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

Ασύρματα δίκτυα αισθητήρων για τη μη επεμβατική παρακολούθηση βιοσήματος

Διπλωματική Εργασία

ΛΟΙΖΟΣ ΛΟΙΖΟΥ

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π

ΑΘΗΝΑ, ΜΑΙΟΣ 2008

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

Ασύρματα δίκτυα αισθητήρων για τη μη επεμβατική παρακολούθηση βιοσήματος

Διπλωματική Εργασία

ΛΟΙΖΟΣ ΛΟΙΖΟΥ

Επιβλέπων: Φίλιππος Κωνσταντίνου

Καθηγητής Ε.Μ.Π

ΑΘΗΝΑ, ΜΑΙΟΣ 2008

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

Ασύρματα δίκτυα αισθητήρων για τη μη επεμβατική παρακολούθηση βιοσήματος

Διπλωματική Εργασία

ΛΟΙΖΟΣ ΛΟΙΖΟΥ

Επιβλέπων: Φίλιππος Κωνσταντίνου

Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή
την ……Μαΐου/2008

……

Φ.Κωνσταντίνου	Ν.Ουζούνογλου	Χ.Καψάλης
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

ΑΘΗΝΑ, ΜΑΙΟΣ 2008

……

Λοΐζος Γ Λοΐζου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Ε.Μ.Π.

Περίληψη

Σκοπός αυτής της εργασίας είναι η μελέτη της ασύρματης μετάδοσης βιοσημάτων, μέσα από ένα ασύρματο δίκτυο αισθητήρων. Η τεχνολογία αυτή αναμένεται τα επόμενα χρόνια να βρει σημαντικό πεδίο εφαρμογών στον τομέα της υγείας και σε πολλούς άλλους τομείς. Στα πλαίσια της μελέτης αυτής, υλοποιήθηκε μια εφαρμογή κατά την οποία λαμβάνεται το σήμα της αναπνοής με επιταχυνσιόμετρο 2 αξόνων που δειγματοληπτείται από ένα mote και στη συνέχεια μεταδίδεται προς ένα σταθμό βάσης μέσω του ασύρματου δικτύου αισθητήρων και καταλήγει σε έναν ηλεκτρονικό υπολογιστή, όπου απεικονίζεται σε πραγματικό χρόνο και υπόκειται σε επεξεργασία. Παράλληλα, εκτός του επιταχυνσιόμετρου, υποστηρίζεται η
προσθήκη και άλλων αισθητήρων καταγραφής.

Η υλοποίηση της εφαρμογής έγινε με χρήση και κατάλληλο προγραμματισμό της ασύρματης πλατφόρμας Tmote Sky (Moteiv), η οποία περιγράφεται αναλυτικά και συγκρίνεται με μια σειρά από άλλες πλατφόρμες που υλοποιούν ασύρματα δίκτυα αισθητήρων. Η λειτουργία του Tmote Sky βασίζεται στο λειτουργικό σύστημα TinyOS.2-χ, σχεδιασμένο ειδικά για ενσωματωμένα συστήματα. Όλη η οργανωτική δομή του TinyOS, οι βιβλιοθήκες και οι εφαρμογές του είναι γραμμένες στη γλώσσα προγραμματισμού NesC. Το επιταχυνσιόμετρο που χρησιμοποιήσαμε ήταν αναλογικό διαξονικό της Memsic που περιγράφεται αναλυτικά.

Περιγράφονται τα τμήματα από τα οποία αποτελείται η εφαρμογή, ο τρόπος σύνδεσής τους καθώς και ο κώδικας που υλοποιήθηκε σε NesC, ενώ παρουσιάζονται και τα εργαλεία που χρησιμοποιήθηκαν για την απεικόνιση και αποθήκευση των σημάτων.

Περιγράφεται ακόμα ο κώδικας που υλοποιήθηκε σε Matlab για τη διαχείριση, απεικόνιση και επεξεργασία των δεδομένων που λαμβάναμε ασύρματα. Έγινε εφαρμογή του φίλτρου ‘κινούμενου μέσου όρου’, ενώ υλοποιήθηκαν και δύο αλγόριθμοι για την ανίχνευση των αναπνοών που εν συνεχεία συγκρίνονται και αξιολογούνται.

Στο τέλος αναφέρονται οι παρατηρήσεις επί των αποτελεσμάτων και δίνονται προτάσεις για περαιτέρω ανάπτυξη της εφαρμογής.

Λέξεις Κλειδιά

Ασύρματα δίκτυα αισθητήρων, ιατρικές εφαρμογές, Tmote Sky, TinyOS.2-X, NesC, Memsic, Zigbee, IEEE 802.15.4,ανίχνευση αναπνοών ,επιταχυνσιόμετρο, αναπνοή, άπνοια, Matlab, , ADC, Ecosensory, μετατροπέας, δειγματοληψία, επεξεργασία σήματος, φίλτρο κινούμενου μέσου όρου,ευαισθησία.

Abstract

The scope of this thesis is the study of Wireless sensor networks (WSN) and their use for non invasive biosignal monitoring. In particular we managed to receive the respiration signal using a dual axis analog accelerometer sample it in a specific frequency and transmit it through the radio to a Basestation leading to a computer. From there the signal is filtered processed and the results are evaluated.

We used the wireless sensor platform Tmote Sky by moteiv, which were programmed and set properly for the application. The orientation of the Tmote Sky is based on the TinyOS.2-X operating system, which has been designed for embedded systems. The structure of TinyOS ,its libraries and its applications are written in Nesc a programming language developed for this purpose.

The accelerometer used was analog dual axis by Memsic. Its fundamentals and a full description of the evaluation board are included.

After receiving the signal using the proper tools a moving average filter was used in order to improve its qualities and make processing easier. Two algorithms were developed in Matlab used to detect respiration in the signal. The two methods are compared and evaluated.

The thesis is concluded with noting on the results obtained and innovative ideas for expansion and further development are presented.

Key words

Wireless sensor Networks (WSN) ,Medical application, Tmote sky, Moteiv,TinyOS.2-X, nesC, Accelerometer, Memsic, Respiration, respiration detection, Matlab, A/D converted (ADC), sampling, signal processing, moving average filter, sensitivity, Ecosensory, zigbee, apnea.

Ευχαριστίες

Εκφράζω της θερμές μου ευχαριστίες προς τον καθηγητή του Ε.Μ.Π κ. Φίλιππο Κωνσταντίνου ,ο οποίος μου εμπιστεύτηκε την εκπόνηση της παρούσας διπλωματικής εργασίας και μου παρείχε όλη την απαραίτητη υποδομή για την επιτυχή διεκπεραίωση της. Ο υποδειγματικός τρόπος διδασκαλίας του, ο ζήλος και η αγάπη προς του φοιτητές του καθώς και οι αμέτρητες γνώσεις που μας παρείχε κατά τα χρονιά των σπουδών μας , μας εμφύσησαν την αγάπη το πάθος που χρειάζεται κάθε νέος άνθρωπος στο ξεκίνημα του καθώς και τη τεχνογνωσία και τη νοοτροπία που απαιτείται από κάθε μηχανικό.

Θερμές ευχαριστίες προς τον υποψήφιο διδάκτορα κ. Αλέξανδρο Καραγιάννη για τον απαράμιλλο ζήλο και διάθεση που επέδειξε κατά την όλη διαδικασία, την πολύτιμη βοήθεια και καθοδήγηση του σε κάθε σημείο της μελέτης με κόστος του δικού του πολυτίμου του χρόνου. Του εύχομαι τα καλύτερα για τη συνέχεια στη προσωπική και επαγγελματική του διαδρομή.

Θα ήθελα να ευχαριστήσω όλα τα μέλη και το προσωπικό του εργαστηρίου Συστημάτων Κινητών Επικοινωνιών για την ανοχή και την κατανόηση που έδειξαν κατά την συνύπαρξη μας στο εργαστήριο καθώς και την πολύτιμη βοήθεια τους.

Τέλος θερμές ευχαριστίες στην οικογένεια μου για την υποστήριξη και τη αγάπη που μου παρείχαν όλα αυτά τα χρόνια κατά τη συνέχεια των σπουδών μου.

Λοΐζος Γ.Λοΐζου.

Περιεχόμενα

Πίνακας εικόνων

Κεφάλαιο 1

Ασύρματα Δίκτυα Αισθητήρων (WSN) και εφαρμογές

Εισαγωγή

Την τελευταία δεκαετία η ελαχιστοποίηση μεγέθους και κόστους που επέφερε η τεχνολογία ημιαγωγών κατέστησε δυνατή τη δημιουργία υπολογιστών μικρότερων από το κεφάλι μιας καρφίτσας με τεράστια υπολογιστική ισχύ και με κόστος που τους καθιστά αναλώσιμους. Ταυτόχρονες εξελίξεις στον τομέα των ασυρμάτων επικοινωνιών ,σχεδίασης αισθητήρων και αποθήκευσης ενέργειας οδήγησαν στην υλοποίηση των WSN (Wireless Sensor Networks). Τα βασικά στοιχεία αυτών των δικτύων είναι ολοκληρωμένοι μικροαισθητήρες σε μέγεθος μερικών χιλιοστών με δυνατότητες επεξεργασίας και ασύρματης εκπομπής δεδομένων . Ήδη σε μεγάλο εύρος εφαρμογών έχει προταθεί και υλοποιηθεί και αναμένετε να προκαλέσει σημαντικές αλλαγές στη καθημερινή μας ζωή.

Μια από τις πρώτες εφαρμογές που αναπτύχθηκε από το πανεπιστήμιο του Berkley υπό τη χρηματοδότηση του DAPRA με υλοποίηση Ασύρματου δικτύου Αισθητήρων μεγάλης κλίμακας και ονομάστηκε Smart Dust (έξυπνη σκόνη).

Εικόνα 1-Smart Dust.Πολλοί αισθητήρες που επικοινωνούν με ένα σταθμό βάσης.

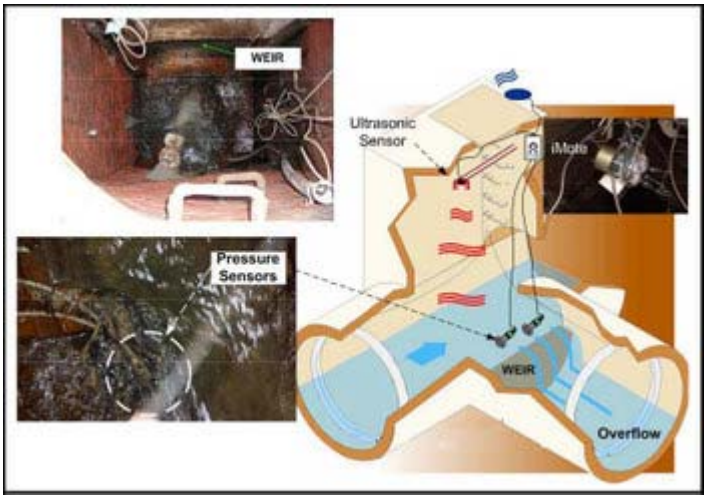
Στόχος του προγράμματος αυτού ήταν η δημιουργία μιας αυτόνομης πλατφόρμας σε χιλιομετρική κλίμακα αισθητήρων και επικοινωνιών για μαζικά διανεμημένα δίκτυα αισθητήρων .Το Smart Dust προοριζόταν αρχικά για τη εξ'αποστάσεως παρακολούθηση εχθρικών στρατευμάτων από το στρατό μέσω χιλιάδων ασύρματων μικροαισθητήρων "motes" διασκορπισμένων στο πεδίο της μάχης.

Εκτός από τις στρατιωτικές εφαρμογές το Smart Dust βρήκε πληθώρα εφαρμογών όπως για παρακολούθηση των ατμοσφαιρικών και καιρικών συνθηκών. Αξιοσημείωτο είναι η βιοτεχνολογική προσέγγιση τις ιδέας με motes από χημικά συστατικά αντί για ηλεκτρονικά κυκλώματα.

Ένα βασικό συστατικό των WSN είναι το μικρό, ανοικτού κώδικα ,ενεργειακά αυτόνομου λειτουργικού συστήματος γνωστού ως Tiny Micro threading Operating System ή TinyOS που αναπτύχθηκε στα εργαστήρια του πανεπιστημίου του Berkley.Το λειτουργικό αυτό προσφέρει το βασικό πλαίσιο και αναπτυξιακό περιβάλλον για τα WSN και λειτουργεί σε συνάρτηση με τη ισχύ το μέγεθος και το κόστος. Το TinyOS ελέγχει τόσο τον εξοπλισμό όσο και το δίκτυο, κάνοντας ταυτόχρονα μετρήσεις ,αποφάσεις δρομολογήσεις και έλεγχο και εξοικονόμηση ενέργειας.

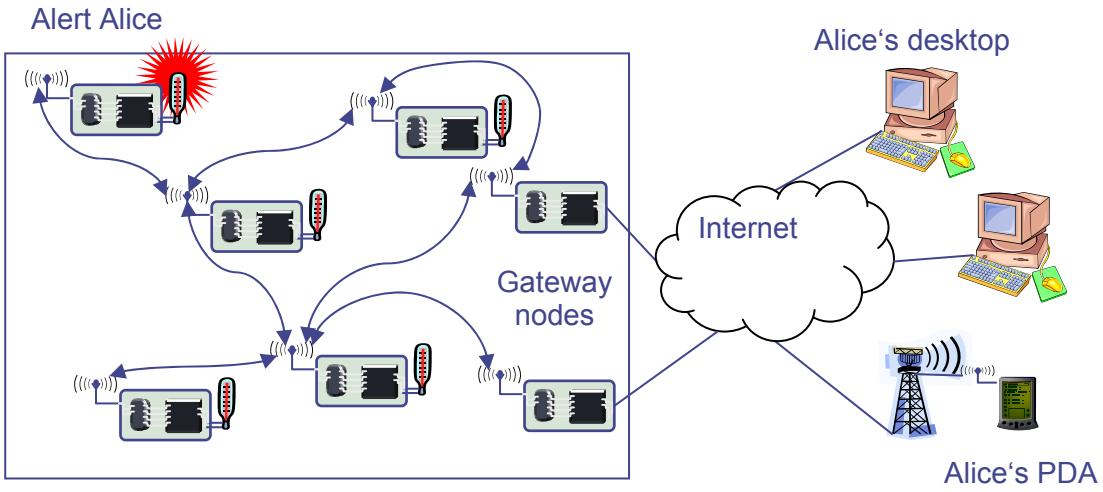
Σήμερα οι εφαρμογές των WSN χωρίζονται σε 3 κατηγορίες ων ακολούθως

1. Εφαρμογές που αφορούν την παρακολούθηση του περιβάλλοντος (εσωτερικού, εξωτερικού ,αστικού ή υπαίθριου).
2. Παρακολούθηση αντικειμένων(μηχανών και κτιρίων)
3. Παρατήρηση της αλληλεπίδρασης και της σχέσης μεταξύ αντικειμένων και περιβάλλοντος.

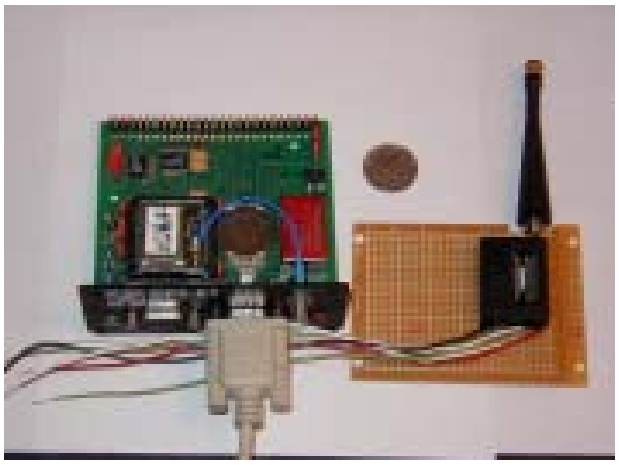


Εικόνα 2-WSN για την έγκαιρη ανίχνευση προβλημάτων σε αγωγούς

Για παράδειγμα πολυεθνικές εταιρίες όπως η British Petroleum (BP) συνειδητοποίησαν τις τεράστιες προοπτικές των WSN τεχνολογιών και έχουν επενδύσει στην ανάπτυξη τους σε μεγάλη κλίμακα. Σε μια πειραματική εφαρμογή μετρούσαν κατά τη διάρκεια των γεωτρήσεων τους τις μη φυσιολογικές δονήσεις και προειδοποιούν του μηχανικούς για πιθανή επερχόμενη βλάβη του εξοπλισμού BP στοχεύει στη χρήση των WSN για την εξ' αποστάσεως παρακολούθηση του επιπέδου πληρότητας των δεξαμενών υγραερίου. Με τη χρήση υπερηχητικού αισθητήρα στον πάτο της δεξαμενής μετράτε η πληρότητα και εν συνεχεία εκπέμπετε μέσω δορυφόρου χαμηλής τροχιάς σε ένα σταθμό βάσης με αποτέλεσμα να ενημερώνονται οι πελάτες πριν να τελειώσουν τα αποθέματα τους. Το να επιτύχεις τέτοιου είδους κάλυψη με ενσύρματα μέσα δεν θα ήταν απλά μη συμφέρον οικονομικά και δύσκολο να υλοποιηθεί αλλά ανέφικτο.



Το πανεπιστήμιο του Princeton έχει εφαρμόσει ένα άλλο το “Zebranet” με το οποίο παρακολουθείται η μετανάστευση, η συνύπαρξη με άλλα είδη και η νυχτερινή συμπεριφορά των πληθυσμών ζέμπρας στην Αφρική. Αυτό το εγχείρημα κατέστη δυνατό μόνο μέσω ενός Ad hoc WSN με το κατάλληλο εύρος ζώνης και ισχύ επεξεργασίας.



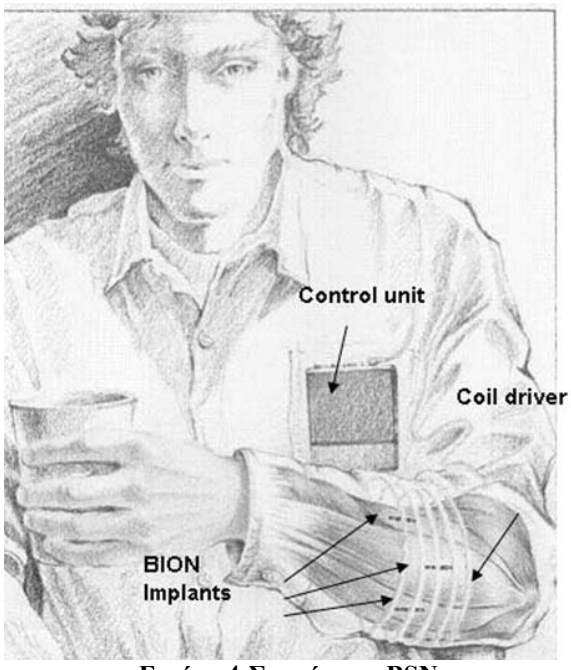
Εικόνα 3-Σύστημα εντοπισμού που αναπτύχθηκε στο Berkley

Το σύστημα εντοπισμού που χρησιμοποιήθηκε από το Berkley στην εξέλιξη του smart dust.

Ενώ η τεχνολογία ασύρματων δικτύων συνεχίζει να εξελίσσετε και να βρίσκει εφαρμογές σε όλο και ευρύτερο πεδίο η πρόκληση βρίσκετε στην ανάπτυξη των λεγόμενων προσωπικών δικτύων τα οποία αφορούν την τηλεϊατρική παρακολούθηση του ανθρώπινου σώματος. Το ανθρώπινο σώμα αποτελείται από ένα πολύπλοκο εσωτερικό περιβάλλον το οποίο αποκρίνεται και επιδρά με το εξωτερικό περιβάλλον. Με τοποθέτηση των αισθητήρων πάνω στο σώμα ή και χειρουργικά μέσα σε αυτό επιτυγχάνεται η τηλεϊατρική παρακολούθηση του ανθρώπινου οργανισμού μέσω του ασύρματου δικτύου. Επί της ουσίας το περιβάλλον ανθρώπινου σώματος είναι μικρής κλίμακας και απαιτεί διάφορους τύπους παρακολούθησης και συχνοτήτων το οποίο καθιστά τα προσωπικά δίκτυα διαφορετικά από τα άλλα WSN. Αυτές οι απαιτήσεις οδηγούν στην ανάπτυξη των γνωστών wireless Body Sensor Area Network(BSN) ή patient Personal Area Network(pPAN).

Ασύρματα δίκτυα αισθητήρων στην περιοχή του ανθρωπίνου σώματος (BSN)

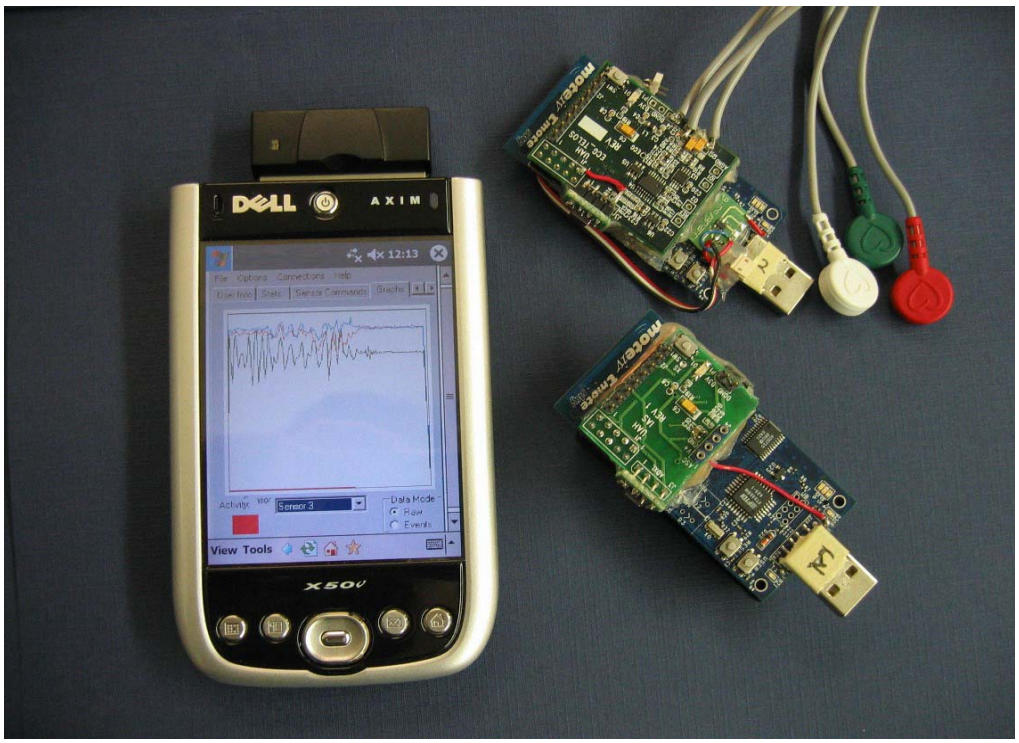
Το σύγχρονο τεχνολογικό επίπεδο έχει προσφέρει στην κοινωνία καινοτομίες σε όλους τους τομείς της ζωής. Τέτοιες καινοτομίες στα πλαίσια της εφαρμογής και ενσωμάτωσης νέων τεχνολογιών παρατηρούνται και στον τομέα της υγείας και της ευρύτερης ιατρικής. Οι υπηρεσίες της τηλεϊατρικής προσανατολίζονται στην κατά το δυνατό αποδέσμευση του ασθενούς (χρήστη) από τους νοσοκομειακούς περιορισμούς στα πλαίσια των ελευθεριών που παρέχονται από τα συστήματα κινητών και προσωπικών επικοινωνιών. Η βασική επιδίωξη των υπηρεσιών αυτών είναι ο διαρκείς εξ αποστάσεως έλεγχος της κατάστασης της υγείας του ασθενούς μέσω της συλλογής επεξεργασίας , αξιολόγησης, αξιοποίησης και αποθήκευσης κατάλληλης πληροφορίας.



Εικόνα 4-Συστήματα BSN

Η καινοτομία τους έγκειται στις συνθήκες πλήρους κινητικότητας που παρέχουν στους χρήστες τους σε συνδυασμό με την αίσθηση ασφάλειας που συνεπάγεται η διαρκής αλλά ταυτόχρονα διακριτική και μη παρεμβατική παρακολούθηση της υγείας τους.

Η αλματώδης ανάπτυξη της ηλεκτρονικής και των αισθητήρων σε συνδυασμό με της πλήρη εξάπλωση και εξάρτηση μας από τις ευρέως διαθέσιμες υποδομές ασυρμάτων και κινητών επικοινωνιών κατέστησαν σήμερα δυνατή την υλοποίηση απεριόριστων εφαρμογών και υπηρεσιών στον τομέα των BSN.



