOALS += binlink
include $(MAKERULES)
COMPONENT=MDA100CB
SENSORBOARD=mda100cb
```

A.2 Κώδικας Κόμβου-Αισθητήρα MDA300

A.2.1 MDA300.nc

```
includes sensorboardApp;
      configuration XMDA300 { }

implementation {  components Main,      GenericCommPromiscuous as Comm,
```

```
    MULTIHOPROUTER,MDA300M,   LEDS_COMPONENT,XCommandC, Bcast,
SamplerC,TimerC;

Main.StdControl -> MDA300M;
Main.StdControl -> MULTIHOPROUTER.StdControl;
Main.StdControl -> Comm;
Main.StdControl -> TimerC;
LEDS_WIRING(MDA300M)
MDA300M.Timer -> TimerC.Timer[unique("Timer")];
MDA300M.SamplerControl -> SamplerC.SamplerControl;
MDA300M.Sample -> SamplerC.Sample;
MDA300M.PlugPlay -> SamplerC.PlugPlay;
MDA300M.XCommand -> XCommandC;
MDA300M.XEEControl -> XCommandC;
MDA300M.RouteControl -> MULTIHOPROUTER;
MDA300M.Send -> MULTIHOPROUTER.MhopSend[AM_XMULTIHOP_MSG];
MULTIHOPROUTER.ReceiveMsg[AM_XMULTIHOP_MSG]                              ->
Comm.ReceiveMsg[AM_XMULTIHOP_MSG];
MDA300M.HealthMsgGet -> MULTIHOPROUTER.HealthMsgGet;
MDA300M.health_packet -> MULTIHOPROUTER.health_packet;}
```

A.2.2 MDA300M.nc

```
includes XCommand;
      includes sensorboard;
      module MDA300M    {    provides interface StdControl;
uses {interface Leds;   interface MhopSend as Send;   interface
RouteControl; interface XCommand; interface XEEControl;    interface
StdControl as SamplerControl; interface   Sample;        interface   Timer;
command result_t PlugPlay();
command void health_packet(bool enable, uint16_t intv);
command HealthMsg* HealthMsgGet();    } }
      implementation {     enum {   PENDING = 0,      NO_MSG  =  1        };
enum {      MDA300_PACKET1 = 1,     MDA300_ERR_PACKET = 0xf8         };
bool       sleeping;
bool sending_packet;
uint16_t    seqno;
```

```
XDataMsg  *tmppack;
TOS_Msg packet;
TOS_Msg msg_send_buffer;
TOS_MsgPtr msg_ptr;
HealthMsg *h_msg;
bool  bBoardOn=TRUE;
uint16_t msg_status, pkt_full;
char test;
uint8_t samplebatt=0;
int8_t record[25];
task void send_radio_msg();
static void initialize()
static void start()     {
bBoardOn=TRUE;
call SamplerControl.start();
if(call PlugPlay())     {
bBoardOn=TRUE;
record[0]   =   call   Sample.getSample(0,ANALOG,ANALOG_SAMPLING_TIME   |
EXCITATION_25 | DELAY_BEFORE_MEASUREMENT);
call Leds.greenOn();        }
else {            bBoardOn=FALSE;   }
atomic samplebatt=1;
call Sample.sampleNow();
return;    }

/*Initialize the component. Initialize Leds*/

command result_t StdControl.init() {
      call Leds.init();
      atomic {
msg_ptr = &msg_send_buffer;   }

/*Start the component. Start the clock. Setup timer and sampling*/

command result_t StdControl.start() {
h_msg = call HealthMsgGet();
h_msg->rsvd_app_type = SENSOR_BOARD_ID;
call Timer.start(TIMER_REPEAT, timer_rate);
```

```
call health_packet(TRUE,TOS_HEALTH_UPDATE);
return SUCCESS;    }

/*Stop the component.*/

command result_t StdControl.stop() {
call Sample.stop(0);
call SamplerControl.stop();
      return SUCCESS;    }
task void send_radio_msg()    {
uint8_t i;
uint16_t  len;
XDataMsg *data;
if(sending_packet)
return;
atomic sending_packet=TRUE;
      data = (XDataMsg*)call Send.getBuffer(msg_ptr, &len);
tmppack=(XDataMsg *)packet.data;
      for (i = 0; i <= sizeof(XDataMsg)-1; i++)
((uint8_t*)data)[i] = ((uint8_t*)tmppack)[i];
      if(bBoardOn)      {         data->xmeshHeader.packet_id = 6;         }
else   {         data->xmeshHeader.packet_id = 7;         }
      data->xmeshHeader.board_id  = SENSOR_BOARD_ID;
      data->xmeshHeader.parent    = call RouteControl.getParent();
      data->xmeshHeader.packet_id = data->xmeshHeader.packet_id | 0x80;

      if (TOS_LOCAL_ADDRESS != 0) {
      call Leds.yellowOn();
call PowerMgrDisable();
TOSH_uwait(1000);
if   (call   SendUART.send(TOS_BCAST_ADDR,   sizeof(XDataMsg),msg_ptr)   !=
SUCCESS)        {
      atomic sending_packet = FALSE;
      call Leds.greenToggle();
      call PowerMgrEnable();           }       }

  event result_t SendUART.sendDone(TOS_MsgPtr msg, result_t success)    {
{      atomic msg_ptr = msg;
msg_ptr->addr = TOS_BCAST_ADDR;
```

```
if       (call       Send.send(BASE_STATION_ADDRESS,MODE_UPSTREAM,msg_ptr,
sizeof(XDataMsg)) != SUCCESS) {
atomic sending_packet = FALSE;
call Leds.yellowOff();        }
if (TOS_LOCAL_ADDRESS != 0)
call PowerMgrEnable();}
return SUCCESS;     }#endif

event result_t
      Sample.dataReady(uint8_t  channel,uint8_t  channelType,uint16_t  data)
      {
switch (channelType) {
      case ANALOG:
switch (channel) {
      case 0:
tmppack=(XDataMsg *)packet.data;
tmppack->xData.datap6.adc0 =data ;
atomic {msg_status|=0x01;}
break;
      default:
break;    }
break;
      case BATTERY:
      if(samplebatt==0) break;
      atomic {
      samplebatt=0;
      tmppack=(XDataMsg *)packet.data;
      tmppack->xData.datap6.vref =data ;
      msg_status|=0x40;}
if(!bBoardOn)         {
post send_radio_msg();      }
break;
      default:
break;    }
if (sending_packet)
return SUCCESS;
      if (msg_status == pkt_full) {
atomic msg_status = 0;
call StdControl.stop();
```

```
post send_radio_msg();        }
return SUCCESS;         }

  /*Timer Fired */
      event result_t Timer.fired() {
if (sending_packet && msg_status!=0)
return SUCCESS;
start();
      return SUCCESS;}
event result_t XCommand.received(XCommandOp *opcode) {
switch (opcode->cmd) {
case XCOMMAND_SLEEP:
sleeping = TRUE;
call StdControl.stop();
call Timer.stop();
call Leds.set(0);
break;
case XCOMMAND_WAKEUP:
if (sleeping) {
initialize();
call Timer.start(TIMER_REPEAT, timer_rate);
sleeping = FALSE;             }
break;
default:
break; }
return SUCCESS;  }
event result_t XEEControl.restoreDone(result_t result)  {
      if(result) {
      call Timer.stop();
      call Timer.start(TIMER_REPEAT, timer_rate);          }
      return SUCCESS; } }
```

A.2.3 Sensorapp.h

```
typedef struct XMeshHeader{
uint8_t  board_id;
uint8_t  packet_id;
```

```
uint16_t parent;
      }__attribute__ ((packed)) XMeshHeader;

typedef struct PData1 {
uint16_t vref;
uint16_t humid;
uint16_t humtemp;
uint16_t adc0;
      } __attribute__ ((packed)) PData1;

typedef struct XDataMsg {
XMeshHeader xmeshHeader;
union {
PData1    datap1;
xData;
      }  __attribute__ ((packed)) XDataMsg;

enum {  AM_XSXMSG = 0,  };

enum {        AM_XMULTIHOP_MSG = 51,          };

uint32_t XSENSOR_SAMPLE_RATE = 2000;
uint32_t   timer_rate;
```

A.2.4 Makefile

```
include Makefile.component
include ../../MakeXbowlocal
GOALS += binlink
include $(MAKERULES)
COMPONENT=MDA300
SENSORBOARD=mda300
```

A.3 Κώδικας Σταθμού-Βάσης

A.3.1 XMeshBaseStation.nc

```
includes sensorboardApp;

configuration XMeshBaseStation {

}

implementation

{

    components Main,

      MULTIHOPROUTER,XMeshBaseStationM,

      LEDS_COMPONENT

      HEARTBEAT_COMPONENT

      XCommandC;

    Main.StdControl -> XMeshBaseM;