Εικόνα 5-Ασύρματο δίκτυο αισθητήρων με τελικό αποδέκτη ένα PDA

Φορητά συστήματα ελέγχου υγείας που ενσωματώνονται σε ένα σύστημα τηλεϊατρικής είναι η νέα τεχνολογία πληροφοριών που μπορεί να υποστηρίξει την έγκαιρη ανίχνευση μη φυσιολογικών καταστάσεων και να προλάβει τις συνέπειες τους. Πολλοί ασθενείς μπορούν να ωφεληθούν από τη συνεχή παρακολούθηση σαν μέρος μιας διαγνωστικής διαδικασίας, τη βέλτιστη συντήρηση από μια χρόνια νόσο ή κατά τη διάρκεια εποπτευομένης αποκατάστασης από μια χειρουργική διαδικασία. Σημαντικοί περιορισμοί για την ευρύτερη αποδοχή των ήδη υπαρκτών συστημάτων για το συνεχή έλεγχο είναι:

α)τα πολλά καλώδια μεταξύ των αισθητήρων και μιας μονάδας επεξεργασίας,

β) έλλειψη ολοκληρωμένων συστημάτων των μεμονωμένων αισθητήρων,

γ) παρεμβολές στα κοινά ασύρματα κανάλια επικοινωνίας

δ) ανύπαρκτη υποστήριξη για την συλλογή μεγάλου όγκου δεδομένων

Παραδοσιακά συστήματα ιατρικού έλεγχο, όπως τα όργανα Holter έχουν χρησιμοποιηθεί μόνο για τη συλλογή στοιχείων χωρίς τη δυνατότητα ταυτόχρονης επεξεργασίας. Συστήματα με πολλαπλούς αισθητήρες που χρησιμοποιούνται για τη φυσική αποκατάσταση έχουν πολλά καλώδια μεταξύ των ηλεκτροδίων και του ελέγχου συστήματος ελέγχου τα οποία περιορίζουν τη δραστηριότητα του ασθενή και το επίπεδο άνεσης και έτσι επηρεάζονται αρνητικά και τα μετρούμενα αποτελέσματα. Μια φορητή συσκευή ελέγχου υγείας χρησιμοποιεί το προσωπικό δίκτυο περιοχής σώματος και μπορεί να ενσωματωθεί στον ιματισμό του χρήστη. Αυτή η οργάνωση συστημάτων, εντούτοις, είναι ακατάλληλη για μεγάλο, συνεχή έλεγχο, ιδιαίτερα κατά τη διάρκεια κανονικής δραστηριότητας ,εντατική εξάσκηση ή με υπολογιστικά υποβοηθουμένη αποκατάσταση.

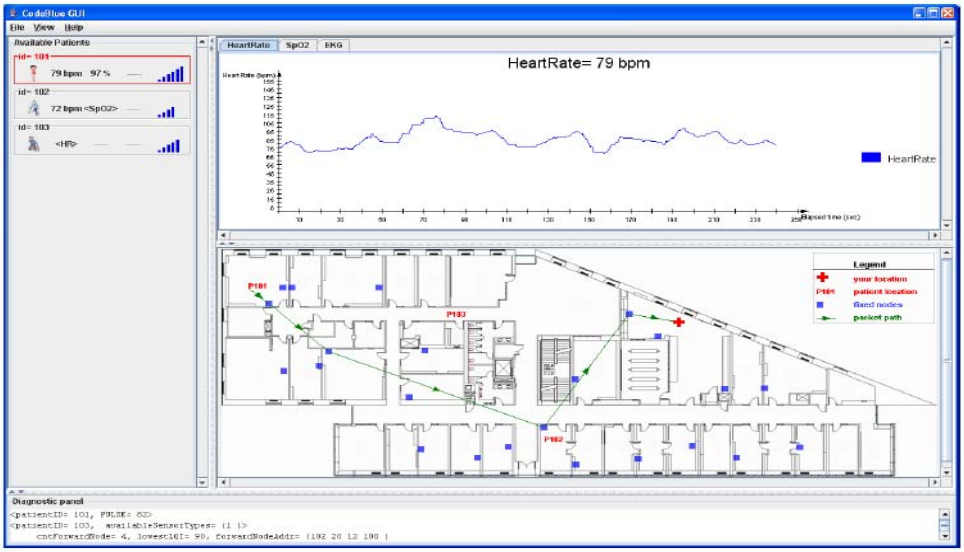
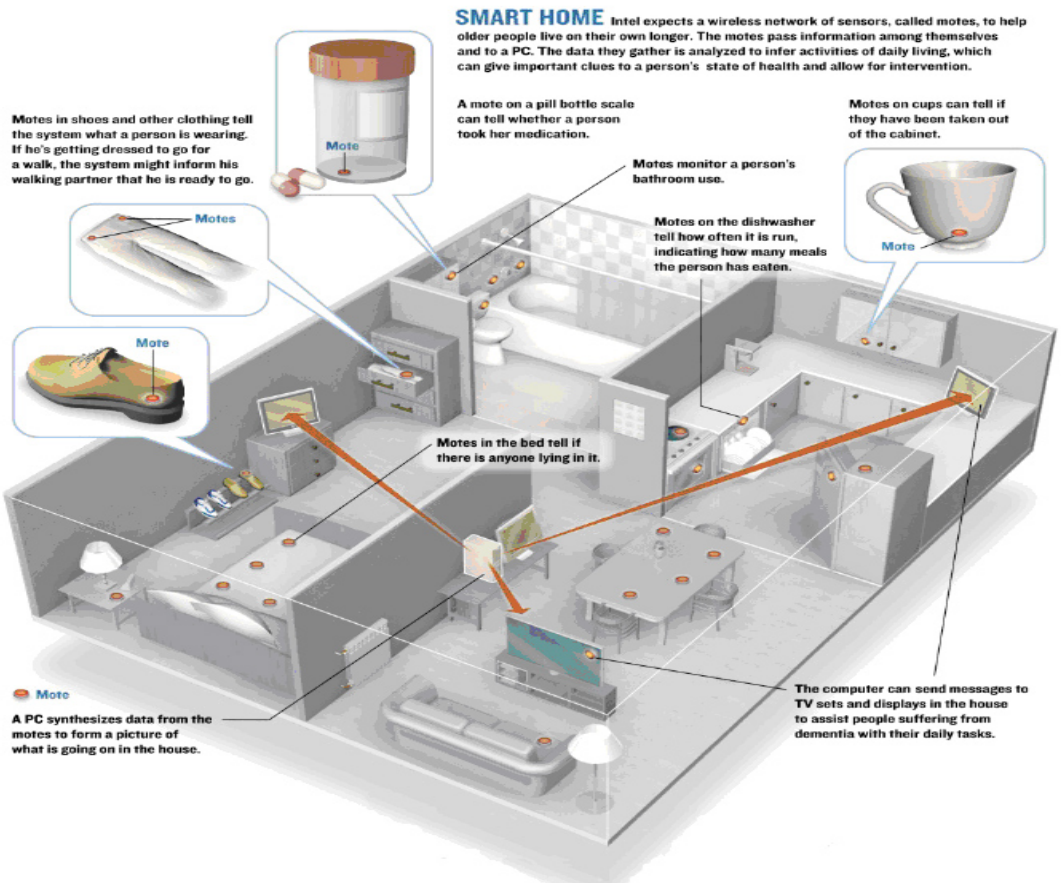


Figure 9: **The CodeBlue user interface.** *This is an actual screenshot of the CodeBlue GUI running in our building with three patient sensors reporting data to a laptop.*

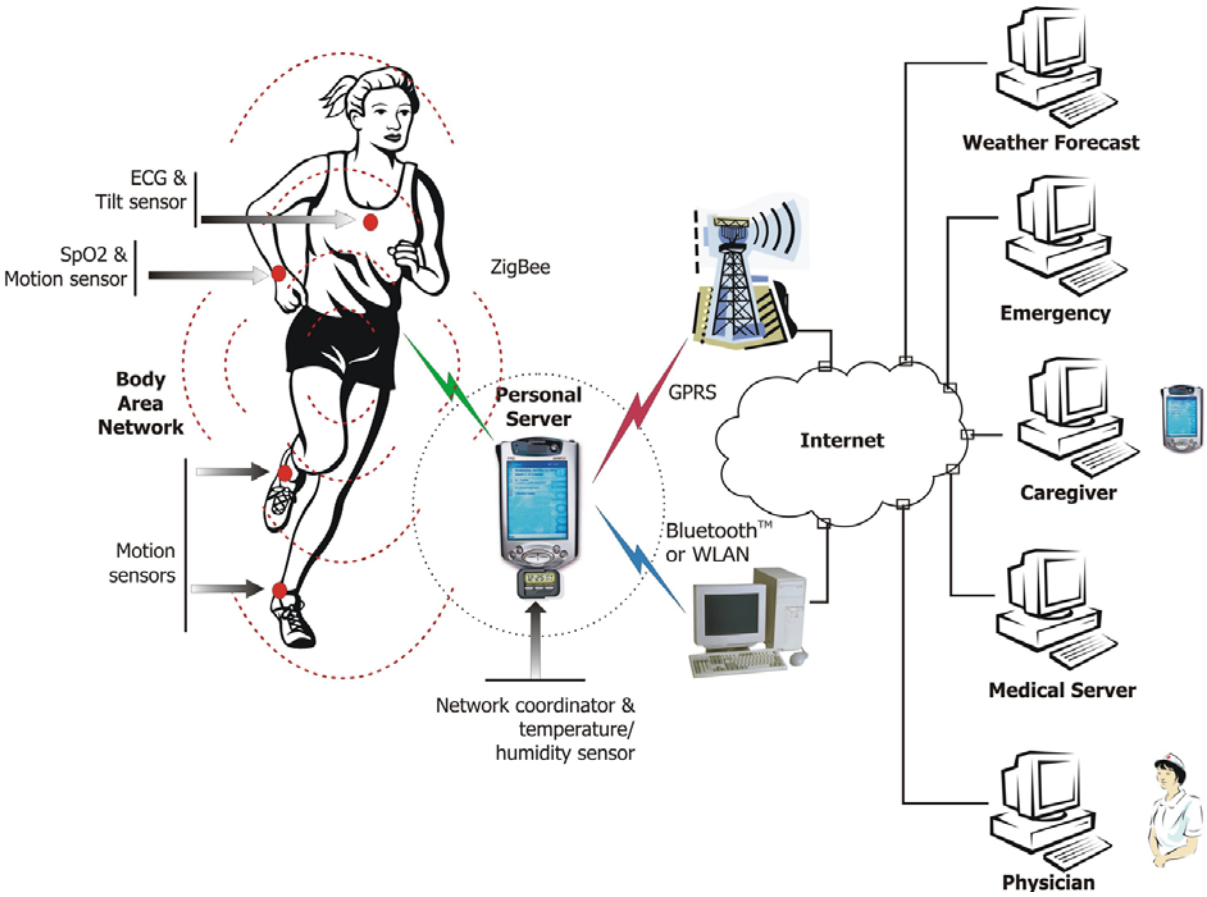
Εικόνα 6-Το interface της εφαρμογής Code Blue

Πρόσφατες πρόοδοι τεχνολογίας στην ασύρματη δικτύωση ,τη μικροϋπολογιστή επεξεργασία την ολοκλήρωση φυσικών αισθητήρων, ενσωματωμένων μικροελεγκτών και διεπαφών σε ένα ενιαίο κύκλωμα υπόσχονται μια νέα γενεά ασύρματων αισθητήρων κατάλληλων για πολλές εφαρμογές .Εντούτοις, οι υπάρχουσες τηλεμετρικές συσκευές χρησιμοποιούν ασύρματα κανάλια επικοινωνίας για να μεταφέρουν αποκλειστικά μη επεξεργασμένα δεδομένα από τους αισθητήρες στο σταθμό ελέγχου, ή υψηλού επιπέδου πρωτόκολλα όπως το Bluetooth τα οποία είναι σύνθετα, πολύπλοκα και ενεργοβόρα . Ακόμα είναι επιρρεπής σε παρεμβολές από άλλες συσκευές που λειτουργούν στην ίδια ζώνη συχνοτήτων. Αυτά τα χαρακτηριστικά περιορίζουν τη χρήση τους για παρατεταμένο ιατρικό έλεγχο. Άλλα απλά, ακριβή μέσα εκτός εργαστηρίου δεν είναι διαθέσιμα προς το παρόν. Μόνο εκτιμήσεις μπορούν να ληφθούν από ερωτηματολόγια, μετρήσεις του καρδιογραφήματος, πεδιομέτρων ή επιταχυνσιομέτρων.



Εικόνα 7

Η αυξανόμενη δύναμη επεξεργασίας συστημάτων επιτρέπει την πολύπλοκη επεξεργασία δεδομένων σε πραγματικό χρόνο μέσα στα όρια του συστήματος. Κατά συνέπεια ένα τέτοιο σύστημα μπορεί να υποστηρίξει βιοανάδραση και να δώσει προειδοποίηση για τυχόν προβλήματα. Η χρήση των τεχνικών βιοανάδρασης έχει κερδίσει την προσοχή μεταξύ των ερευνητών στον τομέα της φυσικής ιατρικής και διάγνωσης και παρακολούθησης εξ αποστάσεως των ασθενών. Τα εντατικά προγράμματα παρακολούθησης των ασθενών έχουν αποδειχτεί σημαντικά για την λειτουργίας της μηχανικά υποβοηθούμενης αποκατάστασης των ασθενών.



Εικόνα 8

Τα φορητά συστήματα τεχνολογίας και βιοανάδρασης εμφανίζονται να είναι έγκυρη λύση, δεδομένου ότι μειώνουν τον χρόνο προετοιμασίας του ασθενής πριν από κάθε σύνοδο και απαιτούν λιγότερο χρόνο συμμετοχής των παθολόγων και των θεραπόντων. Η φορητή ασύρματη τεχνολογία επιτρέπει τους αισθητήρες να είναι τοποθετημένοι στον ασθενή για μεγάλες περιόδους, επομένως εξαλείφετε η ανάγκη τοποθέτησης σε κάθε σύνοδο. Αντί αυτού, ένας προσωπικός κεντρικός

υπολογιστής όπως ένα PDA μπορεί σχεδόν αμέσως να εκκινήσει μια νέα περίοδο άσκησης όποτε ο ασθενής είναι έτοιμος και πρόθυμο να αρχίσει. Εκτός από αποκατάσταση στον προσωπικό χώρο, αυτή η ρύθμιση μπορεί επίσης να είναι ευεργετική μέσα στο ευρύτερο κλινικό πλαίσιο, όπου ο πολύτιμος χρόνος των παθολόγων και οι θεράποντες θα μπορούσαν να εξοικονομηθούν. Επιπλέον, το σύστημα μπορεί να δώσει έγκαιρες προειδοποιήσεις ή συναγερμό στον ασθενή, ή σε μία εξειδικευμένη ιατρική υπηρεσία απάντησης σε περίπτωση σημαντικών αποκλίσεων από τα φυσιολογικά ή σε καταστάσεις έκτακτων ιατρικών αναγκών.

Χαρακτηριστικά παραδείγματα πιθανών εφαρμογών περιλαμβάνουν το εγκεφαλικό, τη φυσική αποκατάσταση στο χώρο του ασθενούς μετά από χειρουργικές επεμβάσεις, ανάρρωση από μυοκαρδιακό έμφραγμα ,αποκατάσταση τραυματισμών εγκεφάλου. Η αξιολόγηση και αποτελεσματικότητα των διαδικασιών αποκατάστασης έχει περιοριστεί σε εργαστηριακό επίπεδο και λίγα είναι γνωστά για αποκατάσταση υπό τις πραγματικές συνθήκες. Η μικρή σε μέγεθος ,ασύρματη, φορητή τεχνολογία προσφέρει μια τεράστια ευκαιρία για να αντιμετωπίσει αυτό το ζήτημα.

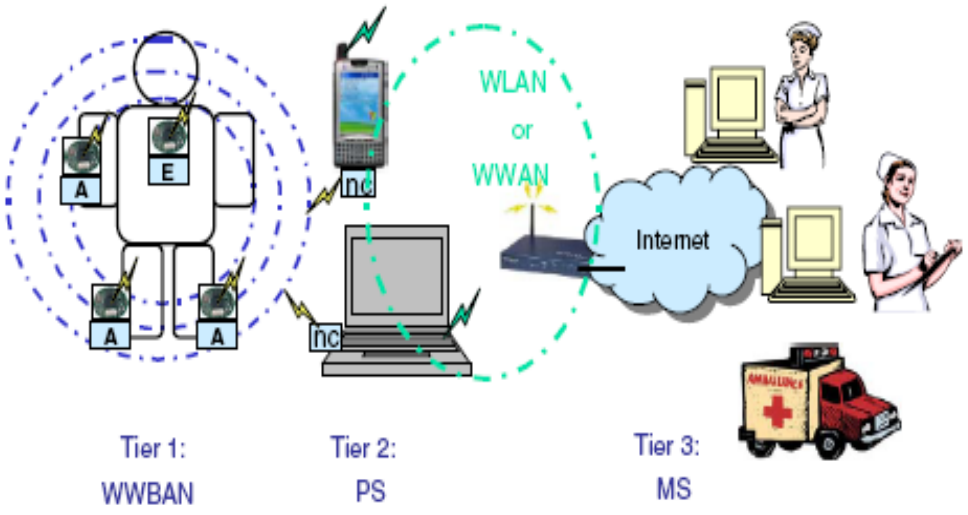


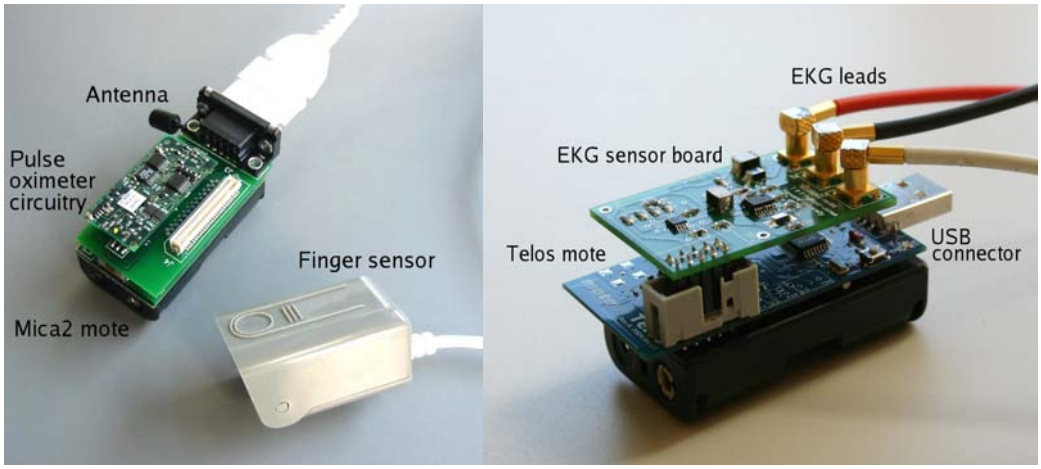
Fig. 1. WWBAN integrated into a telemedical system for health monitoring.

Εικόνα 9-εφαρμογές των WSN στον χώρο της υγείας.

Μια τέτοια εφαρμογή είναι το ActiS, βασισμένο σε μία τυποποιημένη ασύρματη πλατφόρμα αισθητήρων με μια εξειδικευμένη βάση για μονοκαναλικό

βιοενισχυτής και δύο επιταχύνσιόμετρα. Σαν αισθητήρα καρδιάς το ActiS μπορεί να χρησιμοποιηθεί για να ελέγξει τη καρδιακή λειτουργία και τη θέση του ανώτερου κορμού. Ο ίδιος αισθητήρας μπορεί χρησιμοποιηθεί στη παρακολούθηση της θέσης και λειτουργίας των πάνω και κάτω ακροτάτων .Ένα φορητό σύστημα με το ActiS επιτρέπει την πρόσβαση στον μεταβολικό ρυθμό και να δώσει τη συσσωρεμένη καταναλισκομένη ενέργεια σαν παράμετρο διαχείρισης πολλών ιατρικών καταστάσεων.

Μια αρχική έκδοση του ActiS έχει βασιστεί σε ευφυή ασύρματους αισθητήρες και ειδικά ασύρματα πρωτόκολλα αισθητήρων στην ελεύθερη ζώνη συχνοτήτων 900 MHz για επιστημονικά και ιατρικά όργανα .Η πρόσφατη εισαγωγή IEEE προτύπων για χαμηλής ισχύος προσωπικά δίκτυα περιοχής (802.15.4) και η ZigBee λίστα πρωτοκόλλου καθώς επίσης και το νέο ZigBee συμβατό σε Telos πλατφόρμα αισθητήρων οδήγησε στην ανάπτυξη νέων συστημάτων. Η Υποστήριξη **TinyOS** για την επιλεγμένη πλατφόρμα αισθητήρων διευκολύνει τη γρήγορη εφαρμογή και ανάπτυξη . Τυποποιημένη αρχιτεκτονική υλικού και λογισμικού που να διευκολύνει τα λειτουργικά συστήματα και τις συσκευές αναμένεται για να επηρεάσει σημαντικά την επόμενη γενεά συστημάτων υγείας .Αυτή η τάση μπορεί επίσης να παρατηρηθεί στα πρόσφατα αναπτυγμένα φυσιολογικά συστήματα οργάνων ελέγχου από το Harvard and Welch-Allen .



Εικόνα 10 –Το σύστημα Codeblue από το Harvard

Αρχιτεκτονική συστημάτων

Συνεχείς τεχνολογικές πρόοδοι στα ολοκληρωμένα κυκλώματα, την ασύρματη επικοινωνία, και τους αισθητήρες επιτρέπουν την ανάπτυξη μικροσκοπικών, μη επεμβατικών αισθητήρων που επικοινωνούν ασύρματα με έναν προσωπικό κεντρικό υπολογιστή και στη συνέχεια μέσω του Διαδικτύου με μια μακρινή βάση έκτακτων αναγκών ,καιρικών προβλέψεων ή ιατρικό κεντρικός υπολογιστής βάσεων δεδομένων που χρησιμοποιεί βασική γραμμή (ιατρική βάση δεδομένων), αισθητήρες (WBAN) και περιβαλλοντικές πληροφορίες (πρόβλεψη έκτακτης ανάγκης ή καιρού), με αλγόριθμους που μπορούν να οδηγήσουν στην εξακρίβωση της κατάστασης του ασθενούς και την εξαγωγή συγκριμένων οδηγιών προς αυτούς.

Ο προσωπικός κεντρικός υπολογιστής, που τρέχει σε ένα PDA ή ένα κινητό τηλέφωνο τρίτης γενιάς, παρέχει την διεπαφή ανθρώπου-υπολογιστή και επικοινωνεί με το μακρινό κεντρικό υπολογιστή. Το σχήμα 1 παρουσιάζει μια γενικευμένη επισκόπηση μιας multi-tier αρχιτεκτονικής συστημάτων όπου το χαμηλότερο επίπεδο καλύπτει ένα σύνολο ευφυείς φυσικούς αισθητήρες το δεύτερο επίπεδο είναι ο προσωπικός κεντρικός υπολογιστής (ΔιαδίκτυοPDA, κινητό τηλέφωνο, ή προσωπικός υπολογιστής) και το τρίτο επίπεδο καλύπτει το δίκτυο της μακρινής υγειονομικής περίθαλψης, κεντρικοί υπολογιστές και σχετικές υπηρεσίες (νοσοκόμος, παθολόγος, κλινική ,Έκτακτη ανάγκη, καιρός). Κάθε επίπεδο αντιπροσωπεύει ένα αρκετά σύνθετο υποσύστημα με τοπική ιεραρχία που υιοθετείται για να εξασφαλίσει αποδοτικότητα, φορητότητα, ασφάλεια, και μειωμένο κόστος

Επίπεδο αισθητήρων

Ένα wBSN μπορεί να περιλάβει διάφορους φυσικούς αισθητήρες ανάλογα με την εφαρμογή και τους χρήστες. Πληροφορίες από διάφορους αισθητήρες μπορούν να συνδυαστούν για να παραγάγουν τις νέες πληροφορίες όπως οι συνολικές ενεργειακές δαπάνες. Ένα εκτενές σύνολο φυσικών αισθητήρων μπορεί να περιλαμβάνει τα εξής:

- "ένας αισθητήρας ECG (ηλεκτροκαρδιογραφημάτων) για τον έλεγχο της καρδιακής δραστηριότητα

- "ένας αισθητήρας EMG (ηλεκτρομυογραφία) για τον έλεγχο της δραστηριότητας των μυών.
- "ένας αισθητήρας EEG (ηλεκτροεγκεφαλογραφήματος) για τον έλεγχο ηλεκτρική δραστηριότητα εγκεφάλου
- "ένας αισθητήρας πίεσης αίματος
- "ένας αισθητήρας κλίσης για τον έλεγχο της θέσης κορμών
- "ένας αισθητήρας αναπνοής για τον έλεγχο της αναπνοής
- "αισθητήρες μετακίνησης που χρησιμοποιούνται για να υπολογίσουν τη δραστηριότητα του χρήστη
- "αισθητήρας έξυπνων καλτσών" ή μια εξοπλισμένη με αισθητήρες σόλα παπουτσιών για να σκιαγραφήσει τις φάσεις μεμονωμένων βημάτων

Αυτοί οι αισθητήρες παράγουν τα χαρακτηριστικά αναλογικά σήματα που διασυνδέονται στις τυποποιημένες ασύρματες πλατφόρμες δικτύων που παρέχουν υπολογιστική επεξεργασία, αποθήκευση, και ικανότητες επικοινωνίας. Πολλαπλοί αισθητήρες μπορούν να μοιραστούν έναν ενιαίο ασύρματο κόμβο δικτύων. Επιπλέον οι αισθητήρες μπορούν να διασυνδεθούν με μια ευφυή πλατφόρμα αισθητήρων που παρέχει την δυνατότητα επεξεργασίας δεδομένων από τους αισθητήρες και επικοινωνεί με ένα τυποποιημένο ασύρματο δίκτυο μέσω των τμηματικών διεπαφών.

Οι ασύρματοι κόμβοι αισθητήρων πρέπει να ικανοποιήσουν τις ακόλουθες απαιτήσεις: ελάχιστο βάρος, μικροσκοπική μορφή, χαμηλή ισχύς λειτουργία για να επιτρέψει παρατεταμένη λειτουργία συνεχής ένταξη σε ένα WBAN, πρωτόκολλα διεπαφών, συγκεκριμένη βαθμονόμηση στο επίπεδο του ασθενούς ,συντονισμός και προσαρμογή. Αυτές οι απαιτήσεις αποτελούν ένα προκλητικό στόχο, αλλά κρίσιμο εάν θέλουμε να πάμε πέρα από απαρχαιωμένα συστήματα στην υγειονομική περίθαλψη όπου ένας προμηθευτής δημιουργεί όλα τα προϊόντα. Μόνο υβριδικά συστήματα με αυτό εφαρμοσμένα τμήματα υλικού και λογισμικού, που κατασκευάζονται από διαφορετικό προμηθευτής υπόσχονται τον πολλαπλασιασμό και τη δραματική μείωση κόστους .

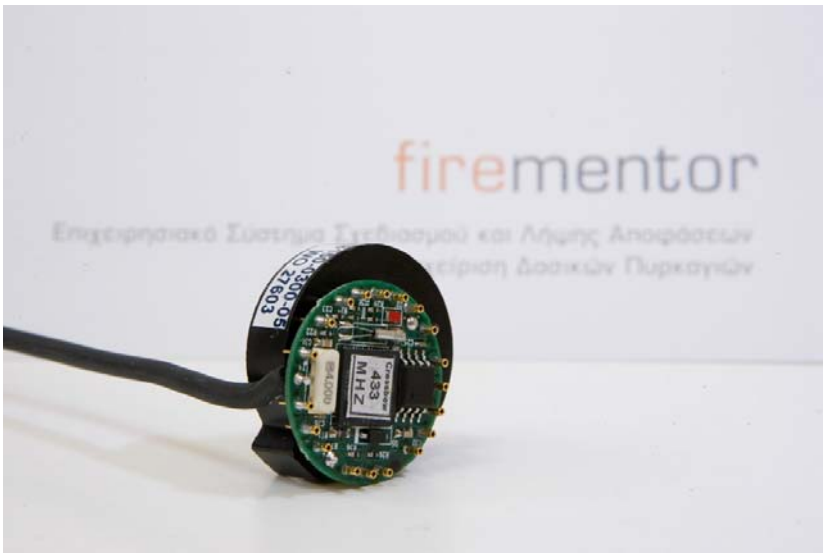
Οι ασύρματοι κόμβοι δικτύων μπορούν να εφαρμοστούν ως μικροσκοπικά μπαλώματα ή ενσωματωμένοι στα ενδύματα ή τα παπούτσια. οι κόμβοι δικτύου συλλέγουν συνεχώς και επεξεργάζονται τις πληροφορίες, τις αποθηκεύουν τοπικά, και τις στέλνουν στον κεντρικό υπολογιστή. Ο τύπος και η φύση μιας εφαρμογής υγειονομικής περίθαλψης καθορίζουν τη συχνότητα των σχετικών διεργασιών (δειγματοληψία, επεξεργασία, αποθήκευση, και επικοινωνία). Ιδανικά οι αισθητήρες διαβιβάστε τη θέση και τα δεδομένα τους περιοδικά επομένως μειώνουν σημαντικά την κατανάλωση ισχύος και να παρατείνουν τη ζωή των μπαταριών. Όταν η τοπική ανάλυση των στοιχείων είναι αναποτελεσματική ή δείχνει μια κατάσταση έκτακτης ανάγκης, το ανώτερο επίπεδο της ιεραρχίας μπορεί να εκδώσει ένα αίτημα να μεταφερθούν τα αρχικά σήματα στα ανώτερα επίπεδα όπου η προηγμένη επεξεργασία και αποθήκευση είναι διαθέσιμα.

Εφαρμογές WSN

Η ολοένα και μεγαλύτερη ανάπτυξη των ασύρματων δικτύων αισθητήρων έχει οδήγηση σε ένα μεγάλο εύρος εφαρμογών. Εφαρμογές που η κάθε μια θέτει τις δικές της παραμέτρους στην υλοποίηση όπως είναι ο τύπος του φαινόμενου προς παρατήρηση, ο όγκος των δεδομένων ,η κρισιμότητα της πληροφορίας .

Στην κατηγορία των περιβαλλοντικών εφαρμογών έχουμε

- Την καταγραφή της εξελικτικής διαδικασίας ενός οικοσυστήματος (υδάτινου, χερσαίου, δασικού, αστικού)



Εικόνα 11-Ασύρματος αισθητήρας τύπου crossbow για την εφαρμογή firementor

- Την καταγραφή του μικροκλίματος σε εργασιακούς χώρους και άλλες μεγάλες εγκαταστάσεις για τη βελτιστοποίηση της χρήσης των κλιματιστικών συστημάτων .

- Πρόληψη και ανίχνευση εκδήλωσης φωτιάς σε υπαίθριους ή κλειστούς χώρους.

- Παρακολούθηση της εξέλιξης γεωργικών καλλιεργειών

- Καταγραφή γεωφυσικών φαινόμενων.

Στην κατηγορία των εφαρμογών οικιακού αυτοματισμού έχουμε

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

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

Εφαρμογές ασφαλείας

- Παρακολούθηση χώρων και για λόγούς ασφαλείας και ενημέρωση κάποιας εποπτεύουσας εφαρμογής σε τακτά χρονικά διαστήματα ή όταν λάβει χώρα ένα περιστατικό ενδιαφέροντος, όπως παραβίαση χώρου ή πυρκαγιά.

Στρατιωτικές εφαρμογές

- Έλεγχος των κινήσεων του αντιπάλου

Εικόνα 12-Το στρατιωτικό σύστημα υποβήθησης στρατιωτών SaS

- Διαφύλαξη της ασφάλειας μίας περιοχής

Εικόνα 13- Στιγμιότυπο απο το στρατιωτικό σύστημα υποβοήθησης SAS

Στιγμιότυπο του συστήματο υποβήθησης στρατιωτών(SAS)

- Απομακρυσμένο έλεγχο υλικού

Εφαρμογές αντιμετώπισης φυσικών καταστροφών

- Πρόληψη και διάγνωση σεισμικής δραστηριότητας
- Εντοπισμός παγιδευμένων ατόμων(Ray et al)

- Οι Kattapali et al ενσωματώνουν σε κατάλληλα σημεία κατά την κατασκευή υποδομών αισθητήρες που καταγράφου την καταπόνηση του υλικού και τη μετακίνηση του.

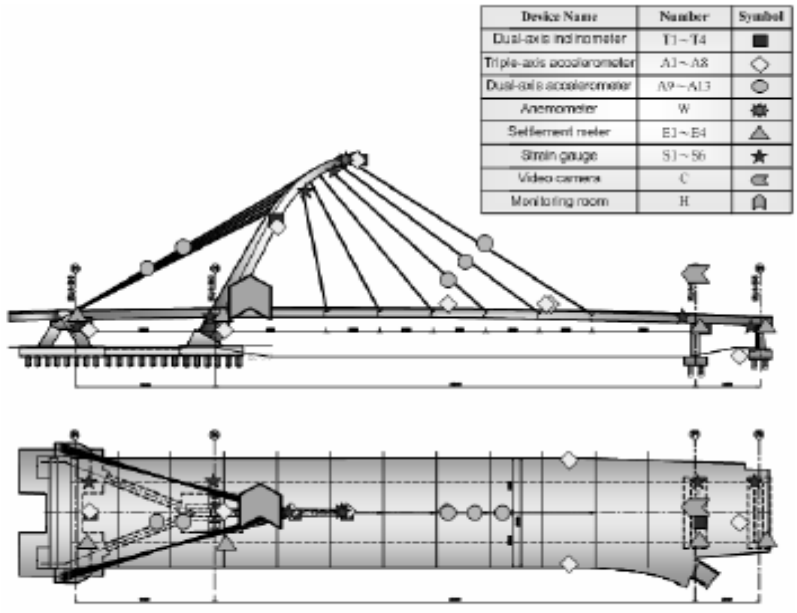


Figure 3. The layout diagram of sensors

Εικόνα 14-WSN για την παρακολούθηση καταστροφικών ζημιών σε γέφυρες

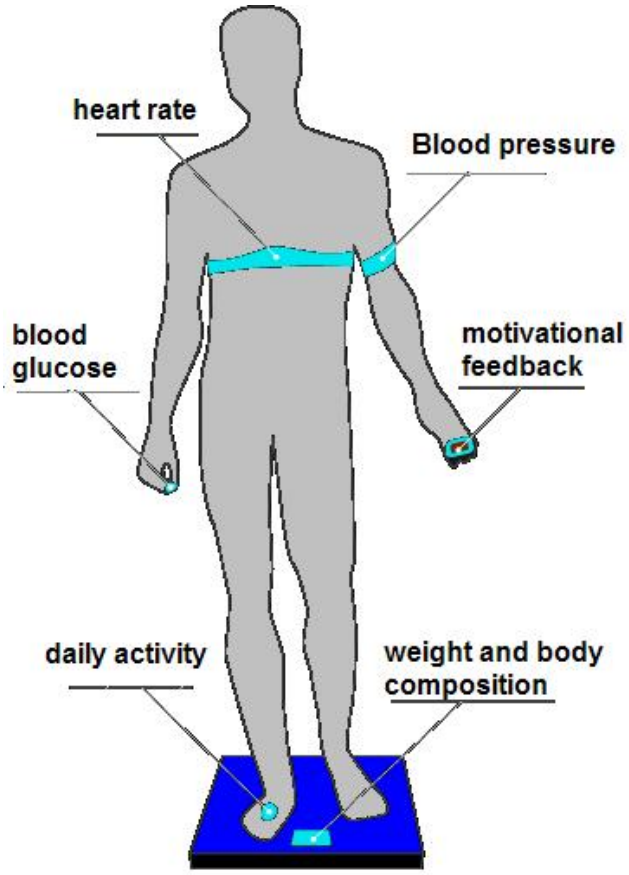
Εφαρμογές στον χώρο της υγείας.

- Συστήματα καταγραφής κρίσιμων βιοσημάτων

Τέτοιου είδους βιοσήματα είναι το ηλεκτροκαρδιογράφημα, το ηλεκτρομυογράφημα, το ηλεκτροεγκεφαλογράφημα, ο κορεσμός του οξυγόνου στο αίμα, η θερμοκρασία του ασθενούς, η αναπνοή που αποτελεί και το εργαστηριακό αντικείμενο της παρούσας διπλωματικής εργασίας, η πίεση κ.α. Έχουν αναπτυχθεί διάφορες τέτοιου είδους εφαρμογές σε πειραματικό στάδιο που μπορούν να αναζητηθούν στη βιβλιογραφία. Οι πιο γνωστές είναι το Codeblue από το πανεπιστήμιο του Χάρβαρντ, το ActiS στο πανεπιστήμιο της Αλαμπάμα και το MIThril στο MIT.

- Σε συνεργασία με περιβαλλοντικούς αισθητήρες για επιδημιολογικές μελέτες.

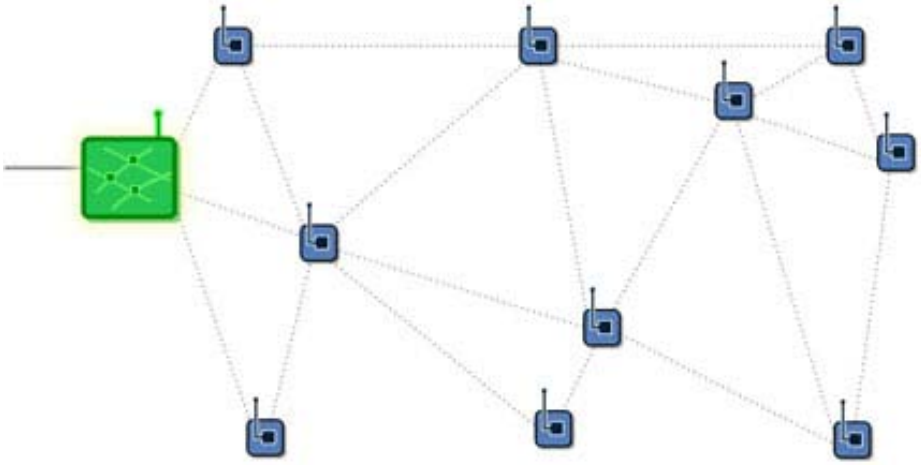
- Χορήγηση φαρμάκων



Εικόνα 15-Εφαρμογές στον χώρο της υγείας.

Αρχές σχεδιασμού ασύρματων δικτύων αισθητήρων

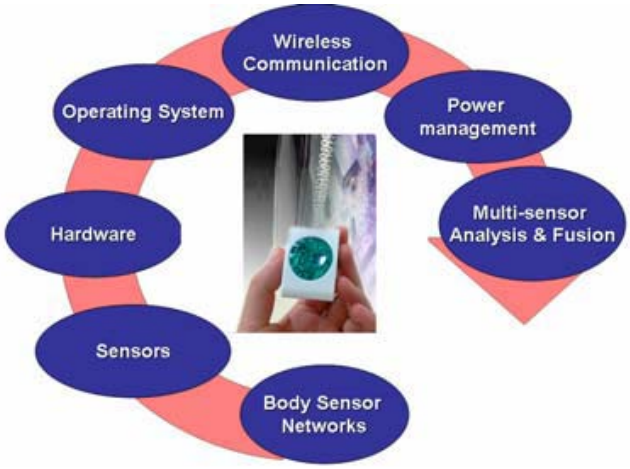
Όπως και με κάθε εφαρμογή έτσι και στα WSN οι ιδιαιτερότητες της κάθε εφαρμογής καθορίζουν τις παραμέτρους σχεδιασμού του δικτύου. Οι παράμετροι αυτοί είναι : η τοπολογία του δικτύου, η αξιοπιστία του, η κάλυψη και επεκτασιμότητα ποιότητα υπηρεσίας, ο συγχρονισμός και χρόνος απόκρισης, η ενεργειακή αποδοτικότητα, η ασφάλεια, το κόστος παραγωγής και ευκολία υλοποίησης.



Εικόνα 16-ασύρματο δίκτυο αισθητήρων με multi-hop λειτουργίες όπου τα δεδομένα καταλήγουν σε ένα τελικό αποδέκτη

Προκλήσεις μελλοντικής εξέλιξης

Η εξελισσόμενη νανοτεχνολογία υπόσχετε μια νέα εποχή στο σχεδιασμό και κατασκευή αισθητήρων αξιόπιστων και σε μεγέθη της τάξης μερικών νανομέτρων που θα δώσει μια νέα ώθηση στα WSN. Η βιοσυμβατότητα είναι ένα άλλο κεφάλαιο προς μελέτη αφού πολλοί αισθητήρες εμφυτεύονται στον ανθρώπινο σώμα σε συνάρτηση με ένα άλλο βασικό κεφάλαιο των ασύρματων αισθητήρων την ενεργειακή κατανάλωση και το χρόνο ζωής. Η αξιοπιστία και η ασφάλεια των δεδομένων που χαρακτηρίζονται προσωπικά και ανήκουν μόνο στο απόρρητο γιατρού ασθενή. Άλλη μια παράμερος που αποτελεί πρόκληση προς εξέλιξη είναι η γνώση του εκάστοτε περιβάλλοντος και της κατάστασης του ασθενούς.



Εικόνα 17-Ένας κόμβος WSN

Κεφάλαιο 2

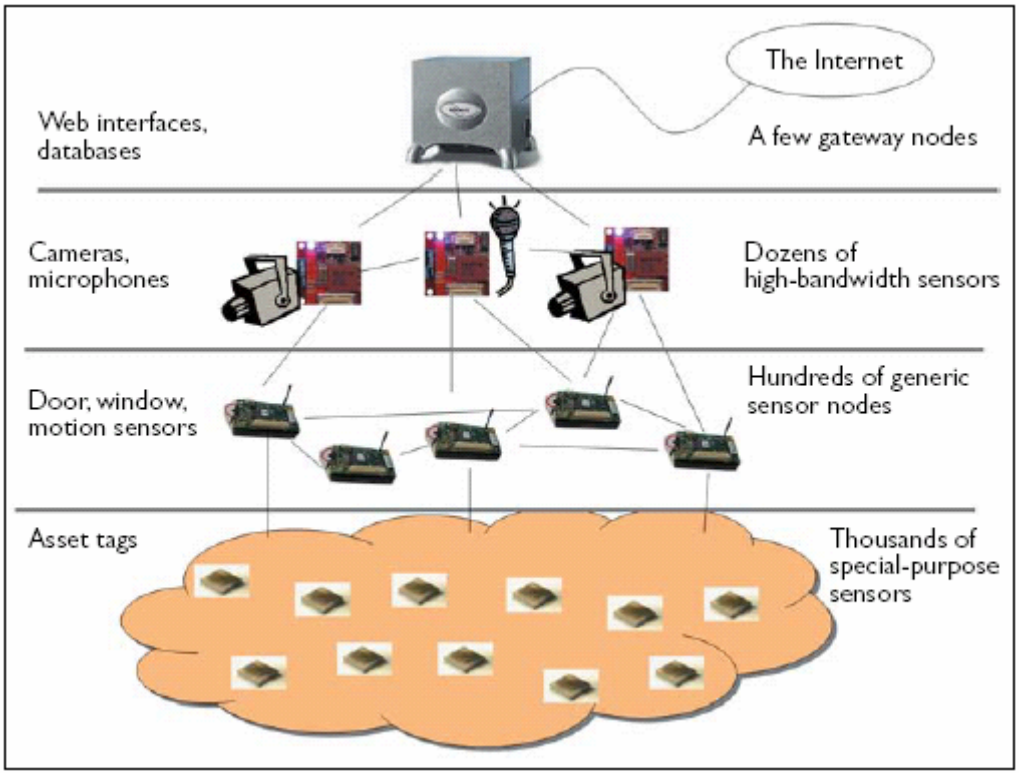
Το υλικό (Hardware) και οι πλατφόρμες των WSN

Όπως αναφέραμε στο προηγούμενο κεφάλαιο, τα ασύρματα δίκτυα αισθητήρων συνδυάζουν δυνατότητες επεξεργασίας, αίσθησης (sensing) και επικοινωνίας, σε μικροσκοπικά ενσωματωμένες συσκευές. Στη συνέχεια τα πρωτόκολλα επικοινωνίας συνδυάζουν κατάλληλα τις ανεξάρτητες συσκευές, για τη δημιουργία ενός διασυνδεδεμένου βροχωτού δικτύου (mesh network), όπου τα δεδομένα δρομολογούνται ανάμεσα σε όλους τους κόμβους.

Στο κεφάλαιο αυτό παρουσιάζονται μερικές τυπικές πλατφόρμες δικτύων αισθητήρων, όπου οι συσκευές έχουν διαστάσεις από μερικά χιλιοστά μέχρι το μέγεθος ενός υπολογιστή παλάμης. Παράλληλα αναλύεται η πλατφόρμα Tmote Sky της εταιρείας Moteiv που χρησιμοποιήσαμε και στην εφαρμογή μας. Σημαντική για τη λειτουργία οποιασδήποτε συσκευής δικτύου αισθητήρων είναι η δυνατότητα να ικανοποιεί αδιάλειπτα τις μεγάλες απαιτήσεις κάθε εφαρμογής. Αντίθετα με τα κινητά τηλέφωνα και τους ασύρματους φορητούς υπολογιστές, η περιοδική τροφοδοσία δεν είναι δυνατή για τα περισσότερα ασύρματα δίκτυα αισθητήρων. Στον τομέα των δικτύων αισθητήρων, μονάδες αισθητήρων (sensor nodes) ειδικού σκοπού, σχεδιάζονται με τέτοιο τρόπο ώστε να θυσιάζουν την ευελιξία προκειμένου να είναι όσο το δυνατόν μικρότερες και σχετικά φτηνές. Γενικευμένες μονάδες αισθητήρων παρέχουν διεπαφές (interfaces) με μεγάλες δυνατότητες επέκτασης, ώστε να δημιουργούν ευέλικτες συνδέσεις με μια σειρά από απλούς αισθητήρες. Μονάδες αισθητήρων μεγάλου εύρους ζώνης έχουν ενσωματωμένες τις δυνατότητες επεξεργασίας και επικοινωνίας, που είναι απαραίτητες ώστε να ανταποκρίνονται σε πολύπλοκες ακολουθίες δεδομένων, συμπεριλαμβανομένης της επεξεργασίας κινούμενης εικόνας (video) και ήχου. Μονάδες που λειτουργούν ως πύλες (Gateway nodes) παρέχουν μια σημαντική σύνδεση μεταξύ του δικτύου αισθητήρων και των παραδοσιακών υποδομών διαδικτύωσης, συμπεριλαμβανομένων του Ethernet, του 802.11 προτύπου επικοινωνίας και των διευρυμένων δικτύων.

Πλατφόρμες στις οποίες βασίζονται τα δίκτυα αισθητήρων

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



Εικόνα 18-Ιεραρχική ανάπτυξη ενός ασύρματου δικτύου αισθητήρων

Αυτή η βαθμωτή αρχιτεκτονική είναι κοινή σε όλα σχεδόν τα δίκτυα αισθητήρων και γίνεται εύκολα κατανοητή με ένα παράδειγμα. Ας θεωρήσουμε ένα δίκτυο αισθητήρων ενός προηγμένου συστήματος ασφαλείας, στο οποίο η πλειονότητα των αισθητήρων καλύπτει σπάσιμο τζαμιών, κλείσιμο επαφών και ανίχνευση κίνησης. Το πλήθος των αισθητήρων και των κατάλληλων θέσεων τους απαιτούν να τροφοδοτούνται από μπαταρία. Συμπληρώνονται από μερικούς περισσότερο εξελιγμένους αισθητήρες, όπως είναι οι κάμερες ,οι ανιχνευτές ήχων και χημικών, τοποθετημένοι σε καίρια σημεία. Τα απλά και τα σύνθετα δεδομένα των αισθητήρων δρομολογούνται μαζί, μέσω ενός δικτύου, σε μια μονάδα παρακολούθησης και ελέγχου του κτιρίου, που παρέχει τη δυνατότητα συνεχούς παρακολούθησης. Οι αισθητήρες που είναι τοποθετημένοι σε παράθυρα και πόρτες

για ανίχνευση εισβολής είναι παραδείγματα γενικευμένων μονάδων αισθητήρων (generic sensing devices*)*. Η λειτουργία τους είναι απλή και συγκεκριμένη και απαιτεί την τροφοδοσία από μπαταρία μεγάλης διάρκειας. Επιπλέον, οι ρυθμοί επεξεργασίας και επικοινωνίας που διαθέτουν, είναι οι ελάχιστοι. Αντίθετα, οι αισθητήρες ήχου, εικόνας και χημικών είναι παραδείγματα *μονάδων μεγάλου εύρους ζώνης*, που απαιτούν επικοινωνία και μεγαλύτερη υπολογιστική ισχύ. Μπορεί σε κάποιες περιπτώσεις να απαιτούν τροφοδότηση από μπαταρία αλλά συχνά χρειάζεται να συνδεθούν με το δίκτυο παροχής ηλεκτρικής τάσης, για να λειτουργήσουν σε μακρά διάρκεια.

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

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

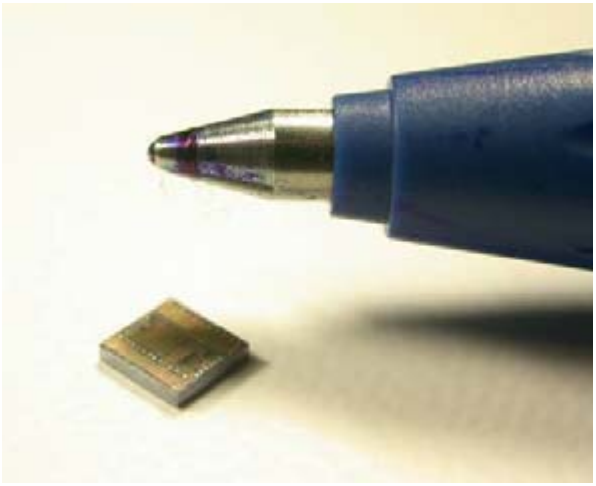
Στον Πίνακα παρατίθενται τα τυπικά χαρακτηριστικά λειτουργίας των τεσσάρων κατηγοριών των μονάδων-κόμβων: πλατφόρμα-αισθητήρας ειδικού σκοπού (specialized sensing platform), πλατφόρμα-αισθητήρας γενικού σκοπού (generic sensing platform), πλατφόρμα-αισθητήρας μεγάλου εύρους ζώνης (high bandwidth sensing)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

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

Εικόνα 19-Τυπικά χαρακτηριστικά λειτουργίας των 4 κατηγοριών ασύρματου δικτύου.

Η μονάδα Spec

Η μονάδα **Spec** είναι ενδεικτική της τάξης αισθητήρων ειδικού σκοπού. Είναι μια μονάδα μονού στοιχείου (single-chip node), σχεδιασμένη ιδιαιτέρως για παραγωγή εξαιρετικά χαμηλού κόστους και λειτουργία χαμηλής ισχύος. Απαιτώντας μόνο 2.5mm*2.5mm πυριτίου, περιλαμβάνει μνήμη RAM και ικανότητες επεξεργασίας και επικοινωνίας. Προκειμένου να μειωθεί το μέγεθος και η πολυπλοκότητα, η μονάδα Spec κατασκευάστηκε έτσι ώστε να έχει διεπαφή μόνο με απλούς αισθητήρες και να επικοινωνεί σε μικρές αποστάσεις. Οι πρώτες εκδοχές της περιλάμβαναν μόνο πομπό, ενώ οι επόμενες έχουν πλήρη πομποδέκτη. Η μονάδα Spec είναι ιδανική για εφαρμογές παρακολούθησης 'κινητών αντικειμένων αξίας'. Εξοπλισμένη με μικρή μπαταρία είναι ικανή να λειτουργεί για πολλά χρόνια.