    HEARTBEAT_WIRING()

    LEDS_WIRING(XMeshBaseStationM)

    XMeshBaseM.XCommand -> XCommandC;

    XMeshBaseM.RouteControl -> MULTIHOPROUTER;

}
```

A.3.2 XMeshBaseStationM.nc

```
module XMeshBaseStationM {

    provides {

      interface StdControl;

    }

    uses {

      interface RouteControl;

      interface XCommand;

      interface Leds;

    }

}

implementation {

  command result_t StdControl.init() {

        call Leds.init();

        atomic{ TOS_LOCAL_ADDRESS = 0; }

      return SUCCESS;

  }

  command result_t StdControl.start(){

      return SUCCESS;

  }
```

```
  command result_t StdControl.stop() {
      return SUCCESS;
  }
  event result_t XCommand.received(XCommandOp *opcode) {
      switch (opcode->cmd) {
        case XCOMMAND_SLEEP:
              break;
        case XCOMMAND_WAKEUP:
            break;
        default:
      break;
      }
      return SUCCESS;
  }
}
```

A.3.3 Sensorapp.h

```
enum {
    AM_XMULTIHOP_MSG = 51,
};
uint32_t   timer_rate;
```

A.3.4 Makefile

```
include Makefile.component

include ../../MakeXbowlocal

GOALS += basic freq route,hp base binlink

include $(MAKERULES)

DEFINES += -DMHOP_QUEUE_SIZE=12

COMPONENT=XMeshBase

MSG_SIZE=55

INCLUDES += -I${TOSROOT}/tos/lib/XHeartbeat
```

Παράρτημα Β

Κώδικας Δημιουργίας Βάσης Δεδομένων

```
CREATE TABLE mda100_ mesurments ( result_time timestamp without time zone, nodeid integer, parent integer,voltage integer, temp integer, light integer

CREATE TABLE mda300_ mesurments ( result_time timestamp without time zone, nodeid integer, parent integer,adc0 integer, voltage integer)

CREATE TABLE node_health ( result_time timestamp without time zone,epoch integer, nodeid integer, health_pkts integer, node_pkts integer, forwarded integer, dropped integer, retries integer, battery integer, power_sum integer,board_id integer, parent integer, quality_tx integer, quality_rx integer, path_cost integer, parent_rssi integer)
```

Παράρτημα Γ
Κώδικας Δημιουργίας Ιστότοπου

Γ.1 Κώδικας CSS

```
@charset "utf-8";

/* CSS Document */



#container {

	width: 968px;

	background: #FFF;

	margin: 0 auto;

	padding-left: 10px;

	padding-right: 10px;
```

```
	overflow: hidden;
}#main_image_test {
	background-image: url(../images/main.jpg);
	background-repeat: no-repeat;
	height: 376px;
	width: 968px;
}

#center_column {
	width: 968px;
	position: relative;
}

h1 {
	font-family: poiret-one, serif;
	font-size: 60px;
	margin: 0;
}#header {
	color: rgba(255,255,255,1);
	background-color: rgba(51,51,51,1);
```

```
	height: 100px;

	position: relative;

}

#header a {

	font-size: 20px;

	font-weight: bold;

	font-variant: small-caps;

	color: rgba(255,255,255,1);

	text-decoration: none;

	text-align: center;

	width: 100px;

	display: block;

}



#header ul {

	margin: 0px;

	padding: 0px;

	list-style-type: none;

	position: absolute;

	right: 0px;
```

```
	bottom: 0px;
}



body {
	font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
	color: #3B3B3B;
	background-color: rgba(255,255,255,1);
	margin: 0px;
}
#header ul li {
	float: left;
}



#current_result {
	background-color: rgba(255,255,255,1);
	width: 100px;
	position: absolute;
	right: 868px;
	top: 120px;
```

```
}

#all_result {

	background-color: rgba(255,255,255,1);

	width: 100px;

	position: absolute;

	right: 868px;

	top: 200px;

}



#avg_result {

	background-color: rgba(255,255,255,1);

	width: 100px;

	position: absolute;

	right: 868px;

	top: 160px;

}

table.db-table	{ border-right:1px solid #ccc; border-bottom:1px solid #ccc; }

table.db-table th { background:#eee; padding:5px; border-left:1px solid #ccc; border-top:1px solid #ccc; }
```