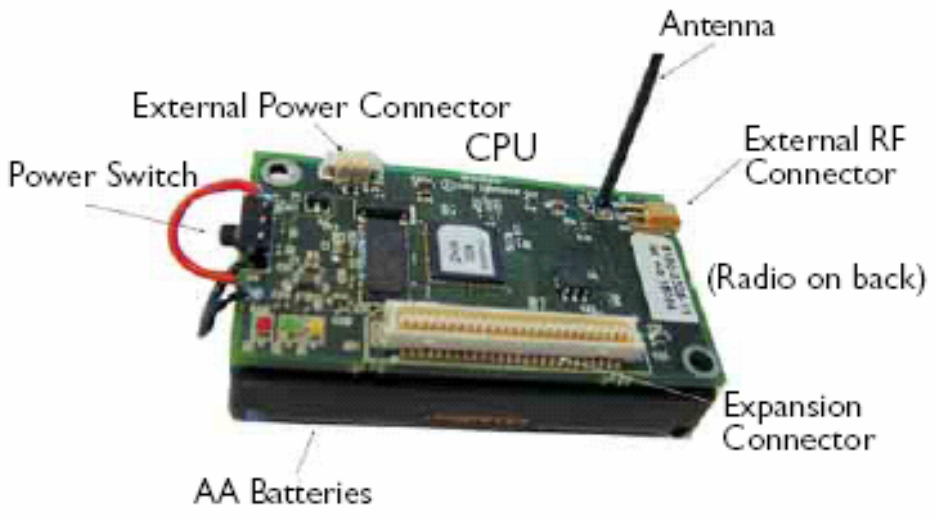


Εικόνα 20-Η μονάδα Spec

Τα motes του Πανεπιστημίου **Berkeley, California** αποτελούν παράδειγμα συσκευών γενικευμένης τάξης (generic sensor devices), που χρησιμοποιούνται σήμερα από περισσότερους από εκατό ερευνητικούς οργανισμούς. Κάποια από αυτά είναι το Mica2 και το Tmote Sky.

Το MicaZ ,Mica2

Το **Mica2** είναι ένα από τα πιο πρόσφατα ανεπτυγμένα εμπορικά διαθέσιμα μοντέλα, που ενσωματώνει εξαρτήματα για μέγιστη ευελιξία, με το **Micaz** να αποτελεί την πιο σύγχρονη εξέλιξη του.

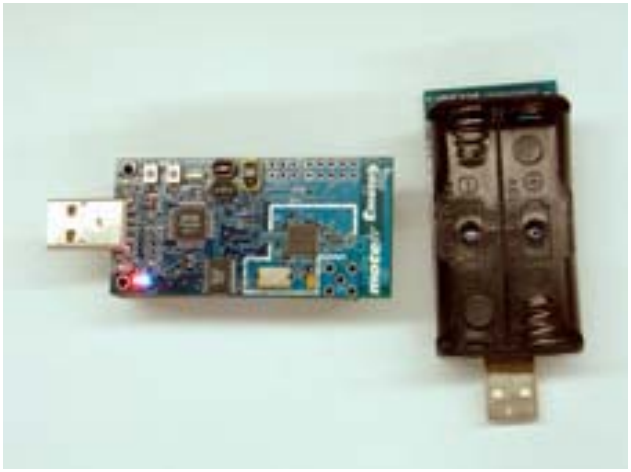


Εικόνα 21-Το Mica2

Περιλαμβάνει ένα μεγάλο σύνδεσμο διεπαφής παρέχοντας τη δυνατότητα προσάρτησης μιας σειράς από αισθητήρες. Διαθέτοντας μεγάλο πλήθος από I/O pins και δυνατότητες επέκτασης, το Mica2 είναι μια από τις καλύτερες επιλογές κόμβων-αισθητήρων σε περιπτώσεις όπου το μέγεθος και το κόστος δεν είναι σημαντικοί παράγοντες. Για παράδειγμα, συνδέεται εύκολα σε ανιχνευτές κίνησης και σε επαφές παραθύρων και θυρών, που είναι απαραίτητα για το σύστημα ασφάλειας σε κτίρια. Επιπλέον, το Mica2 είναι ικανό να δέχεται μηνύματα από μονάδες-κόμβους Spec, που είναι τοποθετημένοι σε αντικείμενα αξίας, όπως οι προσωπικοί και φορητοί υπολογιστές, για περιπτώσεις κλοπής. Η μνήμη και η επεξεργαστική ισχύς που είναι διαθέσιμη στο Mica2, είναι ικανές για τη διαχείριση πολλών δεδομένων που στέλνονται από τις μονάδες Spec. Παρόλο που το Mica2 μπορεί να συνδεθεί με ένα μεγάλο πλήθος αισθητήρων, δεν μπορεί να ανταποκριθεί στο μεγάλο εύρος δεδομένων που προέρχονται από σύνθετους αισθητήρες. Αποτυγχάνει στην επεξεργασία κινούμενης εικόνας και ήχου μεγάλου εύρους ζώνης.

Tmote sky

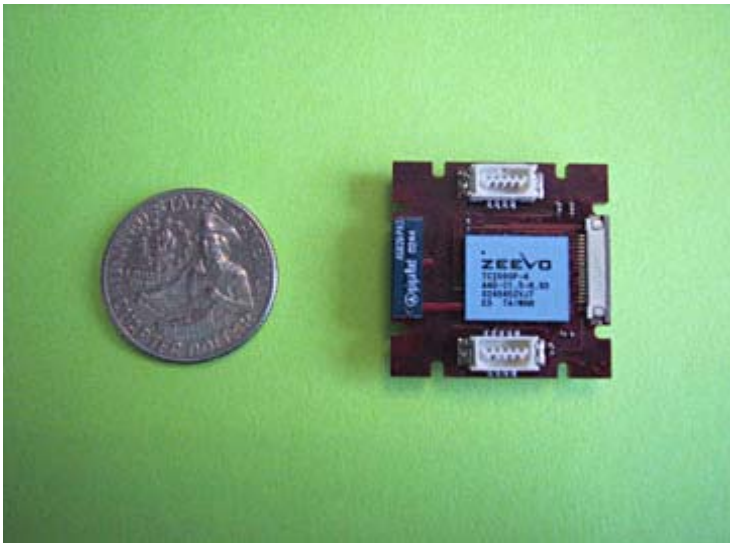
Το **tmote-sky** (το προηγούμενο μοντέλο ονομαζόταν Telosb) αποτελεί επίσης μια μονάδα που συνδυάζει ενσωματωμένους αισθητήρες, δυνατότητες ασύρματης επικοινωνίας και προγραμματιστικές δυνατότητες. Τα χαρακτηριστικά του θα περιγράφουν αναλυτικά στη συνέχεια.



Εικόνα 22-Tmote sky

Το imote

Το **iMote**, που δημιούργησε η Intel Research τον Μάιο του 2003, έχει σχεδιαστεί ως πλατφόρμα αισθητήρων μεγάλου εύρους ζώνης και περιλαμβάνει πολύ μεγαλύτερη μνήμη RAM και ισχύ επεξεργασίας, όπως επίσης πομποδέκτη βασισμένο σε τεχνολογία Bluetooth, ικανό να επικοινωνεί σε ταχύτητες μεγαλύτερες από 500Kbps.



Εικόνα 23- Η πλατφόρμα της intel iMote

Η πλατφόρμα Stargate

Η πλατφόρμα **Stargate**, που ανέπτυξε η Intel και πούλησε η Crossbow Technology, είναι αντιπροσωπευτική των συσκευών κατηγορίας πύλης (gatewayclass devices) και περιλαμβάνει επεξεργαστή Intel 400 MHz, μνήμη RAM μερικών megabytes και δυνατότητα αποθήκευσης μέχρι την τάξη των gigabytes. Είναι ικανή να συνδέεται ευθέως με συσκευές βασισμένες στο Mica2 και το iMote και να διαβιβάζει δεδομένα από χαμηλής ισχύος δίκτυα σε παραδοσιακά ασύρματα δίκτυα όπως είναι το 802.11 και το Ethernet. Επιπλέον, οι διατάξεις μνήμης και επεξεργασίας του, του επιτρέπουν να λειτουργεί ως Web front-end σε δίκτυα αισθητήρων, όπου οι χρήστες έχουν πρόσβαση στα δεδομένα του μέσω Web browser.

Εικόνα 24-Η πλατφόρμα stargate της CrossBow

Το λειτουργικό σύστημα που τρέχει σε συγκεκριμένη πλατφόρμα πρέπει να είναι συμβατό με τις δυνατότητες του υλικού (hardware) της πλατφόρμας. Για συσκευές ειδικού και γενικού σκοπού, ένα ειδικό λειτουργικό σύστημα καλούμενο **TinyOS**, το οποίο θα περιγραφεί στο επόμενο κεφάλαιο, έχει σχεδιαστεί ώστε να τρέχει σε πλατφόρμες με περιορισμένη υπολογιστική ισχύ και μνήμη. Αντίθετα με πολλά ενσωματωμένα λειτουργικά συστήματα, αυτό παρέχει ισχυρή ενοποίηση ανάμεσα σε ασύρματη σύνδεση και λειτουργίες δικτύου. Παρόλα αυτά, καθώς αυξάνουν οι δυνατότητες των πλατφορμών, όπως για παράδειγμα συμβαίνει στην πλατφόρμα Stargate, απαιτείται όλο και περισσότερη συμμετοχή από το λειτουργικό σύστημα ώστε να υποστηριχτούν πιο σύνθετες εφαρμογές. Πολυεπεξεργασία (multiprocessing), μεταγωγή εκτέλεσης διεργασιών με βάση την προτεραιότητα (preemptive task switching) ή ακόμα υποστήριξη εικονικής μνήμης, είναι επιθυμητά στη διεκπεραίωση πολλαπλών λειτουργιών του συστήματος. Η μονάδα Stargate τρέχει μια ενσωματωμένη εκδοχή του λειτουργικού συστήματος Linux. Όχι μόνο προσφέρει ένα πλήθος δυνατοτήτων του συστήματος αλλά, επιπλέον, το Linux παρέχει μια πληθώρα οδηγών συσκευής (device drivers) για κάρτες Ethernet και κάρτες ασύρματης δικτύωσης 802.11 που είναι απαραίτητες για να επιτρέψουν στους κόμβους-πύλες να συνδεθούν σε ένα ευρύ φάσμα συστημάτων δικτύωσης.

BTnode

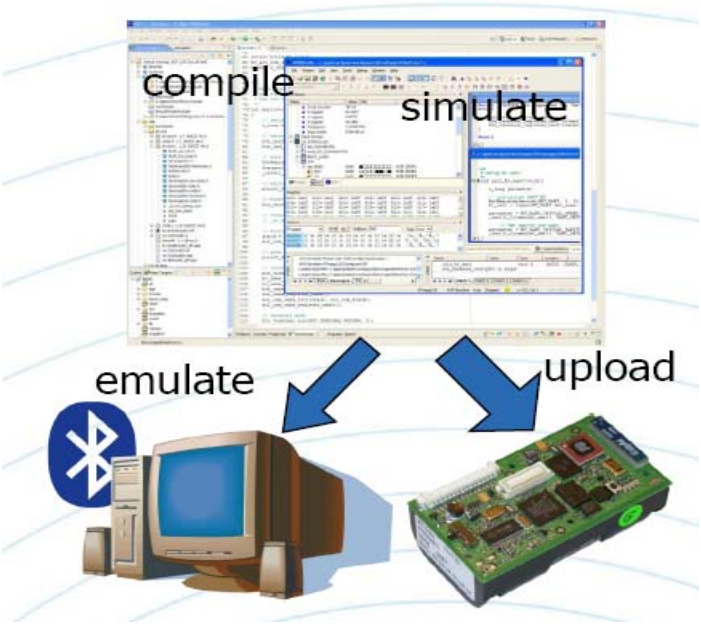
Το BTnode είναι μια αυτόνομη ασύρματη πλατφόρμα επικοινωνίας και υπολογισμών βασισμένη σε ένα ραδιοπομπό Bluetooth και έναν μικροελεγκτή. Χρησιμεύει ως μια πλατφόρμα επίδειξης για την έρευνα σε κινητά και ειδικά συνδεδεμένα δίκτυα (MANETs) και διανεμημένα δίκτυα αισθητήρων. Το BTnode έχει αναπτυχθεί από κοινού στο ETH Ζυρίχηςαπό την εφαρμοσμένη μηχανική υπολογιστών και το εργαστήριο δικτύων (TIK) και την ερευνητική ομάδα για τα διανεμημένα συστήματα. Το χαμηλής ισχύος ασύρματο σύστημα εκπομπής είναι το ίδιο όπως χρησιμοποιείται και στα Berkley motes Mica2. Και τα δύο συστήματα εκπομπής μπορούν να χρησιμοποιηθούν ταυτόχρονα ή να κλείνουν ανεξάρτητα όταν δεν βρίσκονται σε χρήση, μειώνοντας αρκετά τη κατανάλωση ισχύος της συσκευής.

Εικόνα 25-BT Node

Τα χαρακτηριστικά του BT node

- Microcontroller: Atmel ATmega 128L (8 MHz @ 8 MIPS)
- Memories: 64+180 Kbyte RAM, 128 Kbyte FLASH ROM, 4 Kbyte EEPROM
- Bluetooth subsystem: Zeevo ZV4002, supporting AFH/SFH
- Scatternets with max. 4 Piconets/7 Slaves, BT v1.2 compatible
- Low-power radio: Chipcon CC1000 operating in ISM band 433-915 MHz
- External Interfaces: ISP, UART, SPI, I2C, GPIO, ADC, Timer, 4 LEDs
- Standard C Programming, **TinyOS** compatible

Η πλατφόρμα ξεφεύγει από τη βασικότερη φιλοσοφία των WSN που αφορά την χαμηλότερη δυνατή κατανάλωση ενέργειας από του κόμβους και στοχεύει στην προσαρμοστικότητα ,την εύκαμπτη και γρήγορη εφαρμογή. Για τον λόγω αυτό εκτός από την συμβατότητα της με το **TinyOS** χρησομοποιεί και το BTnut το οποίο είναι ένα πολύ ελαφρύ λειτουργικό σύστημα που χρησιμοποιεί την απλή γλώσσα C. Έχει γραφικό περιβάλλον και βιβλιοθήκες όπως και απλές εφαρμογές.



Αρχιτεκτονικές διαφορές

Η συνολική αρχιτεκτονική δομή και στις 4 κατηγορίες πλατφορμών δικτύων αισθητήρων είναι αξιοσημείωτα όμοια, παρά τις σημαντικές διαφορές στις δυνατότητες των συσκευών. Η αρχιτεκτονική ομοιότητα προκύπτει από την απαίτηση να υποστηρίζουν την ασύρματη δικτύωση. Αντίθετα, οι βασικές τους διαφορές προκύπτουν από την επιθυμία των σχεδιαστών τους να βελτιστοποιήσουν την κατανάλωση ενέργειας καθεμιάς πλατφόρμας για συγκεκριμένη κατηγορία εφαρμογής. Κάποιες από τις θεμελιώδεις αποφάσεις που πρέπει να λάβουν οι μηχανικοί εφαρμογών περιλαμβάνουν το μέγεθος της on-board μνήμης, εάν θα συμπεριλάβουν μνήμη αναλαμπής (flash memory), το μέγεθος της ισχύος της κεντρικής μονάδας επεξεργασίας (CPU) καθώς, επίσης, τον τύπο και το εύρος

ζώνης της ασύρματης ζεύξης. Αφού οι περισσότερες υλοποιήσεις μεταχειρίζονται εξεζητημένα συστατικά στοιχεία, κάποιες από αυτές τις αποφάσεις υπαγορεύονται από τη διαθεσιμότητα των κατάλληλων μερών. Στο τέλος, το κόστος και η κατανάλωση ενέργειας είναι οι κύριοι παράγοντες που επηρεάζουν τον τελικό σχεδιασμό της κάθε μονάδας- αισθητήρα. Μια κύρια διαφορά ανάμεσα σε μονάδες δικτύου αισθητήρων και πιο παραδοσιακών υπολογιστικών πλατφορμών, περιλαμβανομένων των προσωπικών υπολογιστών, των υπολογιστών παλάμης (PDAs), ακόμα και των ενσωματωμένων συσκευών είναι η ακραία έμφαση που δίνουν τα δίκτυα αισθητήρων στη διαχείριση της ενέργειας. Μια πληθώρα εφαρμογών απαιτούν τροφοδότηση με μπαταρία για μεγάλα χρονικά διαστήματα. Προκειμένου να διαχειρίζεται αποτελεσματικά η ισχύς, κάθε υποσύστημα της πλατφόρμας τροφοδοτείται ανεξάρτητα. Για παράδειγμα, ο πομποδέκτης πρέπει να λειτουργεί μόνο κατά τη διάρκεια της ενεργής επικοινωνίας και, αν είναι δυνατόν, να κλείνει την κεντρική μονάδα επεξεργασίας στις περιόδους μη επεξεργασίας. Όμοια, πρέπει να είναι σε θέση να κόβει την τροφοδοσία στα υποσυστήματα αισθητήρων και μονάδων εισόδου-εξόδου, ξεχωριστά, όταν είναι ανενεργά.

Το λειτουργικό σύστημα **TinyOS**, σε πολλές περιπτώσεις, ελέγχει την δραστηριότητα και την ισχύ των διαφόρων υποσυστημάτων. Ο χρονισμός των περιόδων που σταματάει η τροφοδοσία (power-down cycles) καθορίζεται από ένα μεγάλο αριθμό παραγόντων, όπως είναι οι απαιτήσεις της εφαρμογής και το συγκεκριμένο υλικό που χρησιμοποιείται. Στο **TinyOS**, η διαχείριση ισχύος αφορά κάθε τομέα του συστήματος και όλα τα επιμέρους στοιχεία είναι σχεδιασμένα ώστε να μην καταναλώνουν ισχύ όταν είναι ανενεργά. Για να διευκολυνθεί η σωστή διαχείριση ισχύος, οι πλατφόρμες των δικτύων αισθητήρων δίνουν απευθείας στις εφαρμογές, λεπτομερή έλεγχο του υποκείμενου υλικού. Παραδοσιακές αντιλήψεις διαστρωμάτωσης για τις στοίβες τόσο του δικτύου όσο και των αισθητήρων οδηγούν σε αναποτελεσματική χρήση της ισχύος. Πρόσφατη έρευνα προτείνει μια κοινή προσέγγιση αυτής της πρόκλησης στο πεδίο των πλατφορμών, με τη χρήση 3 πρόσθετων αρχιτεκτονικών στοιχείων:

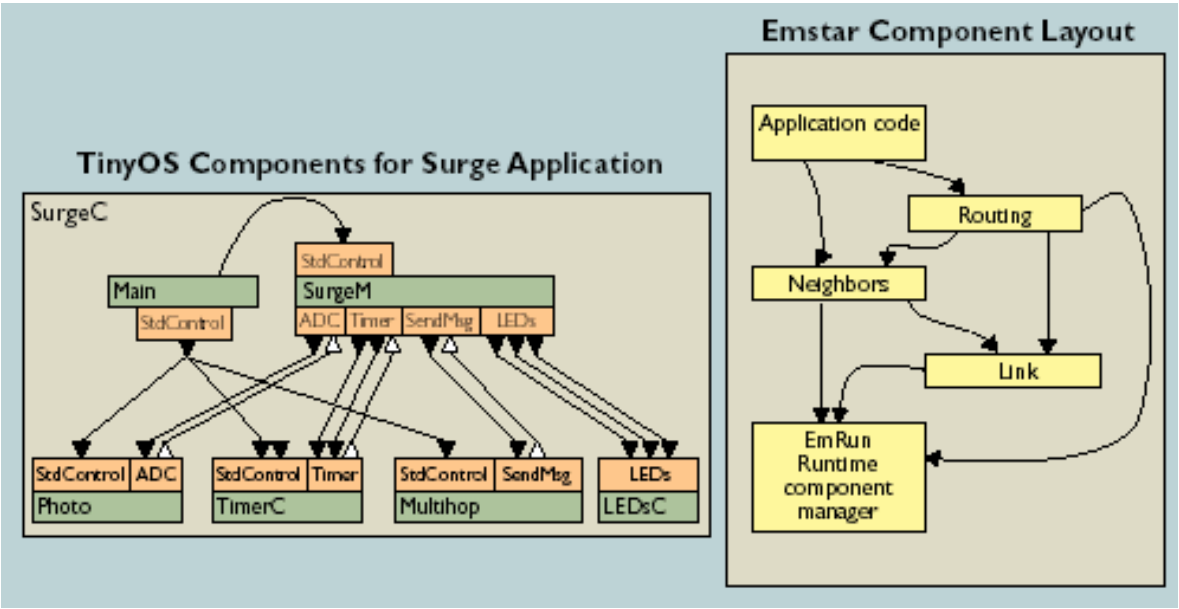
• Ένα πλαίσιο στοιχείων γενικού σκοπού που καταργεί τη διαστρωμάτωση

• Λειτουργίες υλικού που είναι διαθέσιμες σε εφαρμογές και σε εξατομικευμένο λογισμικό (middleware)

• Εικονικοποίηση (virtualization), μεταφρασμένα προγράμματα ή απλοποιημένη διαδικασία προγραμματισμού για την ανάπτυξη εφαρμογών δικτύων αισθητήρων

Στις συσκευές κατηγορίας mote (mote-class devices), όπως είναι το Spec και το Mica2, το **TinyOS** παρέχει ένα χαμηλού επιπέδου έλεγχο υλικού, μέσω ενός ενσωματωμένου στοιχείου που απαλείφει τη διαστρωμάτωση. Στο **TinyOS**, επιτρέπεται στα στοιχεία επιπέδου εφαρμογής να έχουν απευθείας πρόσβαση στο υλικό, όπως απαιτείται. Ενώ αυτή η δυνατότητα εμφανίζεται και σε άλλα ενσωματωμένα λειτουργικά συστήματα, γενικώς απουσιάζει από άλλα πιο παραδοσιακά λειτουργικά συστήματα, συμπεριλαμβανομένου και του Linux.

Όταν το Linux χρησιμοποιείται σε μονάδες κατηγορίας πύλης (gateway-class nodes), όπως είναι το Stargate, χρειάζεται επιπρόσθετη υποστήριξη για ακριβή έλεγχο του υλικού, για την οποία έχουν μεριμνήσει οι σχεδιαστές. Στο Stargate, οι καταχωρητές (processor registers) και οι γραμμές εισόδου-εξόδου γενικού σκοπού, γίνονται διαθέσιμες στις εφαρμογές μέσω οδηγών (drivers) ειδικού σκοπού. Στη συνέχεια, τα περιβάλλοντα ανάπτυξης των δικτύων αισθητήρων (όπως είναι το Emstar), χρησιμοποιούν αυτούς τους οδηγούς για να παρέχουν στις εφαρμογές, τον έλεγχο πάνω στον χρονισμό και στην κατάσταση των περιφερειακών (hardware peripherals) που χρειάζονται .



Εικόνα 26-Μοντέλο της εφαρμογής Emstar

Οι προσπάθειες ανάπτυξης του TinyOS και του ενσωματωμένου Linux, υιοθέτησαν το virtualization (εικονικοποίηση) της επεξεργασίας και των πόρων επικοινωνίας, για να απλοποιήσουν τη διαδικασία εξέλιξης των δικτύων αισθητήρων. Ένα τίμημα για την παροχή ακριβούς ελέγχου υλικού σε λογισμικό επιπέδου εφαρμογών είναι ότι κάποιες φορές, συγκεκριμένα δομικά στοιχεία του υλικού καθιστούν το λογισμικό του δικτύου αισθητήρων, μη συμβατό. Τόσο το **TinyOS** όσο και το **Emstar** παρουσιάζουν κάποιες αφηρημένες έννοιες για το υλικό που προσπαθούν να διατηρήσουν τη συμβατότητα, χωρίς να θυσιάζουν τον ακριβή έλεγχο. Το καθένα παρέχει την επιλογή της χρήσης υψηλού επιπέδου μεταφραστών για τη διευκόλυνση ανάπτυξης της εφαρμογής.

Η εξέλιξη στο υλικό και το λογισμικό των πλατφορμών

Η πρόσφατη έρευνα και ανάπτυξη των πλατφορμών 1ης γενιάς ασύρματων δικτύων αισθητήρων επαναπροσδιορίζεται για να βοηθήσει τους μηχανικούς συστημάτων να ορίσουν μια νέα γενιά υλικού που θα εξυπηρετεί καλύτερα τις ανάγκες των δικτύων.

Αναλύοντας την εξέλιξη στο υλικό των δικτύων αισθητήρων πρέπει να τονίσουμε την επίδραση του νόμου του Moore, στο σχεδιασμό και την εξέλιξη των δικτύων. Για όλες τις κατηγορίες πλατφορμών, εκτός από τις μονάδες αισθητήρων ειδικού σκοπού, ο νόμος του Moore εγγυάται αύξηση της απόδοσης για δεδομένη ισχύ. Όπως φαίνεται στον Πίνακα η μονάδα Mica2 έχει σχεδόν οχταπλάσια μνήμη και εύρος ζώνης επικοινωνίας από τον προκάτοχό του, τη μονάδα **Rene**, σχεδιασμένη το 1999, παρότι έχουν ίδια ισχύ και κόστος. Οι συσκευές κατηγορίας πύλης (gateway devices) και μεγάλου εύρους ζώνης (high-bandwidth devices) έχουν επιτύχει παρόμοια άλματα απόδοσης, χωρίς σημαντική αλλαγή στις απαιτήσεις ισχύος και κόστους. Αντίθετα, οι μονάδες αισθητήρων ειδικού σκοπού, όπως είναι η μονάδα Spec, χρησιμοποιούν προχωρημένες τεχνικές που απορρέουν από το νόμο του Moore, για να μειώσουν την κατανάλωση ισχύος και το κόστος, ενώ διατηρούν την ίδια απόδοση.

Μέρος της αυξημένης απόδοσης των μονάδων αισθητήρων γενικευμένης τάξης οφείλεται στους νέους CMOS ραδιοπομπούς, που έχουν σχεδιαστεί για εκπομπή χαμηλού ρυθμού και χαμηλή κατανάλωση ισχύος. Επιπλέον της αύξησης της απόδοσης των πομπών, οι διεπαφές επικοινωνίας που παρέχονται από πομπούς χαμηλής ισχύος, περιλαμβάνουν τώρα εξειδικευμένη υποστήριξη υλικού

για να βοηθήσουν στη μείωση του υψηλού φόρτου της κεντρικής μονάδας επεξεργασίας. Ελεγκτές χαμηλής ισχύος μπορούν να στείλουν δεδομένα μέσω RF καναλιού, με πολλαπλάσιες ταχύτητες των πομπών της προηγούμενης γενιάς. Επιπρόσθετα, προηγούμενοι σχεδιασμοί υλικού χρησιμοποιούσαν τον μικροελεγκτή για να καθορίζει τον κύκλο λειτουργίας του πομπού και να ελέγχει για δραστηριότητα στο κανάλι. Οι επόμενης γενιάς πομποί έχουν ενσωματωμένους μηχανισμούς που εκτελούν αυτόματα αυτή τη λειτουργία.

Σημερινές πλατφόρμες δικτύων αισθητήρων οργανωμένες κατά τάξη συσκευής

Node	CPU	Power	Memory	I/O and Sens	Radio	Remarks
Special -purpose Sensor Notes						
Spec 2003	4-8Mhz Custom 8-bit	3mW peak 3uW idle	3K RAM	I/O Pads on chip, ADC	50-100Kbps	Full custom silicon, traded RF range and accuracy for low-power
Generic Sensor Notes						
Rene 1999	ATMEL 8535	.036mW sleep 60mW active	512B RAM 8K Flash	Large expansion connector	10Kbps	Primary TinyOS development platform.
Mica-2 2001	ATMEGA 128	.036mW sleep 60mW active	4K RAM 128K Flash	Large expansion connector	76Kbps	Primary TinyOS development platform.
Telos 2004 (Tmote Sky)	Motorola HCS08	.001 mW sleep 32mW active	4K RAM	USB and Ethernet	250Kbps	Supports IEEE 802.15.4 standard. Allows higher-layer Zigbee stardard. 1.8V operation
Mica-Z 2004	ATMEGA 128		4K RAM 128K Flash	Large expansion connector	250Kbps	Supports IEEE 802.15.4 standard. Allows higher-layer Zigbee stardard.
High Bandwidth Sensor Notes						
BT Node	ATMEL Mega 128L 7.328Mhz	50MW idle 285MW active	128KB Flash 4KB EEPROM 4KB SRAM	8-channel 10-bit A/D, 2 UARTS Expandable connectors	Bluetooth	Easy connectivity with cell phones. Supports TinyOS. Multihop using multiple radios/nodes.
Imote	ARM	1mW idle	64KB SRAM	UART, USB,	Bluetooth	Multihop using

2003	7TDMI 12-48MHz	120mW active	512KB Flash	GPIO, 12C,SPI .	1.1	scatternets, easy connections to PDAs, phones,TinyOS 1.0, 1.1
Gateway Nodes						
Stargate 2003	Intel PXA255		64KNSRM	2 PCMICA/CF, com ports, Ethernet, USB	Serial connection to sensor network	Flexible I/O and small form factor power management.
Inrysnc Cerfcube 2003	Intel PXA255		32KB Flash 64KB SRAM	Single CF card, general-purpose I/O		Small form factor, robust industrial support, Linux and Windows CE support.
PC 104 nodes	X86 processor		32KB Flash 64KB SRAM	PCI Bus		Embedded Linux or Windows support.

Πρότυπα λογισμικού και διεπαφών

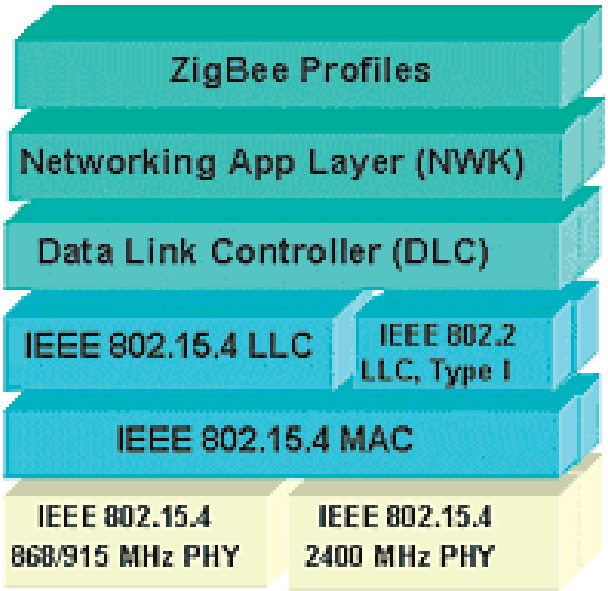
Μηχανικοί και ερευνητές που δραστηριοποιούνται στο χώρο της ασύρματης τεχνολογίας χαμηλής ισχύος χρησιμοποιούν όλο και περισσότερο το πρότυπο 802.15.4. Το πρότυπο αυτό παρέχει μια προδιαγραφή του καναλιού RF και του πρωτοκόλλου σηματοδοσίας. Το πρωτόκολλο **Zigbee**, που βασίζεται πάνω στο 802.15.4, είναι μια προδιαγραφή του πρωτοκόλλου επικοινωνίας συσκευών σε επίπεδο εφαρμογής και θα περιγράφει αναλυτικά στη συνέχεια. Για να εισάγουμε το **Zigbee** και το 802.15.4 στη λογική των πλατφορμών που μελετάμε εδώ, το 802.15.4 αποφασίζει ποιο υλικό ασύρματης επικοινωνίας θα χρησιμοποιηθεί και το **Zigbee** καθορίζει το περιεχόμενο των μηνυμάτων που μεταδίδονται από κάθε μονάδα δικτύου. Ακολουθώντας τη διαθεσιμότητα των πρώτων 802.15.4 πομπών στις αρχές του 2004, οι ερευνητές αποφάσισαν να αναπτύξουν τους **TinyOS** οδηγούς, ώστε οι υπάρχουσες εφαρμογές να μπορέσουν να εκμεταλλευτούν τις δυνατότητες των 802.15.4 στοιχείων (chips).

Μολονότι η διαδικασία προτυποποίησης προοδεύει, δεν είναι σίγουρο αν ένα σύνολο τυποποιημένων πρωτοκόλλων θα είναι κάποτε ικανό να ικανοποιήσει όλες τις απαιτήσεις των εφαρμογών. Αντίθετα με τις παραδοσιακές εφαρμογές Internet, που σχεδόν όλες χρησιμοποιούν πρωτόκολλα **TCP/IP**, οι εφαρμογές του δικτύου

αισθητήρων απαιτούν πρωτόκολλα που είναι βελτιστοποιημένα για τα μοναδικά τους σχήματα επικοινωνίας (communication patterns). Σε αυτό το περιβάλλον, η ικανότητα του **TinyOS** να επιτρέπει σε όσους αναπτύσσουν εφαρμογές να συγκεντρώνουν πρωτόκολλα από ανεξάρτητα δίκτυα, θα συνεχίσει να είναι η προτιμώμενη στρατηγική ανάπτυξης δικτύων αισθητήρων.

Το πρωτόκολλο επικοινωνίας ZigBee

Το πρωτόκολλο **Zigbee** παρέχει ένα ανοικτό πρότυπο ασύρματης δικτύωσης χαμηλής ισχύος, για παρακολούθηση και έλεγχο συσκευών. Χρησιμοποιώντας το πρότυπο IEEE 802.15.4 - που επικεντρώνεται σε δικτύωση χαμηλών ταχυτήτων και ορίζει τα πρωτόκολλα χαμηλών επιπέδων, όπως είναι π.χ. το φυσικό επίπεδο (PHY) και επίπεδο ελέγχου πρόσβασης μέσου (MAC) – το Zigbee ορίζει τα ανώτερα επίπεδα της στοίβας πρωτοκόλλων, από το επίπεδο δικτύου έως της εφαρμογής, περιλαμβάνοντας κατανομές εφαρμογής (application profiles). Μπορούμε να φανταστούμε το 802.15.4 σαν το φυσικό ραδιοστρώμα και το **Zigbee** σαν το λογισμικό λογικού δικτύου και εφαρμογών. Το **Zigbee** χρησιμοποιεί την ISM (Industrial, Scientific and Medical) ζώνη συχνοτήτων, που επιτρέπει απεριόριστη γεωγραφική χρήση.



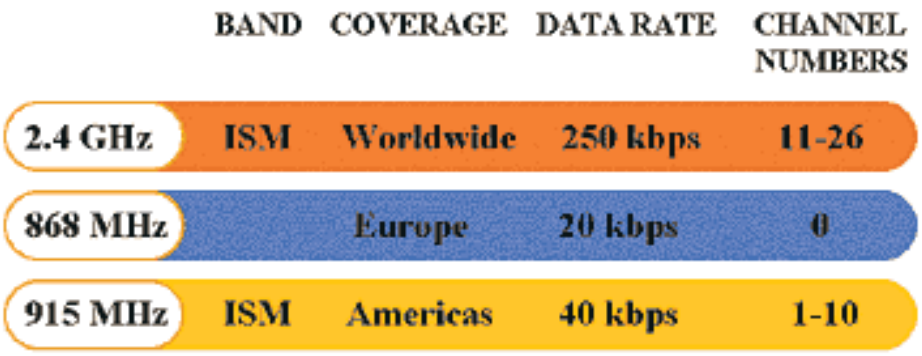
Εικόνα 27-Η δομή του προτοκόλλου Zigbee

Το πρωτόκολλο **Zigbee** αποσκοπεί σε εφαρμογές κτιριακού ελέγχου, στον αυτοματισμό, την ασφάλεια, τα ηλεκτρονικά προϊόντα, τα περιφερειακά Η/Υ, την ιατρική παρακολούθηση και τα παιχνίδια. Οι εφαρμογές αυτές απαιτούν τεχνολογία που επιτρέπει τροφοδότηση με μπαταρίες μεγάλης διάρκειας, αξιοπιστία, αυτόματη ή ημιαυτόματη εγκατάσταση, την δυνατότητα εύκολης προσθήκης ή απομάκρυνσης κόμβων, καθώς και συστήματα χαμηλού κόστους.

Το **Zigbee** και το υποκείμενο πρότυπο 802.15.4, προσφέρουν στο σχεδιαστή του συστήματος συσκευές διαφόρων τάξεων: τη συσκευή μειωμένης λειτουργικότητας (reduced-functionality device, RFD), τη συσκευή πλήρους λειτουργικότητας (full functional device, FFD) και το συντονιστή δικτύου (network coordinator). Όλα τα Zigbee δίκτυα έχουν τουλάχιστον μία από τις παραπάνω συσκευές. Οι περισσότερες εφαρμογές αισθητήρων τοποθετούνται στην RFD κατηγορία, με τα εκτεταμένα δίκτυα να χρησιμοποιούν τόσο τις συσκευές FFD όσο και τους συντονιστές δικτύου προκειμένου να δημιουργήσουν τις απαραίτητες, για την τοπολογία του δικτύου, συνδέσεις. Τα δίκτυα Zigbee σχηματίζονται αυτόνομα, βασισμένα στη συνδεσιμότητα και τη λειτουργία.

Αξιοπιστία δεδομένων

Η αξιόπιστη μεταφορά δεδομένων είναι καθοριστικής σημασίας στις Zigbee εφαρμογές. Το υποκείμενο πρότυπο 802.15.4 παρέχει υψηλή αξιοπιστία μέσω διαφόρων μηχανισμών σε πολλαπλά επίπεδα. Για παράδειγμα, χρησιμοποιεί 27 κανάλια σε 3 διαφορετικές ζώνες συχνοτήτων



Εικόνα 28-Το πρότυπο IEEE 802.15.4

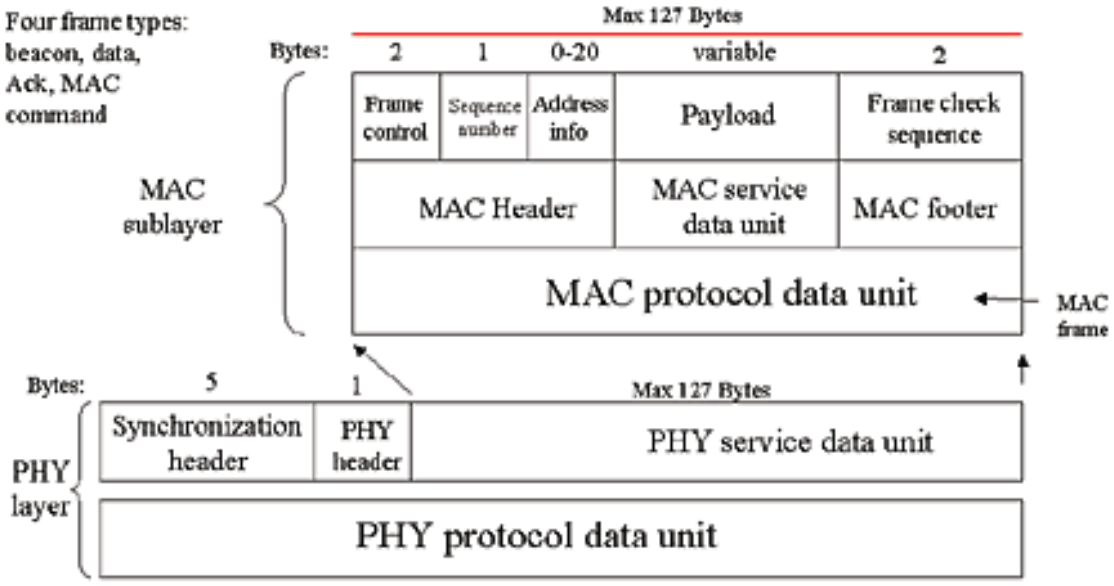
Το πρότυπο IEEE 802.15.4 παρέχει 3 ζώνες συχνοτήτων για επικοινωνία. Οι διαφορές από χώρα σε χώρα στη χρήση, τη διάδοση, τις απώλειες και την ταχύτητα αφήνουν τους σχεδιαστές του Zigbee να βελτιστοποιήσουν την απόδοση του

συστήματος.

Η ζώνη των 2.4 GHz χρησιμοποιείται παγκοσμίως, έχει 16 κανάλια και υποστηρίζει μετάδοση δεδομένων με μέγιστη ταχύτητα 250 Kbps. Έχουν οριστεί επίσης και χαμηλότερες ζώνες συχνοτήτων. Η ζώνη 902–928 MHz παρέχει στην Αμερική και σε μεγάλο μέρος των ακτών του Ειρηνικού 10 κανάλια με μέγιστη ταχύτητα 40 Kbps. Οι ευρωπαϊκές εφαρμογές χρησιμοποιούν 1 κανάλι στη ζώνη 868–870 MHz, με μέγιστη ταχύτητα 20 Kbps. Αυτή η ποικιλία συχνοτήτων επιτρέπει σε εφαρμογές με κατάλληλη ρύθμιση υλικού, να προσαρμόζονται στις τοπικές συνθήκες παρεμβολής και διάδοσης.

Σε ένα συγκεκριμένο κανάλι, ο πομποδέκτης 802.15.4 βασίζεται σε έναν σύνολο μηχανισμών για να βεβαιώσει την αξιόπιστη μετάδοση δεδομένων. Αρχικά, το φυσικό επίπεδο χρησιμοποιεί διαμόρφωση BPSK στις ζώνες 868/915 MHz και O-QPSK στα 2.4 GHz. Και οι δύο διαμορφώσεις είναι στιβαρές και απλές και λειτουργούν καλά σε περιβάλλον χαμηλού σηματοθορυβικού λόγου (SNR).

Η πληροφορία κωδικοποιείται στο φέρον με τεχνική DSSS (Direct-Sequence Spread Spectrum), μια ενδογενώς στιβαρή μέθοδος που βελτιώνει την απόδοση πολλαπλών διαδρομών (multipath performance) και την ευαισθησία του δέκτη μέσω κέρδους από την επεξεργασία σήματος. Το μέγεθος της ωφέλιμης πληροφορίας (data payload) κυμαίνεται από 0 μέχρι 104 bytes, που είναι παραπάνω από αρκετό για να ικανοποιήσει τις περισσότερες ανάγκες των αισθητήρων.



Εικόνα 29-Το πακέτο δεδομένων Zigbee

Στη μονάδα δεδομένων του πρωτοκόλλου MAC, τα ωφέλιμα δεδομένα (data payload) αποτελούνται από τις διευθύνσεις αποστολέα και παραλήπτη, έναν αριθμό που επιτρέπει στον παραλήπτη να αναγνωρίσει ότι όλα τα πακέτα που μεταδόθηκαν έχουν παραληφθεί, bytes ελέγχου πλαισίου που καθορίζουν παραμέτρους του περιβάλλοντος του δικτύου και άλλες σημαντικές και, τέλος, από ένα πεδίο επαλήθευσης που επιτρέπει στον παραλήπτη να πιστοποιήσει ότι το πακέτο παραλήφθηκε χωρίς απώλειες. Αυτό το MAC πλαίσιο παρατίθεται στην επικεφαλίδα συγχρονισμού (synchronization header) και την φυσική επικεφαλίδα (PHY header), του φυσικού επιπέδου, και παρέχει ένα στιβαρό μηχανισμό που επιτρέπει στον παραλήπτη να αναγνωρίσει και να αποκωδικοποιήσει γρήγορα το πακέτο που έχει ληφθεί.

Αφού ληφθεί το πακέτο, ο παραλήπτης εκτελεί έναν 16-bit κυκλικό έλεγχο πλεονασμού (cyclic redundancy check, CRC) για να επιβεβαιώσει ότι το πακέτο δεν αλλοιώθηκε κατά τη μετάδοση. Αν όλα είναι εντάξει, ο παραλήπτης μπορεί, ανάλογα με τις ανάγκες της εφαρμογής και του δικτύου, να μεταδώσει αυτόματα ένα πακέτο γνωστοποίησης που επιτρέπει στο σταθμό μετάδοσης να μάθει ότι το πακέτο παραλήφθηκε σε αποδεκτή μορφή. Αν ο έλεγχος δείξει ότι το πακέτο αλλοιώθηκε, το πακέτο απορρίπτεται και δε μεταδίδεται καμία γνωστοποίηση. Εάν ο σχεδιαστής έχει ρυθμίσει το δίκτυο ώστε να απαιτείται γνωστοποίηση, τότε ο σταθμός μετάδοσης επανεκπέμπει το αρχικό πακέτο, όσες φορές έχει προκαθοριστεί ώστε να εξασφαλιστεί η επιτυχής αποστολή του πακέτου. Αν το μονοπάτι ανάμεσα σε πομπό και δέκτη χάσει την αξιοπιστία του ή το δίκτυο καταρρεύσει, τότε το Zigbee παρέχει στο δίκτυο δυνατότητες επανόρθωσης, εφόσον εναλλακτικά μονοπάτια μπορούν να εγκατασταθούν αυτόνομα.

Διάρκεια πηγής τάσης τροφοδοσίας

Ένας ασύρματος κόμβος δικτύων αισθητήρων, όπως σε οποιοδήποτε υπολογιστή γενικού σκοπού, αποτελείται από επεξεργαστή, μνήμη αποθήκευση, συσκευές επικοινωνίας, και συσκευές εισόδου εξόδου. Αλλά, "ασύρματο" δεν αναφέρεται στις επικοινωνίες αλλά ισχύει και για την πηγή ενέργειας. Το πιο σημαντικό κομμάτι του ασύρματου κόμβου δικτύων αισθητήρων είναι η συσκευή ενεργειακής αποθήκευσης, συνήθως μια μπαταρία. Πάνω από όλα είναι η συνειδητοποίηση αυτής της πεπερασμένης ενεργειακής πηγής που οδηγεί τον σχεδιασμό του υπόλοιπου

συστήματος. Οι ασύρματες συσκευές ενεργειακής σάρωσης μπορούν να βοηθήσουν στην παράταση του ενεργειακού αποθέματος, αλλά το εύρος επέκτασης δικτύων αισθητήρων διαμορφώνεται τελικά από το ποσό ενέργειας που διαθέτει κάθε κόμβος. Όταν η ενέργεια τελειώσει, είτε η εφαρμογή σταματά είτε κάποιο πρόσωπο πρέπει να σταλεί επί τόπου για να ανανεώσει της πηγές, και επίσης να συλλέξει τα δεδομένα.

Αρχίζουμε με την εξέταση των διαθέσιμων επιλογών ενεργειακής αποθήκευσης. Η βασική αλκαλική μπαταρία AA αποθηκεύει 2850 ώρες μA ενέργειας. Ένας λαμπτήρας LED καταναλώνει περίπου 6 mA ρεύματος. Αυτός ο λαμπτήρας θα παραμείνει αναμμένος για περίπου 20 ημέρες, και προς το τέλος αυτού του χρονικού διαστήματος, θα γίνει πιο αμυδρός ο φωτισμός όσο η τάση πέφτει κάτω από 1,5 βολτ. Τώρα εξετάζουμε την απλούστερη και αποτελεσματικότερη μορφή εναλλακτικής ενέργειας, την ηλιακή. Ένα αντιπροσωπευτικό φωτοβολταικό επιφάνειας 30cm^2 μπορεί να παραγάγει ρεύμα 40 mA σε 4,8 βολτ, και περίπου 6 mW/*cm^2 με άμεσο φως του ήλιου. Αυτό απέχει από το συνολικό ποσό διαθέσιμης ηλιακής ενέργειας,
περίπου 100 mW/*cm^2, αλλά η αποδοτικότητα των φωτοβολταικών μονάδων αυξάνει.

Πόση ενέργεια χρειάζονται αυτές οι συσκευές; Καθώς οι σχεδιαστές μικροεπεξεργαστών έχουν κατανοήσει το κενό μπαταρίας-χαμηλής ενέργειας το ρεύμα που απαιτούν οι επεξεργαστές μειώνεται. Ο μικροελεγκτής του Tmote MSP430 έχει ενεργό ρεύμα λειτουργίας 3 mW, αρκετό να τρέξει για έναν μήνα ή δύο στις τυποποιημένες μπαταρίες AA.

Αλλά, αυτό αγνοώντας το κόστος ραδιοεκπομπής. Μεταδίδοντας ένα μήνυμα, ο πομπός καταναλώνει ισχύ 35 mW, και αυτό το κόστος έχει παραμείνει κατά προσέγγιση σταθερό στη διάρκεια των ετών. Η ενέργεια που απαιτείται για να μεταδοθεί σε μια δεδομένη απόσταση d ανάλογη προς d^n, όπου το n ποικίλλει. Η ασύρματη μετάδοση απαιτεί πολλή ενέργεια, αλλά, δεν γίνεται συνεχώς.

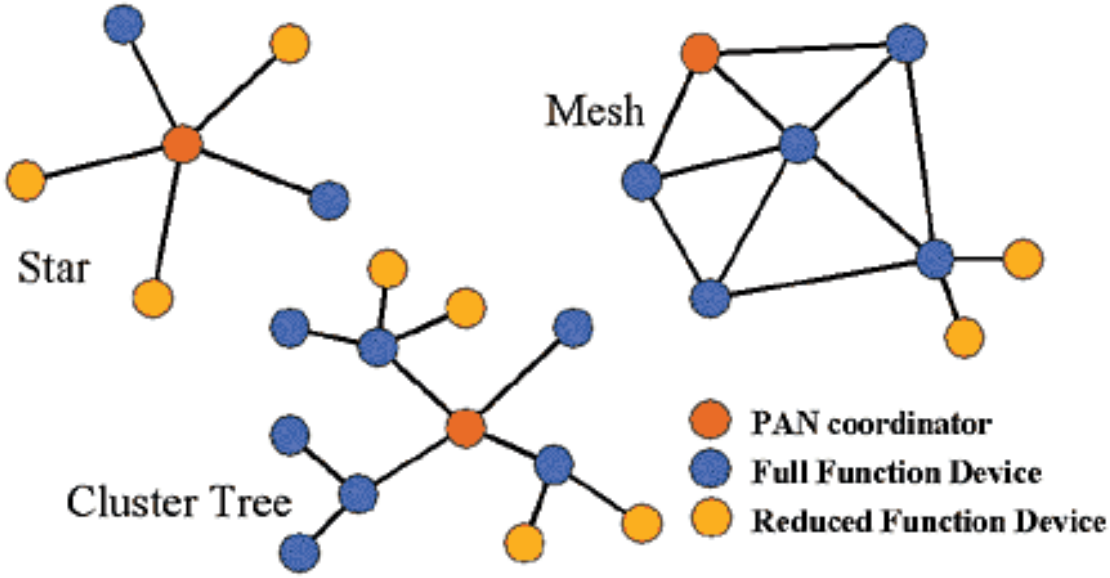
Εντούτοις, η λήψη γίνεται συνέχεια. Επιπλέον, δεδομένου ότι οι πομποί έχουν γίνει πιο σύνθετοι, το κόστος λήψης για ένα μήνυμα έχει αυξηθεί δραστικά. Το radio που χρησιμοποιήθηκε στα πρώτα motes UCB κατανάλωνε 9 mW ισχύ περιμένοντας τα μηνύματα. Στα νεότερα motes καταναλώνει 38 mW ισχύος κατά τη λήψη. Το πραγματικό ενεργειακό κόστος της ασύρματης επικοινωνίας δεν είναι στη μετάδοση αλλά στην αναμονή λήψης.