```
table.db-table td {  padding:5px;  border-left:1px  solid  #ccc;  border-top:1px solid #ccc; }
```

Γ.2 Κώδικας Index.html

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>



<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60" height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman" color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF </font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>
```

```
    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>

  <div id="splash"><img src="images/main.jpg" width="968" height="376"
alt="" /></div>

  <div id="center_column">

    <h2>Περιγραφή της Μεταπτυχιακής Διατριβής</h2>

    <p>Σκοπός της παρούσας διατριβής είναι ο σχεδιασμός και η ανάπτυξη
ενός ολοκληρωμένου συστήματος παρακολούθησης περιβαλλοντικών συνθηκών
μέσω ενός ασύρματου δικτύου αισθητήρων. Η διατριβή θα μελετήσει τα δίκτυα
ασύρματων αισθητήρων από την σκοπιά των εμπλεκόμενων τεχνολογιών και τη
δυνατότητα εφαρμογής τους σε πρακτικό επίπεδο και τα πιθανά προβλήματα
που θα πρέπει να αντιμετωπιστούν στην πορεία αυτή. Οι αισθητήρες που θα
ενσωματωθούν στο δίκτυο θα καταγράφουν μετρήσεις σύμφωνα με ένα ορισμένο
ρυθμό δειγματοληψίας. Τα δεδομένα θα αποθηκεύονται προσωρινά στον τοπικό
κόμβο πριν να μεταδοθούν ασύρματα σ&rsquo; έναν σταθμό βάσης που παίζει
τον ρόλο του καταγραφέα δεδομένων (data logger). Τα δεδομένα από το
σταθμό βάσης μεταδίδονται περιοδικά σε μια βάση δεδομένων που
χρησιμοποιείται από τον τελικό χρήστη για επισκόπηση και ανάλυση της
πληροφορίας. </p>
```

```
    <p>Εκπόνηση Διατριβής: Θωμάς Καρανίκας</p>

    <p> Επιβλεψη Διατριβής: Χρήστος Γκουμόπουλος</p>

  </div>

</div>

</body>

</html>
```

Γ.3 Κώδικας Temperature.php

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>



<body>

<div id="container">
```

```
  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>

  <div    id="splash"><img    src="images/temperature.jpg"    width="968"
height="376" alt="" /></div>

  <div id="center_column">

    <div id="all_result"><form name="allresults" method="post">

<input type="submit" name="allresult" value="Timeline" /></form>

      </div>
```

```
     <div id="avg_result"><form name="avgresult" method="post">

<input type="submit" name="avgresult" value="Average estimation" /></form>

      </div>

    <div id="current_result">

      <form name="currentresult" method="post">

<input type="submit" name="currentresult" value="Last measurement" />
</form>

    </div>

    <h2>Temperature</h2>

    <p>Σε αυτήν την ενότητα μπορούμε να δούμε σε πραγματικό χρόνο τις μετρήσεις θερμοκρασίας που παίρνουμε απο τους αισθητήτρες μας, τον μέσο όρο καθώς και το ιστορικό των θερμοκρασιών που έχουμε συλλέξει και αποθηκεύσει στην βάση δεδομένων μας.</p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p>

 <?php

if (isset($_POST['allresult'])) {

header("Location: chartindextemp.php");

exit;
```

```
}

else if (isset($_POST['currentresult'])) {

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT * FROM ( SELECT
MAX(result_time) as timestamp, nodeid FROM mda100_results GROUP BY nodeid
) as current,

mda100_results

WHERE current.nodeid=mda100_results.nodeid AND
current.timestamp=mda100_results.result_time");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {

    $temp=((1/(0.001307050 + 0.000214381 * log( 10000*(1023-
$row['temp'])/$row['temp']) + 0.000000093 * (pow(log( 10000*(1023-
$row['temp'])/$row['temp'] ),3)))) - 273.15);
```

```
            echo "<table cellspacing=0 cellpadding=0 class=db-table
align=center><tr><th>Result                                    Time</th><th>Node
id</th><th>Temperature</th></tr>";

            echo "<tr><td><center>" . substr($row['result_time'],0,19).
"</td>".

                  "<td><center>" . $row['nodeid']. "</td><td><center>" .
substr($temp,0,5) ."</td></tr></table>";

      }

      }

else if (isset($_POST['avgresult'])) {

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT nodeid, avg(temp) as temp
FROM mda100_results GROUP BY nodeid ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {
```

```
 $temp=((1/(0.001307050    +    0.000214381    *    log(    10000*(1023-
$row['temp'])/$row['temp'])   +   0.000000093   *   (pow(log(   10000*(1023-
$row['temp'])/$row['temp'] ),3)))) - 273.15);

            echo   "<table   cellspacing=0   cellpadding=0   class=db-table
align=center><tr><th>Node id</th><th>Temperature </th></tr>";

            echo "<tr><td><center>" . $row['nodeid']. "</td><td><center>"
. substr($temp,0,5) ."</td></tr></table>";

      }

      }

?>

    </p>

  </div>

</div>

</body>

</html>
```

Γ.4 Κώδικας getLinechartdatatemp.php

```
<?php

$connection    =    pg_connect("host=localhost    dbname=mydb    user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");
```

```
    exit;

  }

      $result = pg_exec($connection, "SELECT * FROM mda100_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());

  echo                  "{                  \"cols\":                  [
{\"id\":\"\",\"label\":\"Nodeid\",\"pattern\":\"\",\"type\":\"string\"},
{\"id\":\"\",\"label\":\"Temperature\",\"pattern\":\"\",\"type\":\"number\
"} ], \"rows\": [ ";



  $total_rows = pg_num_rows($result);

  $row_num = 0;

  while($row = pg_fetch_array($result)){

    $row_num++;

    $temp=((1/(0.001307050    +    0.000214381    *    log(    10000*(1023-
$row['temp'])/$row['temp'])   +   0.000000093   *   (pow(log(   10000*(1023-
$row['temp'])/$row['temp'] ),3)))) - 273.15);



if ($row_num == $total_rows){

      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($temp,0,5) . ",\"f\":null}]}";

    } else {
```

```
      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($temp,0,5) . ",\"f\":null}]}, ";

    }

  }

  echo " ] }";

?>
```

Γ.5 Κώδικας chartindextemp.php

```
<html>

<head>

  <!--Load the AJAX API-->

  <script                                              type="text/javascript"
src="http://www.google.com/jsapi"></script>

  <script type="text/javascript" src="jquery-1.9.1.min.js"></script>

  <script type="text/javascript">



  // Load the Visualization API and the Linechart.

  google.load('visualization', '1', {'packages':['corechart']});

  function drawItems(num) {

    var jsonLineChartData = $.ajax({

      url: "getLinechartdatatemp.php",
```

```
      data: 2+num,
      dataType:"json",
      async: false
    }).responseText;
    // Create our data table out of JSON data loaded from server.
    var Linechartdata = new google.visualization.DataTable(jsonLineChartData);
    // Instantiate and draw our Line chart, passing in some options.
    var chart = new google.visualization.LineChart(document.getElementById('center_column'));
    chart.draw(Linechartdata, {
     width: 968,
      height: 376,
        vAxis: {title: 'Temperature', titlePosition:'out' },
        hAxis: {title: 'Time',slantedTextAngle: 70, titlePosition:'out' },
     chartArea: { left:"10%",top:"10%",width:"70%",height:"60%" }
    });
  }
  </script>
  <meta charset="utf-8">
<title>Wireless Sensor Networks</title>
```

```
<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>
```

```
  <div     id="splash"><img     src="images/temperature.jpg"     width="968"
height="376" alt="" /></div>

 <form>

  <select name="users" onChange="drawItems(this.value)">

  <option value="">Select a Node:</option>

  <?php



$connection    =    pg_connect("host=localhost    dbname=mydb    user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result  =  pg_exec($connection,  "SELECT  MAX(nodeid)  as  nodeid  FROM
mda100_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



while ($row = pg_fetch_array($result)) {

echo '<option value="'.$row['nodeid'].'">'.$row['nodeid'].'</option>';