Το συμπέρασμα από την εξέταση των σχετικών ενεργειακών δαπανών του επεξεργαστή και του πομπού είναι ότι για επεξεργασία μιας εντολής (instruction) του CPU με κατανάλωση ισχύος 3 mW σε ένα ρολόι 4MHz απαιτούνται 0,75 nJ ανά εντολή. Για την αποστολή ή λήψη ενός bit για κατανάλωση 35 mW σε ρυθμό 250kbit/ανα κανάλι απαιτούνται 140 nJ ανά εντολή, περίπου 200 φορές περισσότερη ενέργεια. Η λειτουργία του πομπού στοιχίζει πολύ πιο ακριβά ενεργειακά σε σχέση με την επεξεργασία άρα όπου ιεναι δύνατό χρησιμοποιήούμε την επεξεργασία για να ελαττώσετε τον χρόνο εκπομπής εξοικονόμόντας έτσι τεράστια ενέργεια. Αυτό εξυπακούει επεξεργασία μέσα στο δίκτυο, συμπίεσης των δεδομένων, και χρήση της λογικής επεξεργασίας των στοιχείων μέσα στα motes αντί να στέλνει τα ακατέργαστα στοιχεία. Φυσικά, αυτές οι λύσεις εισάγουν τις δικές τους περιπλοκές. Όταν εξετάζουμε το συνδυασμένο κόστος λειτουργίας της πλατφόρμας, 3mW για τον επεξεργαστή και 38 mW για τον ραδιοπομπό, ένα πράγμα γίνεται προφανές. Το συνολικό κόστος 41 mW θα εξαντλήσει ένα ζευγάρι μπαταρίες AA σε μια εβδομάδα. Σαφώς, αυτό είναι πάρα πολύ σύντομο για μια αποτελεσματική υλοποίηση WSN. Ο προτεινόμενος προτείνουν ότι ο ελάχιστος οικονομικώς αποδοτικός χρόνος εφαρμογής για χρησιμοποίηση των ασύρματων δικτύων αισθητήρων είναι ένα με δύο έτη.

Σε πολλές εφαρμογές, δεν είναι εύκολη η συχνή αλλαγή του στοιχείου τροφοδοσίας (μπαταρία) του αισθητήρα. Ο βασικός 802.15.4 κόμβος είναι σημαντικά αποτελεσματικός όσον αφορά την απόδοση της μπαταρίας. Η διάρκεια της μπαταρίας από λίγους μήνες μπορεί να φτάσει τα πολλά χρόνια, όταν στο σύστημα υπάρχουν κόμβοι που εξοικονομούν ενέργεια και παράμετροι δικτύου που βελτιστοποιούν την κατανάλωση ενέργειας, όπως είναι η σήμανση διαλειμμάτων (beacon intervals), οι καθορισμένες χρονοθυρίδες (guaranteed time slots) και οι δυνατότητες ενεργοποίησης/απενεργοποίησης (enablement/disablement options).

Η υλοποίηση του δικτύου παίζει επίσης σημαντικό ρόλο. Τα περισσότερα δίκτυα θεωρούνται ότι έχουν δομή τύπου αστέρα (star) ή συστοιχίας δέντρων (cluster trees), παρά πραγματικού βροχωτού δικτύου (mesh network) επιτρέποντας στις ανεξάρτητες συσκευές να εξοικονομούν ενέργεια. Για μεγαλύτερα φυσικά περιβάλλοντα, ο τύπος ‘συστοιχία δέντρων’ είναι ένας καλός τρόπος να συγκεντρώνονται πολλαπλά δίκτυα τύπου αστέρα σε ένα ευρύτερο δίκτυο. Κάποιες εφαρμογές κάνουν χρήση της βροχωτής (mesh) δομής, που παρέχει ευελιξία στην

αλλαγή δρομολόγησης και τη δυνατότητα στο δίκτυο να επανορθώνεται μόνο του όταν ενδιάμεσοι κόμβοι απομακρύνονται ή τα RF μονοπάτια αλλάζουν.



Εικόνα 30-Μερικές μορφές τοπολογίας δικτύων

Κόστος

Το ZigBee και το 802.15.4 μεγιστοποιούν τη χρησιμότητά τους στον πολυδιάστατο τομέα του κόστους. Υπάρχει επαρκής ευελιξία και στα 2 πρότυπα ώστε να παρέχουν στο σχεδιαστή του συστήματος αισθητήρων μια ποικιλία τρόπων βελτιστοποίησης του κόστους, χωρίς να παραμελείτε η απόδοση του συστήματος. Για παράδειγμα, η διάρκεια της μπαταρίας μπορεί να βελτιωθεί με αύξηση του χρόνου μη εξυπηρέτησης, όπως επίσης, το κόστος και η πολυπλοκότητα κάθε κόμβου θα βελτιωθούν σε βάρος της πολυπλοκότητας του δικτύου.

Η απλότητα του συστήματος και η ευελιξία του πρότυπου 802.15.4 υπόσχονται στους σχεδιαστές του συστήματος ότι θα βρουν τις πλατφόρμες που βασίζονται στο πρωτόκολλο Zigbee, περισσότερο αποτελεσματικές όσον αφορά το κόστος (για μονάδες ίδιου όγκου) από το Bluetooth ή από άλλες αμφίδρομες ασύρματες λύσεις. Ενόσω το κόστος του υλικού των πλατφορμών είναι πάντα κρίσιμο μέρος του συνολικού κόστους του συστήματος, πρέπει να λαμβάνονται επίσης υπόψη τα κόστη της συντήρησης του συστήματος, της ευελιξίας και της διάρκειας ζωής της μπαταρίας.

Εύρος Μετάδοσης

Το Zigbee στηρίζεται στο βασικό 802.15.4 πρότυπο για να εγκαθιστά τη ραδιοεπικοινωνία. Αφού το 802.15.4 είναι ένα πρότυπο ασύρματης επικοινωνίας μικρής εμβέλειας, δεν προσπαθεί να ανταγωνιστεί πομπούς υψηλής ισχύος αλλά υπερέχει σε διάρκεια ζωής μπαταρίας και σε χαμηλής ισχύος μετάδοση. Το πρότυπο καθορίζει ονομαστική τιμή ισχύος εκπομπής στα –3 dBm (0.5 mW), με το άνω όριο να ελέγχεται από τις κανονιστικές αρχές (regulatory agencies) της χώρας όπου θα χρησιμοποιηθεί ο αισθητήρας. Σε έξοδο –3 dBm, τα single-hop ranges από 10 μέχρι και πάνω από 100m είναι λογικά, ανάλογα με το περιβάλλον, την κεραία και το φάσμα συχνοτήτων λειτουργίας.

Το Zigbee επεκτείνει το βασικό 802.15.4 πομπό και πρωτόκολλο με μια λειτουργία δικτύου που επιτρέπει multi-hop και ευέλικτη δρομολόγηση, παρέχοντας εύρη επικοινωνίας που ξεπερνούν τη βασική single-hop. Πράγματι, ανάλογα με τις απαιτήσεις για τη λανθάνουσα καθυστέρηση των δεδομένων (data latency), μπορούν πρακτικά να δημιουργηθούν δίκτυα που χρησιμοποιούν δεκάδες κόμβους (hops), με εύρη που αθροιζόμενα φτάνουν από εκατοντάδες σε χιλιάδες μέτρα. Τα δίκτυα μπορούν να έχουν δομή τύπου αστέρα, συστοιχίας δέντρων ή βροχωτού δικτύου, με την καθεμία να παρουσιάζει τις δικές της δυνατότητες.

Ρυθμός Μετάδοσης δεδομένων

Μπορεί να μην είναι εμφανές για ποιο λόγο ένας απλός αισθητήρας θερμοκρασίας ή εντοπισμού εισβολής χρειάζεται να μεταδίδει δεδομένα με 250 Kbps (στα 2.4 GHz) ή ακόμα με 20 Kbps (στα 868 MHz), αλλά ξεκαθαρίζει εάν αναλογιστούμε την ανάγκη για επέκταση της διάρκειας της μπαταρίας. Ακόμα και όταν ο αισθητήρας μεταδίδει μόνο μερικά bits ή bytes, το σύστημα μπορεί να καταστεί πιο αποτελεσματικό αν μεταδίδει και λαμβάνει δεδομένα γρήγορα. Για παράδειγμα, ένας πομπός ισχύος 0.5mW καταναλώνει πολλά milliwatts είτε μεταδίδει με 100 ή με 100.000 bps. Για κάθε συγκεκριμένο ποσό δεδομένων, η μετάδοση σε υψηλότερο ρυθμό επιτρέπει στο σύστημα να κλείνει γρηγορότερα τον πομπό και το λήπτη, εξοικονομώντας σημαντική ενέργεια.

Υψηλότεροι ρυθμοί δεδομένων για συγκεκριμένο επίπεδο ισχύος, σημαίνει ότι υπάρχει μικρότερη ενέργεια ανά μεταδιδόμενο bit, που υποδηλώνει περιορισμένο εύρος. Τόσο το 802.15.4 όσο και το ZigBee αξιολογούν τη διάρκεια της μπαταρίας

περισσότερο από το εύρος κάλυψης και παρέχουν μηχανισμούς που αυξάνουν το εύρος αυτό ενώ είναι πάντα επικεντρωμένοι στη διάρκεια της μπαταρίας

Λανθάνουσα καθυστέρηση δεδομένων

Τα συστήματα αισθητήρων έχουν μεγάλες απαιτήσεις όσον αφορά τη λανθάνουσα καθυστέρηση δεδομένων. Αν γίνει αναγκαία η λήψη των δεδομένων του αισθητήρα μέσα σε δεκάδες milliseconds, σε αντίθεση με τις δεκάδες δευτερολέπτων, τότε αλλάζουν οι απαιτήσεις του δικτύου όσον αφορά τον τύπο και την έκτασή του. Για πολλές εφαρμογές αισθητήρων, η λανθάνουσα καθυστέρηση δεδομένων είναι λιγότερο κρίσιμη από τη διάρκεια της μπαταρίας ή την αξιοπιστία των δεδομένων.

Για απλά δίκτυα τύπου αστέρα (πολλοί πελάτες, ένας συντονιστής δικτύου), το Zigbee μπορεί να παρέχει λανθάνουσες καθυστερήσεις τάξης ~16 ms σε ένα δίκτυο βασισμένο σε σήμανση (beacon-centric network). Μπορούμε να μειώσουμε περαιτέρω τις καθυστερήσεις σε μερικά milliseconds αν ξεφύγουμε από το μοντέλο που βασίζεται σε σήμανση (beacon environment) και είμαστε διατεθειμένοι να ρισκάρουμε ενδεχόμενη παρεμβολή από τυχαία σύγκρουση δεδομένων με άλλους αισθητήρες του δικτύου.

Η λανθάνουσα καθυστέρηση δεδομένων μπορεί να επηρεάσει τη διάρκεια της μπαταρίας. Γενικά, αν χαλαρώσουμε τις απαιτήσεις για την καθυστέρηση των δεδομένων, περιμένουμε η διάρκεια ζωής της μπαταρίας των κόμβων-πελατών να αυξάνει. Αυτό συμβαίνει ακόμα περισσότερο στους κεντρικούς σταθμούς του δικτύου (network hubs), που απαιτούνται για να συντονίσουν και να επιθεωρήσουν το δίκτυο. Ας υποτεθεί ότι ένα απλό δίκτυο έχει μεγάλες απαιτήσεις όσον αφορά τη λανθάνουσα καθυστέρηση δεδομένων (π.χ. ένα ασύρματο πληκτρολόγιο και ποντίκι Η/Υ). Ο χρήστης περιμένει ότι ένα χτύπημα στο πληκτρολόγιο ή μια κίνηση του ποντικιού θα εμφανιστεί στην οθόνη μέσα σε 1 ή 2 ανανεώσεις της οθόνης, γενικά μεταξύ 16 και 32 ms. Για ένα τέτοιο είδος δικτύου τύπου αστέρα, μπορούμε να περιμένουμε ότι η λανθάνουσα καθυστέρηση δεδομένων θα ανταποκριθεί σ' αυτήν την απαίτηση.

Ασφάλεια δεδομένων

Όπως αναφέραμε ξανά, είναι σημαντικό να εφοδιαστεί ένα δίκτυο αισθητήρων με επαρκή ασφάλεια που θα εμποδίζει τα δεδομένα να εκτεθούν σε κίνδυνο, να

κλαπούν ή να αλλοιωθούν. Το πρότυπο IEEE 802.15.4 παρέχει υπηρεσίες επαλήθευσης, κρυπτογράφησης και διασφάλισης ακεραιότητας για ασύρματα συστήματα, που επιτρέπουν στους σχεδιαστές να καθορίσουν οι ίδιοι τα επίπεδα ασφαλείας. Αυτά περιλαμβάνουν την απουσία κάθε ασφάλειας, λίστες ελέγχου πρόσβασης και 32-bit μέχρι 128-bit κρυπτογράφηση AES (Advanced Encryption Standard) με επαλήθευση. Αυτό το πακέτο επιλογών ασφάλειας επιτρέπει στο σχεδιαστή να διαλέξει την ασφάλεια που απαιτεί η εφαρμογή, που είναι, σε ελεγχόμενα πλαίσια, εις βάρος του όγκου δεδομένων, της διάρκειας της μπαταρίας και των απαιτήσεων σε επεξεργαστική ισχύ. Το πρότυπο IEEE 802.15.4 δεν παρέχει κάποιον μηχανισμό μετακίνησης των κλειδιών ασφαλείας γύρω από ένα δίκτυο. Αντίθετα την ανάγκη αυτή καλύπτει το Zigbee.

Το Zigbee περιλαμβάνει σημαντικά στοιχεία που επιτρέπουν την ασφαλή διαχείριση του δικτύου από απόσταση. Για εκείνα τα συστήματα όπου η ασφάλεια των δεδομένων δεν είναι σημαντικός παράγοντας (π.χ. μια ομάδα αισθητήρων που παρακολουθεί το κλίμα σε ένα δάσος), μπορούμε να αποφασίσουμε αντί να περιλάβουμε στοιχεία ασφαλείας, να βελτιώσουμε τη διάρκεια της μπαταρίας και να μειώσουμε το κόστος του συστήματος. Για το σχεδιαστή ενός περιμετρικού συστήματος ασφαλείας με αισθητήρες, σε μια βιομηχανική ή στρατιωτική περιοχή, η ασφάλεια των δεδομένων, και πολύ περισσότερο η δυνατότητα αντιμετώπισης προσπαθειών εξαπάτησης των αισθητήρων, πρέπει να έχει τη μεγαλύτερη προτεραιότητα. Η ασφάλεια, επίσης, μετάδοσης δεδομένων που μπορούν να θεωρηθούν προσωπικά, όπως για παράδειγμα τα φυσιολογικά δεδομένα ενός ατόμου που μεταδίδονται μέσα σε ένα ασύρματο δίκτυο αισθητήρων, μπορεί να ληφθεί υπόψη από το σχεδιαστή του συστήματος.

Το Zigbee συγκριτικά με εναλλακτικές τεχνολογίες

Υπάρχει ένας αριθμός άλλων ασύρματων τεχνολογιών για μεταδόσεις διαφόρων ταχυτήτων, σε οικιακές, εμπορικές και βιομηχανικές εφαρμογές (π.χ. το Bluetooth, το IEEE 802.11 Wi-Fi και ιδιοταγή συστήματα). Καθεμιά κατέχει ιδιαίτερη θέση στον τομέα της ασύρματης επικοινωνίας αλλά, δεν έχει επιτευχθεί ακόμη η βέλτιστη αλληλοκάλυψη. Για εφαρμογές αισθητήρων όχι πολύ υψηλών ταχυτήτων, η τεχνολογία Bluetooth μπορεί να θεωρηθεί ικανοποιητική. Το πρότυπο αυτό δύναται να σχηματίσει δίκτυα peer-to-peer ή δίκτυα σε σχηματισμό αστέρα (star networks), αλλά αυτά δεν υποστηρίζουν περισσότερες από 8 ενεργές συσκευές συγχρόνως. Το

σχήμα της φασματικής εξάπλωσης με αναπήδηση συχνότητας (frequency hopping spread spectrum, FHSS) του Bluetooth, αναγκάζει συσκευές που δεν έχουν ακόμα ενσωματωθεί στο δίκτυο, να επανασυγχρονίζονται για 3-30 secs πριν να είναι ικανές να απαιτήσουν σύνδεση, κάνοντας το χρόνο απόκρισης σε διακοπτόμενη λειτουργία αρκετά μεγάλο για πολλές εφαρμογές. Και για ένα σύστημα προορισμένο για μεγάλης διάρκειας λειτουργία, με μπαταρία, η ενέργεια που καταναλώνεται κατά τον συγχρονισμό του δικτύου μπορεί να είναι απαγορευτική.

Μολονότι η τεχνολογία Bluetooth είναι κατάλληλη για εφαρμογές φωνής και εφαρμογές υψηλότερων ταχυτήτων (π.χ. κινητά και σταθερά τηλέφωνα), η τεχνολογία Zigbee είναι περισσότερο κατάλληλη για εφαρμογές ελέγχου, που δεν απαιτούν υψηλούς ρυθμούς δεδομένων αλλά πρέπει να έχουν μεγάλη διάρκεια μπαταρίας, δίκτυα ποικίλης τοπολογίας και χαμηλή παρέμβαση από το χρήστη. Επίσης, η στοίβα του Zigbee είναι μικρή (28 KB) συγκρινόμενη με εκείνη του Bluetooth (250 KB). Είναι σημαντικό να σημειωθεί ότι η πάρα πολύ χαμηλή κατανάλωση ενέργειας είναι το κύριο σχεδιαστικό στοιχείο του προτύπου Zigbee, επιτρέποντας συσκευές αυξημένης διάρκειας λειτουργίας, ακόμα και με μπαταρίες μη επαναφορτιζόμενες, σε αντίθεση με τις επαναφορτιζόμενες συσκευές που υποστηρίζει το Bluetooth. Για παράδειγμα, η μετάβαση από την κατάσταση αδρανείας (sleep mode) στην κατάσταση μετάδοσης δεδομένων είναι γρηγορότερη στα Zigbee συστήματα συγκριτικά με εκείνα που χρησιμοποιούν Bluetooth. Τα Zigbee δίκτυα μπορούν να υποστηρίξουν τουλάχιστον 65.534 συσκευές ανά δίκτυο, σε αντίθεση με τις 8 στα Bluetooth δίκτυα. Ο μέγιστος ρυθμός δεδομένων στην τεχνολογία ZigBee είναι 250 Kbps, ενώ στην Bluetooth είναι 1 Mbps. Στον επόμενο πίνακα παρατίθενται για σύγκριση κάποια βασικά χαρακτηριστικά του Zigbee και άλλων ασύρματων τεχνολογιών.

Standard	**ZigBee™ 802.15**	**Wi-Fi™ 802.11**	**Bluetooth™ 802**
Transmission Range (m	1 – 100*	1 - 100	1 – 10
Battery Life (days)	100 – 1,000	0.5 – 5.0	1 - 7
Network Size (# of node	> 64,000	32	7
Application	Monitoring & Con	Web, Email, Vic	Cable Replacem
Stack Size (KB)	4 – 32	1,000	250
Throughput (kb/s)	20 – 250	11,000	720

Κάποια βασικά χαρακτηριστικά του Zigbee σε σχέση με άλλα πρότυπα ασύρματης μετάδοσης

Η ασύρματη πλατφόρμα Tmote Sky (Moteiv)

Η πλατφόρμα που χρησιμοποιήσαμε στην εφαρμογή μας είναι η πλατφόρμα Tmote Sky από την εταιρεία Moteiv. Το tmote-sky είναι μια ασύρματη μονάδα (“mote”) πολύ χαμηλής κατανάλωσης ισχύος, για χρήση σε δίκτυα αισθητήρων και σε εφαρμογές καταγραφής και παρακολούθησης σχεδιασμένες με σκοπό τόσο την ανεκτικότητα στο θόρυβο όσο και την ευκολία περαιτέρω ανάπτυξης και αξιοποίησης. Αποτελεί εξέλιξη του Telosb και είναι το πιο πρόσφατο προϊόν σε μια σειρά από motes που αναπτύχθηκαν από το Πανεπιστήμιο της California, Berkeley με σκοπό τη χρήση τους σε ασύρματα δίκτυα αισθητήρων.

Εικόνα 31-Η ασύρματη μονάδα Tmote sky

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

Βασικά γνωρίσματα

• Ασύρματος πομποδέκτης 250kbps 2.4GHz IEEE 802.15.4 Chipcon

• Μικροελεγκτής 8MHz Texas Instruments MSP430 (10k RAM, 48k Flash)

• Ολοκληρωμένος ADC, DAC, Supply Voltage Supervisor και ελεγκτής DMA

• Onboard κεραία με εμβέλεια 50m σε εσωτερικούς χώρους / 125m σε εξωτερικούς.

• Ενσωματωμένοι αισθητήρες υγρασίας, θερμοκρασίας και φωτός.

• Χαμηλή κατανάλωση ρεύματος

• Γρήγορη αφύπνιση (<6μs)

• Κωδικοποίηση και πιστοποίηση αυθεντικότητας στο στρώμα ζεύξης υλικού

• Προγραμματισμός και συλλογή δεδομένων μέσω USB

•Υποστήριξη επέκτασης 16pin και προαιρετικός συνδετήρας SMA για εξωτερική κεραία

•Υποστήριξη λειτουργικού συστήματος **TinyOS**

Κυριότερα τεχνικά χαρακτηριστικά

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

Humidity Sensor	
Humidity Accuracy	3.5% RH
Temperature Accuracy	0.5 °C
Sampling Rate	90 Hz

Τεχνικά χαρακτηριστικά μονάδας Tmote Sky

Η υλοποίηση και ανάπτυξη του tmote στηρίχθηκε σε τρεις βασικούς στόχους: την ακόμα χαμηλότερη κατανάλωση ενέργειας σε σχέση με τις προηγούμενες γενιές πλατφορμών, την ευκολία χρήσης και την ευρωστία περαιτέρω ανάπτυξης και πειραματισμού. Ο σχεδιασμός της μονάδας Tmote Sky στηρίζεται στην ακόλουθη βασική αρχή που αναφέραμε και προηγουμένως: Η μονάδα-κόμβος βρίσκεται σε αδράνεια στο σύνολο του χρόνου, αφυπνίζεται άμεσα με την ύπαρξη ενός συμβάντος, επεξεργάζεται το συμβάν και επιστρέφει σε αδράνεια. Η ολοκληρωμένη σχεδίαση του προσφέρει όμως κάτι παραπάνω από απλά χαμηλή κατανάλωση ενέργειας κατά τη λειτουργία του. Επιτρέπει στους σχεδιαστές να εκμεταλλευτούν την αυξημένη λειτουργικότητά του και να αναπτύξουν πιο εύρωστα συστήματα. Στις επόμενες παραγράφους περιγράφονται πιο αναλυτικά τα κύρια χαρακτηριστικά της μονάδας tmote καθώς και τα πλεονεκτήματα της σε σχέση με άλλες πλατφόρμες, πλεονεκτήματα που μας οδήγησαν τελικά και στην επιλογή του προϊόντος αυτού για την εφαρμογή μας.

Τεχνολογικές τάσεις

Από την στιγμή που κυκλοφόρησε η μονάδα Mica2 το 2002, εμφανίστηκε ένα πλήθος νέων μικροελεγκτών που προσέφεραν μικρότερη κατανάλωση ενέργειας, περισσότερα ενσωματωμένα περιφερειακά και ποικίλα μεγέθη σε μνήμες RAM και flash.

Manufacturer	Device	RAM (kB)	Flash (kB)	Active (mA)	Sleep (μA)	Release
Atmel	AT90LS8535	0.5	8	5	15	1998
	Mega128	4	128	8	20	2001
	Mega165/325/645	4	64	2.5	2	2004
General Instruments	PIC	0.025	0.5	19	1	1975
Microchip	PIC Modern	4	128	2.2	1	2002
Intel	4004 4-bit	0.625	4	30	N/A	1971
	8051 8-bit Classic	0.5	32	30	5	1995
	8051 16-bit	1	16	45	10	1996
Philips	80C51 16-bit	2	60	15	3	2000
Motorola	HC05	0.5	32	6.6	90	1988
	HC08	2	32	8	100	1993
	HCS08	4	60	6.5	1	2003
Texas	TSS400 4-bit	0.03	1	15	12	1974
Instruments	MSP430F14x 16-bit	2	60	1.5	1	2000
	MSP430F16x 16-bit	10	48	2	1	2004
Atmel	AT91 ARM Thumb	256	1024	38	160	2004
Intel	XScale PXA27X	256	N/A	39	574	2004

Η ιστορία των μικροελεγκτών. Ο κύριος πίνακας παρουσιάζει παραδοσιακούς μικροελεγκτές και οι τελευταίες 2 συσκευές είναι 32-bit μικροελεγκτές που παρατίθενται για σύγκριση.

Η μονάδα tmote χρησιμοποιεί τον μικροελεγκτή MSP430, ο οποίος, όπως φαίνεται και από τον παραπάνω πίνακα, έχει τη χαμηλότερη κατανάλωση ενέργειας σε καταστάσεις αδράνειας και ενεργής λειτουργίας. Ο μικροελεγκτής αυτός λειτουργεί με ελάχιστη τάση 1.8 V. Η απαίτηση χαμηλών τιμών τάσης είναι σημαντική για την εξαγωγή όλης της ενέργειας από μια πηγή τάσης. Για παράδειγμα, οι μπαταρίες τύπου AA έχουν τάση αποκοπής στα 0.9V. Αν χρησιμοποιηθούν 2 μπαταρίες σε σειρά, η τάση αποκοπής του συστήματος είναι 1.8V, ακριβώς η ίδια με την ελάχιστη τάση που απαιτεί ο MSP430. Αντίθετα, ο μικροελεγκτής ATmega128 MCU (Mica family) λειτουργεί με ελάχιστη τάση 2.7V, αφήνοντας αχρησιμοποίητο σχεδόν το 50% των μπαταριών τύπου AA. Ο MSP430 έχει επιπλέον τον ταχύτερο χρόνο αφύπνισης από όλους τους μικροελεγκτές, μεταβαίνοντας από την κατάσταση αναμονής (standby 1μA) στην κατάσταση λειτουργίας το πολύ σε 6 μs. Επίσης, διαθέτει έναν ελεγκτή DMA (Direct Memory Access controller) προσφέροντας τη δυνατότητα μείωσης του φορτίου στον πυρήνα του μικροελεγκτή και της κατανάλωση ενέργειας, καθώς και αύξησης της απόδοση.