}
```

```
  ?>

  </select>

  <div id="center_column"></div>

</body>

</html>
```

Γ.6 Κώδικας light.php

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>



<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>
```

```
  <table align=right><tr><td><font size="4" face="Times New Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>

  <div id="splash"><img src="images/light.jpg" width="968" height="376"
alt="" /></div>

  <div id="center_column">

  <div id="all_result"><form name="allresults" method="post">

<input type="submit" name="allresult" value="Timeline" /></form>

      </div>

    <div id="current_result">

      <form name="currentresult" method="post">
```

```
<input type="submit" name="currentresult" value="Last measurement" /></form>

    </div>

    <div id="avg_result"><form name="avgresult" method="post">

<input type="submit" name="avgresult" value="Average estimation" /></form>

      </div>

    <h2>Light</h2>

    <p>Σε αυτήν την ενότητα μπορούμε να δούμε σε πραγματικό χρόνο τις μετρήσεις φωτός που παίρνουμε απο τους αισθητήτρες μας καθώς και το ιστορικό τους που έχουμε συλλέξει και αποθηκεύσει στην βάση δεδομένων μας.</p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p>

<?php

if (isset($_POST['allresult'])) {

header("Location: chartindexlight.php");

exit;

}

else if (isset($_POST['currentresult'])) {
```

```
$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT * FROM ( SELECT
MAX(result_time) as timestamp, nodeid FROM mda100_results GROUP BY nodeid
) as current,

mda100_results

WHERE current.nodeid=mda100_results.nodeid AND
current.timestamp=mda100_results.result_time");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {

      $light=$row['light']/10;



            echo "<table cellspacing=0 cellpadding=0 class=db-table
align=center><tr><th>Result Time</th><th>Node id</th><th>Light
(%)</th></tr>";
```

```
            echo  "<tr><td><center>"  .  substr($row['result_time'],0,19).
"</td>".

                  "<td><center>" . $row['nodeid']. "</td><td><center>" .
substr($light,0,4) ."</td></tr></table>";

      }

      }

else if (isset($_POST['avgresult'])) {

$connection    =    pg_connect("host=localhost    dbname=mydb    user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT nodeid, avg(light) as light
FROM mda100_results GROUP BY nodeid ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {

      $light=$row['light']/10;
```

```
            echo "<table cellspacing=0 cellpadding=0 class=db-table
align=center><tr><th>Node id</th><th>Light (%)</th></tr>";

            echo "<tr><td><center>" . $row['nodeid']. "</td><td><center>"
. substr($light,0,5) ."</td></tr></table>";

      }

      }

?>

    </p>

  </div>

</div>

</body>

</html>
```

Γ.7 Κώδικας getLinechartdatalight.php

```
<?php

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;
```

```
  }

      $result = pg_exec($connection, "SELECT * FROM mda100_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());

  echo                  "{                  \"cols\":                  [
{\"id\":\"\",\"label\":\"Nodeid\",\"pattern\":\"\",\"type\":\"string\"},
{\"id\":\"\",\"label\":\"Light\",\"pattern\":\"\",\"type\":\"number\"}  ],
\"rows\": [ ";



  $total_rows = pg_num_rows($result);

  $row_num = 0;

  while($row = pg_fetch_array($result)){

    $row_num++;

      $light=$row['light']/10;



if ($row_num == $total_rows){

      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($light,0,5) . ",\"f\":null}]}";
```

```
    } else {

      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($light,0,5) . ",\"f\":null}]}, ";

    }

  }

  echo " ] }";

?>
```

Γ.8 Κώδικας chartindexlight.php

```
<html>

<head>

  <!--Load the AJAX API-->

  <script                                              type="text/javascript"
src="http://www.google.com/jsapi"></script>

  <script type="text/javascript" src="jquery-1.9.1.min.js"></script>

  <script type="text/javascript">
```

```
  // Load the Visualization API and the Linechart.

  google.load('visualization', '1', {'packages':['corechart']});



  function drawItems(num) {

    var jsonLineChartData = $.ajax({

      url: "getLinechartdatalight.php",

      data: 2+num,

      dataType:"json",

      async: false

    }).responseText;



    // Create our data table out of JSON data loaded from server.

    var Linechartdata = new
google.visualization.DataTable(jsonLineChartData);
```

```
    // Instantiate and draw our Line chart, passing in some options.

    var chart = new google.visualization.LineChart(document.getElementById('center_column'));

    chart.draw(Linechartdata, {

      width: 968,

      height: 376,

        vAxis: {title: 'Light (%)', titlePosition:'out' },

        hAxis: {title: 'Time',slantedTextAngle: 70, titlePosition:'out' },

     chartArea: { left:"10%",top:"10%",width:"70%",height:"60%" }

    });

  }

  </script>

<meta charset="utf-8">
```

```
<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>
```

```
      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>

  <div id="splash"><img src="images/light.jpg" width="968" height="376" alt="" /></div>

 <form>

  <select name="users" onChange="drawItems(this.value)">

  <option value="">Select a Node:</option>

  <?php



$connection = pg_connect("host=localhost dbname=mydb user=tele password=tiny");

  if (!$connection) {

    print("Connection Failed.");
```

```
    exit;

  }

      $result = pg_exec($connection, "SELECT MAX(nodeid) as nodeid FROM
mda100_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());




while ($row = pg_fetch_array($result)) {

echo '<option value="'.$row['nodeid'].'">'.$row['nodeid'].'</option>';

}









  ?>

  </select>

  <div id="center_column"></div>

</body>
```

```
</html>
```