Η τάση που επικρατεί είναι να παραμένουν περίπου σταθερά τα μεγέθη των

μνημών RAM και flash (όπως εμφανίζονται στον Πίνακα) και να προστίθενται επιπλέον στοιχεία επιτάχυνσης (accelerator modules) υλικού. Ο MSP430 παρέχει τη μεγαλύτερη ενσωματωμένη ενδιάμεση μνήμη RAM (RAM buffer 10 KB), χρήσιμη σε περιπτώσεις on-chip επεξεργασίας σήματος. Η δυνατότητα αποθήκευσης σε μεγαλύτερες μνήμες RAM παρόλο που μπορεί να φανεί χρήσιμη σε πιο απαιτητικές εφαρμογές δεν αποτελεί μέχρι σήμερα, γενικά, περιοριστικό παράγοντα στην εξέλιξη των εφαρμογών δικτύων ασύρματων αισθητήρων (WSN applications).

Ασύρματος πομποδέκτης

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

Radiation Pattern

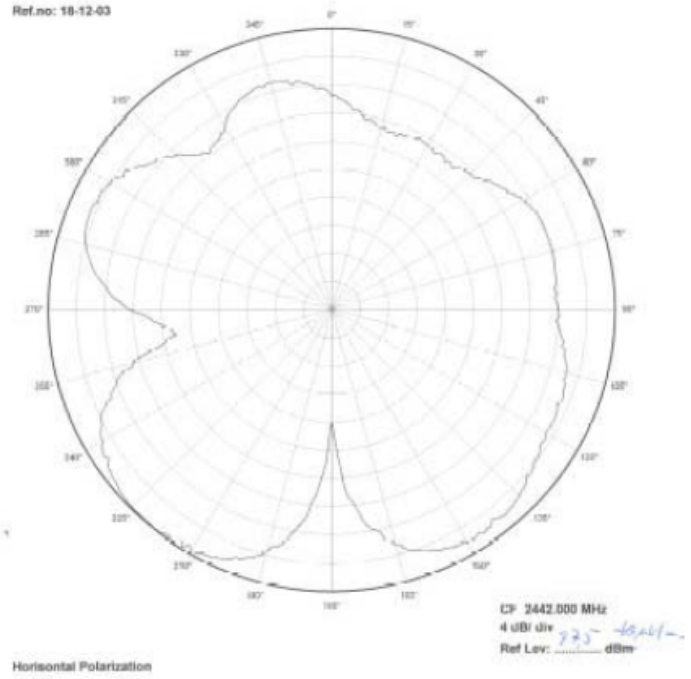


Figure 12 : Radiated pattern of the Inverted-F antenna with horizontal mounting (from Chipcon AS)

Εικόνα 32-Πολικό διάγραμμα της κεραίας σε οριζόντια εκπομπή.

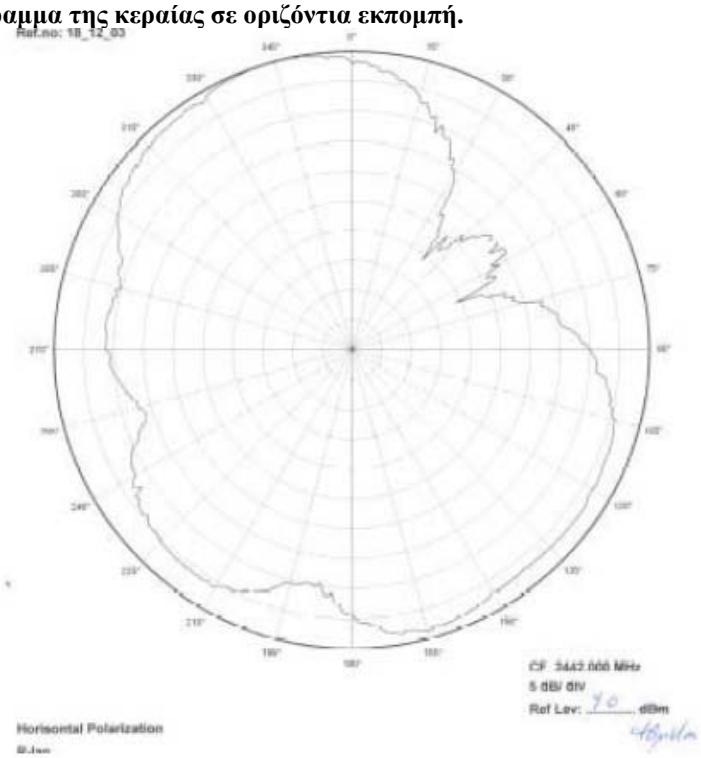


Figure 13 : Radiated pattern of the Inverted-F antenna with vertical mounting (from Chipcon AS)

Εικόνα 33-Πολικό διάγραμμα της κεραίας σε κατακόρυφη εκπομπή.

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

Type	Narrowband				Wideband		
Vendor Part	**RFM** TR100	**Chipcor** CC1000	**Chipcor** CC2400	**Nordic** nRF2401	**Chipcor** CC2420	**Motorola** MC13191/9	**Zeevo** ZV4002
Max Data (kbps)	115.2	76.8					723.2
RX power (m	3.8	9.6	24	18 (25)	19.7	37(42)	65
TX power (mA/dBm)	12/ 1.5	16.5 / 10	19/0	13/0	17.4 / 0	34(30)/ 0	65/ 0
Powerdown power (µA)	1	1	1.5	0.4	1	1	140
Turn on time	0.02	2	1.13	3	0.58	20	*
Modulation	OOK/A	FSK	FSK,GFS	GFSK	DSSS-O-	DSSS-O-QP	FHSS-G
Packet detec	no	no	programn	yes	yes	yes	yes
Address decoding	no	no	no	yes	yes	yes	yes
Encryption support	no	no	no	no	128-bit AI	no	128-bit
Error detecti	no	no	yes	yes	yes	yes	yes
Error correct	no	no	no	no	yes	yes	yes
Acknowledgm s	no	no	no	no	yes	yes	yes
Interface	bit	byte	packet/by	packet/by	packet/by	packet/byte	packet
Buffering (by	no	1	32	16	128	133	yes *
Time-sync	bit	SFD/byte	SFD/pack	packet	SFD	SFD	Bluetoo
Localization	RSSI	RSSI	RSSI	no	RSSI/LQI	RSSI/LQI	RSSI

Χαρακτηριστικά σύγχρονων πομποδεκτών (COTS radios, commercial offthe-shelf) ιδανικών για δίκτυα ασύρματων αισθητήρων .

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

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

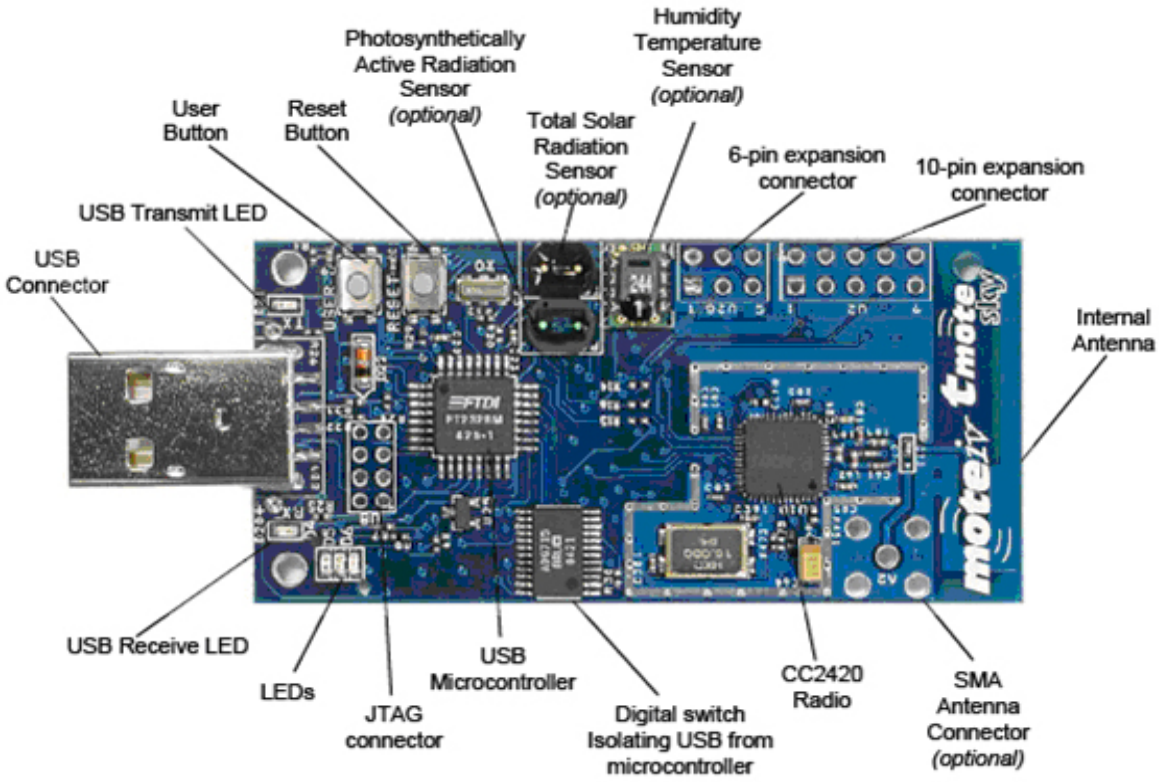
Οι τυπικές συνθήκες λειτουργίας του ασύρματου πομποδέκτη φαίνονται στον επόμενο πίνακα.

	MIN	NOM	MAX	Μονάδ
Τάση λειτουργίας κατά την ασύρματη εκπ (Vreg on)	2.1		3.6	V
Θερμοκρασία λειτουργίας	-40		85	°C
Εύρος συχνοτήτων RF	2400		2483.5	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 dBm		17.4		mA
Κατανάλωση ρεύματος: ασύρματη Λήψη		19.7		mA
Κατανάλωση ρεύματος: Radio on, ταλαντωτή		365		μA
Κατανάλωση ρεύματος: κατάσταση αδρά ταλαντωτής off		20		μA
Κατανάλωση ρεύματος: κατάσταση λειτουργίας, Vreg off			1	μA
Ρεύμα ρυθμιστή τάσης	13	20	29	μA
Χρόνος εκκίνησης ασύρματου ταλαντωτή		580	860	μs

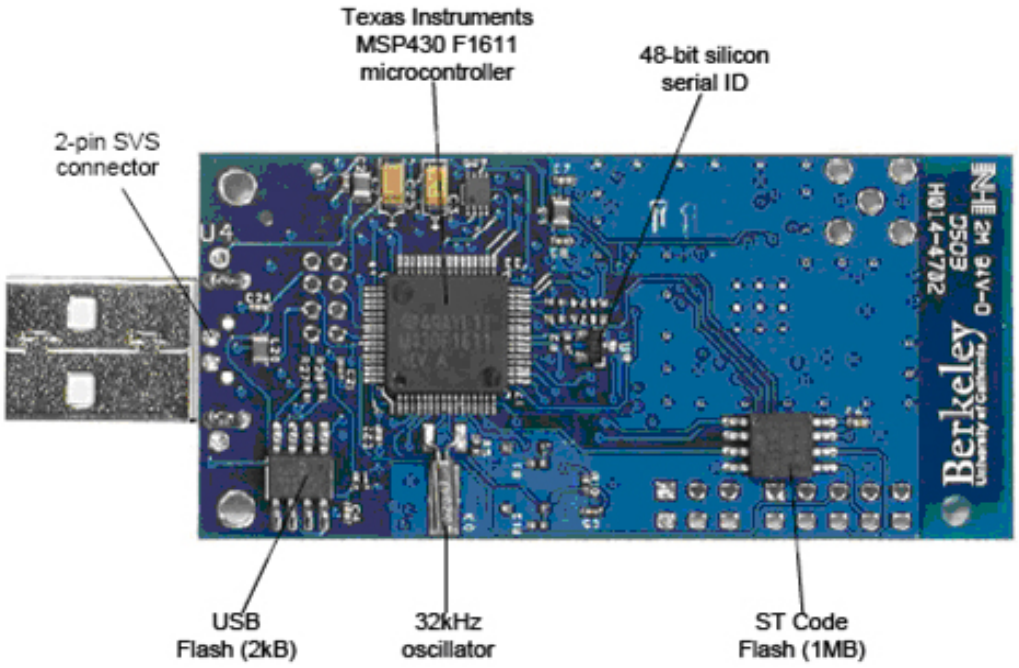
Τυπικές συνθήκες λειτουργίας ασύρματου πομποδέκτη Chipcon CC2420

Ολοκληρωμένη σχεδίαση

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



Εικόνα 34-εμπρόσθια όψη του Tmote sky.

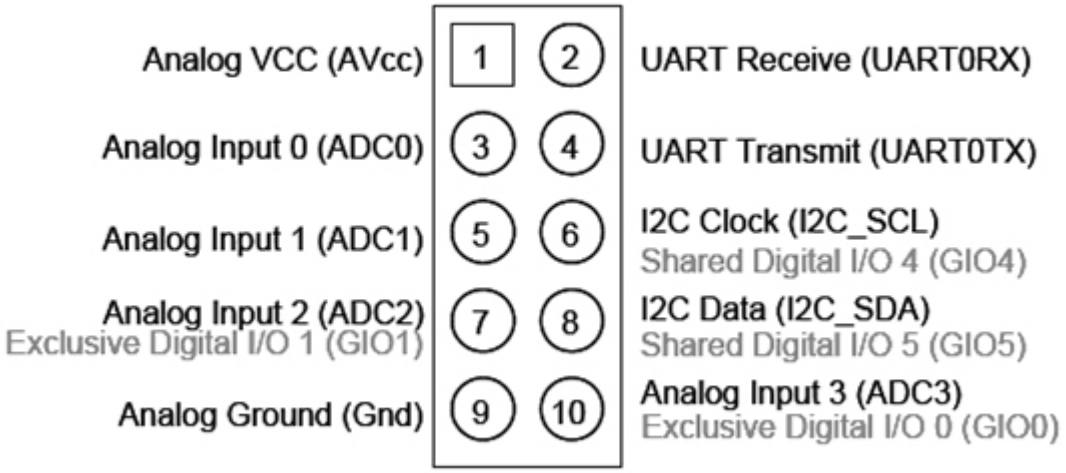


Εικόνα 35-Οπίσθια όψη του Tmote sky.

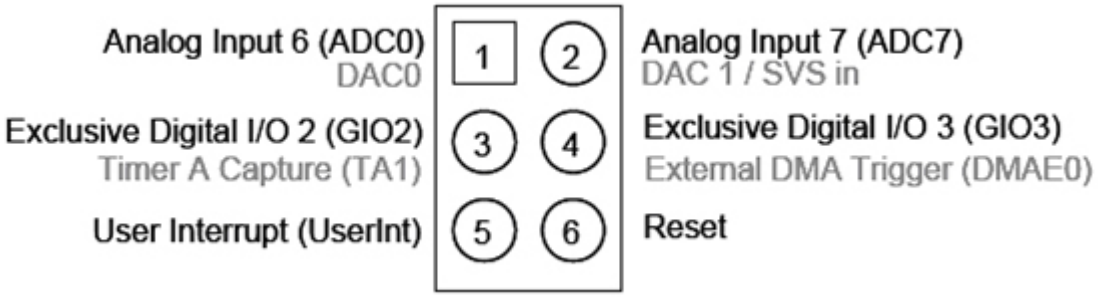
Το Tmote Sky χρησιμοποιεί μια ενσωματωμένη κεραία στα 2.4GHz , η οποία είναι μια μικροταινία σε σχήμα ανεστραμμένου F (Planar Inverted Folded Antenna – PIFA) και η οποία βρίσκεται τυπωμένη στην άκρη της πλακέτας, όπως φαίνεται και στην παραπάνω εικόνα (εμπρός όψη). Η κεραία αυτή επιτυγχάνει εμβέλεια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

Συνδετήρας επέκτασης

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

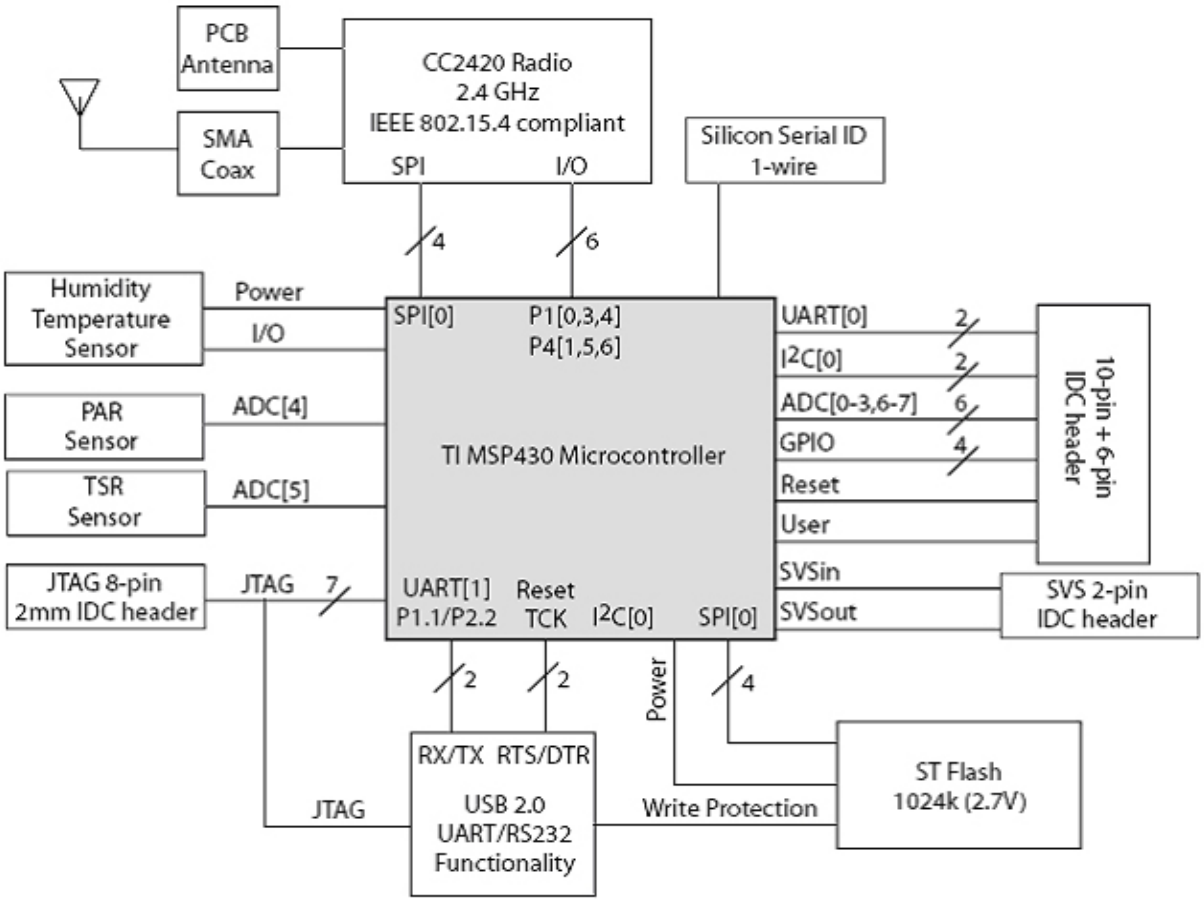


Εικόνα 36-Συνδετήρας επέκτασης 10 θέσεων(pins)



Εικόνα 37-Συνδετήρας επέκτασης 6 θέσεων (pins).

Η συσκευή λειτουργεί με δύο μπαταρίες τύπου AA οι οποίες μπορούν να χρησιμοποιηθούν για λειτουργία στο εύρος τάσης 2.1V με 3.6V DC. Εάν η συσκευή τοποθετηθεί στη θύρα USB για προγραμματισμό ή επικοινωνία με τον Η/Υ τότε μπορεί να τροφοδοτηθεί μέσω της θύρας αυτής. Στην περίπτωση αυτή η τάση τροφοδοσίας είναι 3V και δεν είναι απαραίτητη η χρήση μπαταρίας. Στην εργασία μας η μονάδα που στέλνει ασύρματα το σήμα του ηλεκτροκαρδιογράφου τροφοδοτείται με μπαταρίες, ενώ η μονάδα που επικοινωνεί με τον Η/Υ για τη λήψη, απεικόνιση και επεξεργασία των δεδομένων τροφοδοτείται μέσω της θύρας USB.



Εικόνα 38-Λειτουργικό μπλόκ διάγραμμα του Tmote.

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

Κατανάλωση ενέργειας

Η κατανάλωση ενέργειας ενός αισθητήρα δεν αφορά μόνο τον μικροελεγκτή και/ή τον πομποδέκτη, αλλά επίσης και τα βοηθητικά στοιχεία από τα οποία αποτελείται. Στον Πίνακα παρουσιάζεται η κατανάλωση ρεύματος σε διάφορες λειτουργίες της μονάδας tmote σε σύγκριση με τις πλατφόρμες Mica2 και MicaZ.

Operation	**Telos**	**Mica2**	**MicaZ**
Minimum Voltage	1.8V	2.7V	2.7V
Mote Standby (RTC o	5.1 μA	19.0 μA	27.0 μA
MCU Idle (DCO on)	54.5 μA	3.2 mA	3.2 mA
MCU Active	1.8 mA	8.0 mA	8.0 mA
MCU + Radio RX	21.8 mA	15.1 mA	23.3 mA

MCU + Radio TX (0dB	19.5 mA	25.4 mA	21.0 mA
MCU + Flash Read	4.1 mA	9.4 mA	9.4 mA
MCU + Flash Write	15.1 mA	21.6 mA	21.6 mA
MCU Wakeup	6 μs	180 μs	180 μs
Radio Wakeup	580μs	1800 μs	860 μs

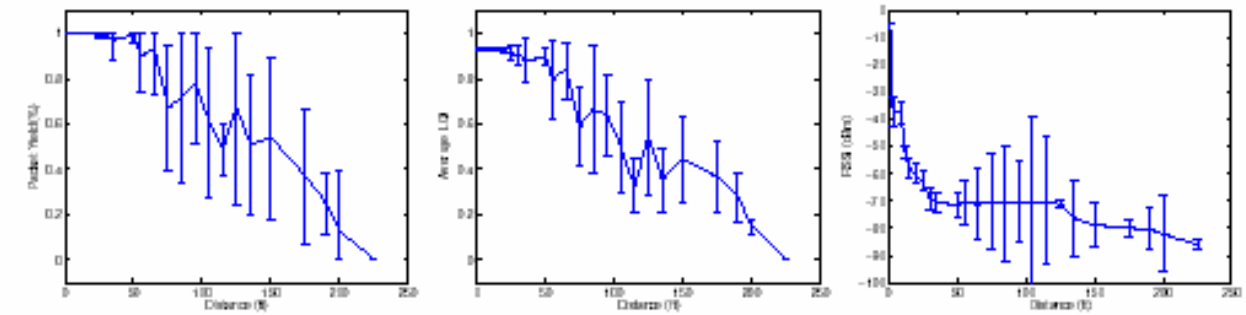
Μετρημένη κατανάλωση ρεύματος του Telos σε σύγκριση με τα Mica2 και MicaZ motes.

Το tmote εμφανίζει χαμηλότερη κατανάλωση μνήμης flash και στον μικροελεγκτή, συγκριτικά με το Mica2 (Atmel με CC1000 radio) και το MicaZ (Atmel με CC2420 radio). Λόγω της ολοκληρωμένης σχεδίασης του, ένα επιπλέον ρεύμα της τάξης των 3μA καταναλώνεται, σε κατάσταση αδράνειας, σε διακόπτες και ενδιάμεσες μνήμες για την προστασία από ροή του ρεύματος προς αποσυνδεδεμένα στοιχεία, και κυρίως το κύκλωμα της USB. Παρά το μικρό trade off, η συνολική κατανάλωση ενέργειας σε έναν κύκλο λειτουργίας (αφύπνιση, δειγματοληψία, μετάδοση και αδράνεια) είναι χαμηλότερη εκείνης των υπόλοιπων πλατφορμών. Η κατανάλωση ενέργειας ισούται με το συνολικό χρόνο ενεργής λειτουργίας της μονάδας, πολλαπλασιασμένο με το ρεύμα που καταναλώνεται σε αυτό το χρόνο. Αφού το Tmote Sky έχει χαμηλότερη κατανάλωση ρεύματος, χαμηλότερο χρόνο αφύπνισης και χαμηλότερη τάση λειτουργίας, μπορεί να επιτύχει μεγαλύτερη διάρκεια ζωής από προηγούμενους σχεδιασμούς. Με 1% duty cycle, το Telos μπορεί να διαρκέσει για σχεδόν 3 χρόνια. Συγκριτικά, η διάρκεια ζωής του Mica2 mote είναι 1.5 χρόνια και του MicaZ mote είναι 1 χρόνος. Η χαμηλότερη κατανάλωση ενέργειας δε σημαίνει ότι το Tmote Sky εμφανίζει μικρότερη λειτουργικότητα, καθώς όλο και ισχυρότερα στοιχεία μικροεπεξεργαστών ενσωματώνονται στους μικροελεγκτές.