Γ.9 Κώδικας moisture.php

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>



<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>
```

```
  <table align=right><tr><td><font size="4" face="Times New Roman" color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF </font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>

  </div>

  <div id="splash"><img src="images/moisture.jpg" width="968" height="376" alt="" /></div>

  <div id="center_column">

    <div id="all_result"><form name="allresults" method="post">
```

```
<input type="submit" name="allresult" value="Timeline" /></form>

      </div>

     <div id="avg_result"><form name="avgresult" method="post">

<input type="submit" name="avgresult" value="Average estimation" /></form>

      </div>

    <div id="current_result">

      <form name="currentresult" method="post">

<input type="submit" name="currentresult" value="Last measurement" /></form>

    </div>

    <h2>Moisture</h2>

    <p>Σε αυτήν την ενότητα μπορούμε να δούμε σε πραγματικό χρόνο τις μετρήσεις της υγρασίας εδάφους που παίρνουμε απο τους αισθητήτρες μας καθώς και το ιστορικό των μετρήσεων που έχουμε συλλέξει και αποθηκεύσει στην βάση δεδομένων μας.</p>

    <p> </p>

    <p> </p>

    <p> </p>
```

```
    <p> </p>

    <p>

 <?php

if (isset($_POST['allresult'])) {

header("Location: chartindexmoisture.php");

exit;

}

else if (isset($_POST['currentresult'])) {

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT * FROM ( SELECT
MAX(result_time) as timestamp, nodeid FROM mda300_results GROUP BY nodeid
) as current,
```

```
mda300_results

WHERE current.nodeid=mda300_results.nodeid AND
current.timestamp=mda300_results.result_time");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {

      $moisture=(($row['adc0']*(1/11.5)) - 34)/10;



            echo "<table cellspacing=0 cellpadding=0 class=db-table
align=center><tr><th>Result Time</th><th>Node id</th><th>Moisture
(%)</th></tr>";

            echo "<tr><td><center>" . substr($row['result_time'],0,19).
"</td>".

                  "<td><center>" . $row['nodeid']. "</td><td><center>" .
substr($moisture,0,5) ."</td></tr></table>";

      }
```

```
      }

else if (isset($_POST['avgresult'])) {

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT nodeid, avg(adc0) as adc0
FROM mda300_results GROUP BY nodeid ");

    if (!$result)

           die("Error in SQL query: " . pg_last_error());



      while ($row = pg_fetch_array($result))

      {

      $moisture=(($row['adc0']*(1/11.5)) - 34)/10;
```

```
            echo "<table cellspacing=0 cellpadding=0 class=db-table
align=center><tr><th>Node id</th><th>Moisture (%)</th></tr>";

            echo "<tr><td><center>" . $row['nodeid']. "</td><td><center>"
. substr($moisture,0,5) ."</td></tr></table>";

      }

      }

?>

    </p>

  </div>

</div>

</body>

</html>
```

Γ.10 Κώδικας getLinechartdatamoisture.php

```
<?php

$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT * FROM mda300_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());

  echo "{ \"cols\": [
{\"id\":\"\",\"label\":\"Nodeid\",\"pattern\":\"\",\"type\":\"string\"},
{\"id\":\"\",\"label\":\"Moisture\",\"pattern\":\"\",\"type\":\"number\"}
], \"rows\": [ ";



  $total_rows = pg_num_rows($result);
```

```
  $row_num = 0;

  while($row = pg_fetch_array($result)){

    $row_num++;

      $moisture=(($row['adc0']*(1/11.5)) - 34)/10;



if ($row_num == $total_rows){

      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($moisture,0,5) . ",\"f\":null}]}";

    } else {

      echo   "{\"c\":[{\"v\":\""   .   substr($row['result_time'],11,8)   .
"\",\"f\":null},{\"v\":" . substr($moisture,0,5) . ",\"f\":null}]}, ";

    }

  }

  echo " ] }";

?>
```

Γ.11 Κώδικας chartindexmoisture.php

```
<html>

<head>

  <!--Load the AJAX API-->

  <script                                          type="text/javascript"
src="http://www.google.com/jsapi"></script>

  <script type="text/javascript" src="jquery-1.9.1.min.js"></script>

  <script type="text/javascript">



  // Load the Visualization API and the Linechart.

  google.load('visualization', '1', {'packages':['corechart']});



  function drawItems(num) {

    var jsonLineChartData = $.ajax({

      url: "getLinechartdatamoisture.php",

      data: 2+num,

      dataType:"json",

      async: false

    }).responseText;
```

```
    // Create our data table out of JSON data loaded from server.

    var Linechartdata = new
google.visualization.DataTable(jsonLineChartData);



    // Instantiate and draw our Line chart, passing in some options.

    var chart = new
google.visualization.LineChart(document.getElementById('center_column'));

    chart.draw(Linechartdata, {

      width: 968,

      height: 376,

        vAxis: {title: 'Moisture (%)', titlePosition:'out' },

        hAxis: {title: 'Time',slantedTextAngle: 70, titlePosition:'out' },

     chartArea: { left:"10%",top:"10%",width:"70%",height:"60%" }

    });




  }



  </script>

<meta charset="utf-8">
```

```
<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>

<body>

<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="Health.php">Health</a></li>

    </ul>
```

```
  </div>

  <div id="splash"><img src="images/moisture.jpg" width="968" height="376"
alt="" /></div>

 <form>

  <select name="users" onChange="drawItems(this.value)">

  <option value="">Select a Node:</option>

  <?php



$connection = pg_connect("host=localhost dbname=mydb user=tele
password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, "SELECT MAX(nodeid) as nodeid FROM
mda300_results ");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());



while ($row = pg_fetch_array($result)) {

echo '<option value="'.$row['nodeid'].'">'.$row['nodeid'].'</option>';
```

```
}




  ?>

  </select>

  <div id="center_column"></div>

</body>

</html>
```

Γ.12 Κώδικας health.php

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Wireless Sensor Networks</title>

<script src="http://use.edgefonts.net/poiret-one.js"></script>

<link href="styles/wsn_site.css" rel="stylesheet" type="text/css">

</head>



<body>
```

```
<div id="container">

  <div id="header">

  <table    align=right><tr><td><img    src="images/ouc.jpg"    width="60"
height="64" alt="" align=right /></td></tr></table>

  <table   align=right><tr><td><font   size="4"   face="Times   New   Roman"
color=white><b> O<font size=2>PEN </font><br> U<font size=2>NIVERSITY OF
</font><br>C<font size=2>YPROUS

      </font></font></td></tr></table>

    <h1>Wireless Sensor Networks</h1>

    <ul>

      <li><a href="index.html">Home</a></li>

      <li><a href="temperature.php">Temperature</a></li>

      <li><a href="light.php">Light</a></li>

      <li><a href="moisture.php">Moisture</a></li>

      <li><a href="health.php">Health</a></li>

    </ul>

  </div>

  <div id="splash"><img src="images/battery.jpg" width="968" height="376"
alt="" /></div>

  <div id="center_column">

    </div>

    <h2>Health</h2>
```

```
    <p>Σε αυτήν τη σελίδα μπορούμε να δούμε τις πληροφορίες για την κατάσταση του δικτύου μας .</p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p> </p>

    <p>

 <?php



$connection = pg_connect("host=localhost dbname=mydb user=tele password=tiny");

  if (!$connection) {

    print("Connection Failed.");

    exit;

  }

      $result = pg_exec($connection, " SELECT nodeid, count(health_pkts)as health_pkts, MIN(battery) as battery,sum(forwarded) as forwarded, sum(dropped) as dropped, MAX(retries) as retries FROM node_health GROUP BY nodeid");

    if (!$result)

            die("Error in SQL query: " . pg_last_error());

            echo "<table cellspacing=0 cellpadding=0 class=db-table align=center><tr><th width=58px>Node id</th><th width=150px>Health
```

```
Packets</th><th  width=58px>Battery</th><th  width=100px>Forwarded</th><th
width=80px>Dropped</th><th width=80px>Retries</th></tr>";




      while ($row = pg_fetch_array($result))

      {

    $battery=$row['battery']/10;



            echo  "<table  cellspacing=0  cellpadding=0  class=db-table
align=center><tr><td width=58px><center>" .$row['nodeid']. "</td>".

                  "<td    width=150px><center>"    .$row['health_pkts'].
"</td><td    width=58px><center>"    .substr($battery,0,3).    "</td><td
width=100px><center>"  .$row['forwarded'].  "</td><td  width=80px><center>"
.$row['dropped'].  "</td><td  width=80px><center>"  .  $row['retries']  .
"</td></tr></table>";

      }




?>

    </p>

  </div>
```

```
</div>

</body>

</html>
```