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

χαμηλής ισχύος, αυτό που κάνει στην πραγματικότητα είναι να χαμηλώνει το duty cycle, επιτρέποντας στον πυρήνα του μικροελεγκτή να παραμένει σε κατάσταση αδράνειας για περισσότερο χρόνο και να εξυπηρετεί λιγότερες διακοπές υλικού (hardware interrupts). Την έννοια αυτών των διακοπών υλικού θα την εξηγήσουμε στο επόμενο κεφάλαιο (Κεφάλαιο 3). Η βελτίωση των επιδόσεων λόγω του DMA μας επιτρέπουν να φτάσουμε σε ρυθμό δειγματοληψίας μέχρι και 200ksamples/sec σε σύγκριση με τη μέγιστη δυνατότητα των 10ksamples/sec σε μικροελεγκτές χωρίς DMA.

Για την ασύρματη επικοινωνία, ο πομποδέκτης της μονάδας tmote, όπως και κάθε άλλος που χρησιμοποιεί το πρότυπο IEE 802.15.4 παρέχει στις εφαρμογές πληροφορίες για το μεταδιδόμενο μήνυμα. Η ενσωματωμένη στο tmote κεραία εμφανίζει κυρίως ομοιοκατευθυντικό διάγραμμα ακτινοβολίας. Από μετρήσεις που έχουν γίνει για την επίδραση της απόστασης στην ισχύ του ληφθέντος σήματος (Received Signal Strength Indicator, RSSI), στο ρυθμό επιτυχών πακέτων και στην ποιότητα της ζεύξης (Link Quality Indicator, LQI) προέκυψαν τα παρακάτω διαγράμματα για τις μέσες τιμές αυτών:



Εικόνα 39- Ποσοστό πακέτων που λαμβάνονται-δείκτης ποιότητας-ισχύς λαμβανομένου σήματος

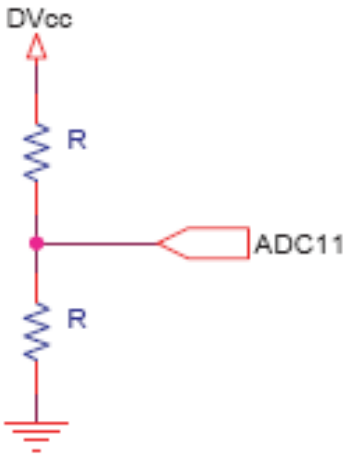
Ποσοστό ληφθέντων πακέτων (αριστερά), δείκτης ποιότητας ζεύξης (κέντρο) και ισχύς λαμβανομένου σήματος (δεξιά), σε εξωτερικό χώρο χρησιμοποιώντας τη μονάδα Tmote Sky και εσωτερική κεραία. Παρουσιάζεται ο μέσος όρος των αποτελεσμάτων για 10 συνυπάρχοντες δέκτες.

Ο δείκτης LQI καθιερώθηκε στο 802.15.4 και μετράει το σφάλμα στην ενδοδιαμόρφωση των επιτυχώς ληφθέντων πακέτων (πακέτα που πέρασαν τον CRC έλεγχο).Ο LQI του πομποδέκτη πλησιάζει σχηματικά το ρυθμό επιτυχών πακέτων. Ο RSSI ακολουθεί εκθετική μείωση καθώς ο ρυθμός επιτυχών πακέτων είναι υψηλός. Μετά από 18,29m (60 feet), το σήμα είναι πιο θορυβώδες και

μειώνεται στην ελάχιστη ευαισθησία του πομποδέκτη. Επίσης, σε πειράματα που έγιναν σε ένα δίκτυο αποτελούμενο από τριάντα μονάδες-κόμβους Tmote Sky ώστε να μετρηθεί το πραγματικό εύρος ζώνης, προέκυψε ότι μια μονάδα (mote) είναι ικανή να χρησιμοποιήσει σχεδόν το μισό ενός πλήρους εύρους ζώνης δεδομένων του καναλιού ή 125kbps. Όταν και οι 30 κόμβοι μεταδίδουν όσο γρηγορότερα γίνεται, το Tmote Sky περιορίζεται σε ένα μέσο ρυθμό λήψης των 150kbps. Η απόδοση βέβαια, όπως είπαμε μπορεί να αυξηθεί χρησιμοποιώντας τον ελεγκτή DMA για την απευθείας μετάδοση δεδομένων χωρίς την μεσολάβηση της MCU καθώς και τη μείωση των συμβάντων διακοπής υλικού και υπερχείλισης της ενδιάμεσης μνήμης.

Αισθητήρες υγρασίας/θερμοκρασίας και φωτός

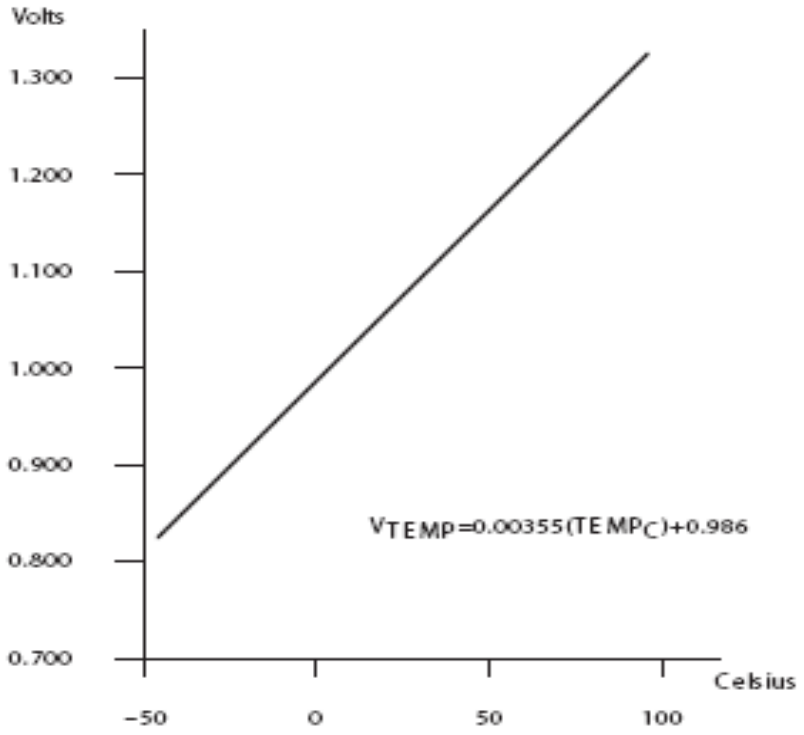
Πάνω στην πλακέτα του sky mote βρίσκονται ενσωματωμένοι αισθητήρες υγρασίας/θερμοκρασίας και φωτός, οι οποίοι μπορούν να χρησιμοποιηθούν σε πλήθος εφαρμογών. Ο αισθητήρας υγρασίας/θερμοκρασίας, κατασκευασμένος από την εταιρεία Sensirion AG, παράγεται χρησιμοποιώντας μια CMOS επεξεργασία και συνδυάζεται με έναν 14bit αναλογικό/ψηφιακό μετατροπέα (A/D converter). Ο αισθητήρας φωτός χρησιμοποιεί φωτοδιόδους, στη συγκεκριμένη περίπτωση κατασκευασμένους από την Hamamatsu Corporation, οι οποίοι ‘αντιδρούν’ στην ακτινοβολία φωτός. Τέλος, ο μικροελεγκτής MPS430 διαθέτει και εσωτερικούς αισθητήρες θερμοκρασίας και τάσης οι οποίοι μπορούν να χρησιμοποιηθούν μέσω της διεπαφής ADC του μικροελεγκτή. Η θύρα τάσης (είσοδος 11) στον 12-bit ADC καταγράφει την έξοδο από έναν διαχωριστή τάσης.



Εικόνα 40-Διάγραμμα του κυκλώματος αισθητήρα θερμοκρασίας.

Η μετατροπή των μονάδων ADC που καταγράφονται, σε μονάδες τάσης γίνεται με βάση την παρακάτω σχέση:

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



Εικόνα 41-Παράστασης τυπικής απόκλισης του εσωτερικού αισθητήρα φωτός.

Κεφάλαιο 3

Το λογισμικό (Software) των WSN

Τα πρώτα ασύρματα ενσωματωμένα συστήματα αισθητήρων έτρεχαν πάνω σε προσωπικούς υπολογιστές και χρησιμοποιούσαν κυρίως προγράμματα Linux. Όταν η ανάπτυξη αυτών των δικτύων πέρασε από τους μικροεπεξεργαστές (microprocessors) στους μικροελεγκτές (microcontrollers), το Linux είχε πάψει πια να αποτελεί την κατάλληλη επιλογή. Οι εφαρμογές των συστημάτων της εποχής εκείνης αναπτύσσονταν κυρίως σε τυπική γλώσσα C ή κατευθείαν σε γλώσσα assembly. Ο προγραμματισμός όμως σε αυτή τη γλώσσα είναι δύσκολο να αναλυθεί και επίσης μπορεί εύκολα να καταλήξει εκτός ελέγχου όταν η πολυπλοκότητα της εφαρμογής αυξηθεί. Σε κλιμακωτά συστήματα το πρόβλημα μπορεί να επιληφθεί με αντικειμενοστραφή προγραμματισμό (object-oriented programming), ο οποίος καθιστά ευκολότερο το διαχωρισμό πολύπλοκων προγραμμάτων σε ανεξάρτητα, ευκολοσύνθετα στοιχεία. Αλλά ο προγραμματισμός με προσανατολισμό στο αντικείμενο απαιτεί δυναμική παραχώρηση μνήμης και τείνει να απαιτεί περισσότερους προγραμματιστικούς πόρους, κάτι το οποίο τον κρίνει ακατάλληλο για ενσωματωμένα συστήματα (embedded systems).

Η γλώσσα NesC, η οποία αναπτύχθηκε από ερευνητές του Πανεπιστημίου UC Berkeley, αντιπροσωπεύει ένα νέο πολλά υποσχόμενο πεδίο για τους σχεδιαστές εφαρμογών. Είναι κατάλληλα σχεδιασμένη για ενσωματωμένα συστήματα δικτύων και υποστηρίζει ένα προγραμματιστικό μοντέλο που ενσωματώνει αντιδραστικότητα με το περιβάλλον, ταυτοχρονισμό και δυνατότητα επικοινωνίας.

Ένα βασικός άξονας επικέντρωσης της NesC είναι η ο ολιστικός σχεδιασμός συστημάτων. Οι εφαρμογές των μονάδων-αισθητήρων (**motes**) είναι βαθιά συνδεδεμένες στο υλικό και κάθε μονάδα τρέχει μια εφαρμογή κάθε φορά.

Αυτή η προσέγγιση αποφέρει τρεις σημαντικές ιδιότητες. Πρώτον, όλοι οι πόροι θεωρούνται στατικοί. Δεύτερον, αντί της χρησιμοποίησης ενός γενικής εξυπηρέτησης λειτουργικού συστήματος, οι εφαρμογές κατασκευάζονται από ένα

σύνολο στοιχείων συστήματος συσχετισμένων με συγκεκριμένο κώδικα. Τρίτον, τα ‘όρια-σύνορα’ υλικού/λογισμικού εξαρτώνται από την εφαρμογή και την πλατφόρμα υλικού που χρησιμοποιείται και είναι σημαντικό να σχεδιάζονται για ευέλικτη αποδόμηση.

Υπάρχει ένας αριθμός μοναδικών προκλήσεων που η γλώσσα NesC πρέπει να επιληφθεί:

Οδήγηση από την αλληλεπίδραση με το περιβάλλον

Σε αντίθεση με τα παραδοσιακά συστήματα υπολογιστών, τα motes χρησιμοποιούνται για τη συλλογή δεδομένων και τον έλεγχο του τοπικού περιβάλλοντος, παρά για γενικής φύσεως υπολογισμούς. Αυτή η ιδιαιτερότητα οδηγεί σε δυο παρατηρήσεις:

Η πρώτη είναι ότι τα motes είναι στοιχειωδώς οδηγούμενα από συμβάντα (event driven), αντιδρώντας σε αλλαγές του περιβάλλοντος (άφιξη ενός μηνύματος, επίκτηση δεδομένων από αισθητήρες) παρά οδηγούμενα από διαδραστική *(interactive)* ή κατά δεσμίδες *(batch)* επεξεργασία.

Η δεύτερη παρατήρηση είναι ότι η ‘άφιξη’ ενός συμβάντος ή η επεξεργασία δεδομένων είναι συντρέχουσες δραστηριότητες, απαιτώντας έτσι μια μεθόδευση για διαχείριση του ταυτοχρονισμού αυτού που επιλαμβάνεται ενδεχόμενων σφαλμάτων *(bugs)* όπως οι συνθήκες συναγωνισμού (race conditions).

Περιορισμένοι πόροι

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

Αξιοπιστία

Αν και είναι αναμενόμενο οι μονάδες αυτές να παθαίνουν βλάβη λόγω σφαλμάτων υλικού, είναι έντονη η ανάγκη για εφαρμογές οι οποίες μπορεί να τρέχουν για μεγάλο χρονικό διάστημα. Για παράδειγμα, οι εφαρμογές παρακολούθησης περιβάλλοντος πρέπει να είναι ικανές να συλλέγουν δεδομένα χωρίς την ανθρώπινη παρέμβαση για μήνες κάθε φορά. Ένας σημαντικός στόχος

είναι η μείωση των σφαλμάτων κατά τη διάρκεια της εκτέλεσης (run-time errors), καθώς δεν υπάρχει ουσιαστικός μηχανισμός ανάκαμψης σφαλμάτων εκτός από την αυτόματη επανεκκίνηση του συστήματος.

Μικρές απαιτήσεις για λειτουργίες πραγματικού χρόνου

Παρόλο που υπάρχουν κάποιες εργασίες που είναι χρονικά κρίσιμες, όπως η διαχείριση της ασύρματης επικοινωνίας ή το calibration των αισθητήρων, σε γενικές γραμμές δεν υπάρχουν μεγάλες απαιτήσεις για πραγματικού χρόνου λειτουργίες. Μάλιστα η εμπειρία έχει δείξει ότι οι όποιοι χρονικοί περιορισμοί μπορούν να ικανοποιηθούν έχοντας απόλυτο έλεγχο της εφαρμογής κα του λειτουργικού συστήματος και παράλληλα μειώνοντας την χρησιμοποίηση (utilization).

Μια από τις λίγες κρίσιμες από πλευράς χρόνου λειτουργίες στα δίκτυα αισθητήρων είναι η ασύρματη επικοινωνία. Δεδομένου όμως της βασικής αναξιοπιστίας της ραδιοζεύξης γενικότερα δε θα χρειαστεί απαραίτητα να ικανοποιήσουμε δύσκολες απαιτήσεις στον τομέα αυτό.

Παρόλο που η γλώσσα προγραμματισμού NesC είναι μια σύνθεση από πολλές υπάρχουσες γλώσσες οι οποίες εστιάζονται στα παραπάνω προβλήματα, εντούτοις παρέχει τρία σημαντικά στοιχεία:

- Η γλώσσα NesC ορίζει ένα μοντέλο στοιχείων που υποστηρίζει συστήματα ηγούμενα από συμβάντα. Το μοντέλο αυτό παρέχει αμφίδρομες διεπαφές (interfaces) προς απλοποίηση της ροής των συμβάντων και επιτρέπει αποδοτική και ελαφριά υλοποίηση χωρίς τη δημιουργία εικονικών συναρτήσεων και δυναμικών στοιχείων.
- Παράλληλα ορίζει ένα απλό αλλά συγκεκριμένο μοντέλο ταυτοχρονισμού σε συνδυασμό με εκτεταμένη ανάλυση κατά τη μεταγλώττιση: ο μεταγλωττιστής (compiler**)** της NesC εντοπίζει τις πλειονότητα των περιπτώσεων ανταγωνισμού δεδομένων (data race) κατά τη διάρκεια της μεταγλώττισης. Αυτός ο συνδυασμός επιτρέπει τη δημιουργία σύγχρονων εφαρμογών που απαιτούν περιορισμένους πόρους.
- Τέλος, η γλώσσα NesC παρέχει μια μοναδική ισορροπία μεταξύ της ανάλυσης προγράμματος, για τη βελτίωση της αξιοπιστίας και τη μείωση του κώδικα, και της δυνατότητας για δημιουργία ολοκληρωμένων εφαρμογών.

Επειδή η γλώσσα NesC έχει αποδειχθεί ότι είναι αποτελεσματική στην περίπτωση ανάπτυξης εφαρμογών για ασύρματα δίκτυα αισθητήρων,

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

Λειτουργικό σύστημα TinyOS

Το **TinyOS** που αναπτύχθηκε και εξελίχθηκε από το πανεπιστήμιο του Berkley είναι ένα μικρό σε μέγεθος, ανοικτού κώδικα ενεργειακά οικονομικό στη διαχείριση των αισθητήρων λειτουργικό σύστημα που τρέχει και διαχειρίζεται κάθε κόμβο του δικτύου. Παρέχει ένα σύνολο από δομικές μονάδες λογισμικού από τις οποίες ο προγραμματιστής μπορεί να διαλέξει τα κατάλληλα στοιχεία (components). Το μέγεθος τέτοιων αρχείων είναι τις τάξης των 200 bytes έτσι το μέγεθος του συνολικού προγράμματος παραμένει το ελάχιστο δυνατό. Το λειτουργικό σύστημα αυτό διαχειρίζεται τόσο το υλικό όσο και το ασύρματο δίκτυο εκτελώντας τις μετρήσεις των αισθητήρων παίρνοντας αποφάσεις δρομολόγησης και ελέγχοντας την κατανάλωση ενέργειας.

Παρά τα πολλά λειτουργικά συστήματα που κατά καιρούς προτάθηκαν για τη διαχείριση των WSN , το **TinyOS** επικράτησε με μεγάλη διαφορά κυρίων λόγω του ελευθέρου κώδικα που μπορεί να βρει κα να χρησιμοποιήσει ο οποιοσδήποτε, της πληθώρας των υλοποιήσεων που αναπτύχθηκαν και της γενικότερης δημοφιλίας του. Λόγω των περιορισμένων πόρων μια νέα γλώσσα προγραμματισμού αναπτύχθηκε, η nesC που υλοποιεί τις δομικές σχεδιαστικές ανάγκες και την επαναχρησιμοποίηση κώδικα του **TinyOS** για μικροσκοπικούς αισθητήρες. Για την υλοποίηση της επαναχρησιμοποίησης κώδικα το **TinyOS** εφαρμόζει μια αρχιτεκτονική κατανομής σε επιμέρους στοιχεία(component – based). Επιπρόσθετα για την βελτιστοποίηση της διαχείρισης ενέργειας χρησιμοποιεί ένα μοντέλο εκτέλεσης βασισμένο σε γεγονότα(event – based) όπου τα γεγονότα οδηγούν τα προγράμματα και οι σχετικοί πόροι αποδεσμεύονται με το πέρας της χρήσης τους.

Μια εύχρηστη αφαιρούμενη διαστρωμάτωση σε επίπεδο υλικού στον πυρήνα του **TinyOS** πραγματοποιεί την εύκολη προσαρμογή του σε διάφορα είδη

πλατφορμών .Η διαστρωμάτωση αυτή επίσης διευκολύνει πολύ την ανάπτυξη των ασύρματων δικτύων . Το **TinyOS** ακόμα προσφέρει μια σειρά από εφαρμογές και εργαλεία ανάπτυξης όπως το Tossim (προσομοιωτής δικτύων του **TinyOS**) ,το deluge και το TinyDB που βοηθούν στην ανάπτυξη και ερεύνα των WSN. Λόγω της αποδοτικής σχεδίασης, της μεγάλης κοινότητας υποστήριξης και του ανοικτού κώδικα το **TinyOS** έγινε το πλέον διαδεδομένο λειτουργικό σύστημα για τα WSN.

Η γλώσσα NesC και βασικά της χαρακτηριστικά

Όπως αναφέραμε και προηγουμένως, η γλώσσα NesC είναι μια γλώσσα προγραμματισμού προορισμένη για ενσωματωμένα συστήματα (embedded systems), τα οποία αποτελούν ένα νέο πολλά υποσχόμενο χώρο για σχεδιαστές εφαρμογών. Ένα παράδειγμα τέτοιου συστήματος είναι και τα δίκτυα αισθητήρων. Η NesC έχει σύνταξη όμοια με τη γλώσσα C, αλλά υποστηρίζει το μοντέλο ταυτοχρονισμού του **TinyOS** καθώς και μηχανισμούς διάρθρωσης, ονοματοδοσίας και διασύνδεσης στοιχείων λογισμικού σε πλήρως λειτουργικά ενσωματωμένα συστήματα δικτύων. Η βασικότερη επιδίωξη της γλώσσας NesC είναι να επιτρέψει στους σχεδιαστές εφαρμογών να κατασκευάσουν στοιχεία που είναι εύκολο να οδηγήσουν σε ολοκληρωμένα, σύγχρονα συστήματα, και ακόμη να εκτελεί εκτεταμένους ελέγχους κατά τη διάρκεια του μεταγλωττισμού.

Οι εφαρμογές γραμμένες σε γλώσσα NesC είναι κατασκευασμένες από στοιχεία (components) με σαφώς προσδιορισμένες αμφίδρομες διεπαφές (interfaces). Επίσης, η γλώσσα NesC προσδιορίζει, όπως αναφέραμε και προηγουμένως, ένα μοντέλο ταυτοχρονισμού βασισμένο σε εργασίες (tasks) και χειριστές διακοπής υλικού (hardware interrupt handlers**)** και ανιχνεύει ανταγωνισμούς δεδομένων κατά τη διάρκεια της μεταγλώττισης.
Μερικά βασικά χαρακτηριστικά της γλώσσας NesC είναι:

- Η γλώσσα NesC είναι προέκταση της C
- Συνολική προγραμματιστική ανάλυση:.
- Η NesC είναι μια ‘στατική’ γλώσσα**:** Δεν υπάρχει δυναμική παραχώρηση μνήμης και το διάγραμμα κλήσεων είναι πλήρως γνωστό κατά τη διάρκεια της μεταγλώττισης
- Η NesC υποστηρίζει και αντανακλά το σχεδιασμό του λειτουργικού **TinyOS**.

Συστατικά και διεπαφές(Components and interfaces)

Η nesC είναι μια γλώσσα βασισμένη στα συστατικά (components). Τα συστατικά nesC χρησιμοποιούν ένα καθαρώς τοπικό namespace. Αυτό σημαίνει αυτό εκτός από δηλώνοντας τις λειτουργίες που εφαρμόζει, ένα συστατικό πρέπει επίσης να δηλώσει τις λειτουργίες που καλεί.Τα ονόματα που ένα συστατικό χρησιμοποιεί για να καλέσει αυτές τις λειτουργίες είναι απολύτως τοπικά: το όνομα στο οποίο αναφέρετε δεν είναι απαραίτητα το ίδιο με αυτό που εφαρμόζει τη λειτουργία. Όταν ένα συστατικό A δηλώνει ότι καλεί μια λειτουργία B, εισάγει ουσιαστικά το όνομα A.B σε ένα σφαιρικό namespace. Ένα διαφορετικό συστατικό, Γ, το οποίο καλεί το η λειτουργία B εισάγει C.B στο σφαιρικό namespace. Ακόμα κι αν και το A και το γ αναφέρονται στη λειτουργία B. μπορεί να αναφέρονται σε απολύτως διαφορετικές εφαρμογές.

Κάθε συστατικό έχει μια προδιαγραφή, ένας κομμάτι κώδικα που δηλώνει τις λειτουργίες που παρέχει(implements) και τις λειτουργίες που χρησιμοποιεί (calls). Παραδείγματος χάριν, αυτή είναι μια προδιαγραφή για ένα πλασματικό συστατικό SmoothingFilterC, που επεξεργάζεται ακατέργαστα στοιχεία:

```
module SmoothingFilterC {
  provides command uint8_t topRead(uint8_t* array, uint8_t len);
  uses command uint8_t bottomRead(uint8_t* array, uint8_t len);
}
```

Listing 3.11: SmoothingFilterC, a simple nesC module

Επειδή το SmoothingFilterC παρέχει τη λειτουργία topRead, πρέπει να την καθορίσει και άλλα συστατικά μπορούν να την καλέσουν. Αντιθέτως, επειδή το SmoothingFilterC χρησιμοποιεί bottomRead, μπορεί να παραπέμψει τη λειτουργία και εξαρτάται έτσι σε κάποιο άλλο συστατικό για να το καθορίσει. Τα συστατικά μπορούν πάντα να παραπέμψουν τις λειτουργίες που καθορίζουν: Το SmoothingFilterC μπορεί να καλέσει το topRead .

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. Παραδείγματος χάριν, η διαχείριση ενέργειας και τα ζητήματα αξιολόγησης σημαίνουν ότι οι εφαρμογές χρειάζεται συχνά να είναι

σε θέση να αρχίσουν και να σταματήσουν τις αφαιρέσεις και τις υπηρεσίες συστημάτων, όπως να ανοίξουν έναν αισθητήρα για να πάρει μια μέτρηση ή ανοίγοντας τη στοίβα σωρό για να ακούσει τα πακέτα. Η διεπαφή StdControl είναι ένας κοινός τρόπος να εκφραστεί αυτό λειτουργία:

```
interface StdControl {
  command error_t start();
  command error_t stop();
}
```

Listing 3.12: The StdControl interface

Ένα συστατικό που αντιπροσωπεύει μια αφαίρεση ή μια υπηρεσία που μπορεί να αρχίσει ή να σταματήσει παρέχει StdControl, ενώ ένα συστατικό που πρέπει να ανοίξει άλλα συστατικά χρησιμοποιεί StdControl. Αυτό είναι συχνά μια ιεραρχική σχέση. Παραδείγματος χάριν, ένα στρώμα δρομολόγησης πρέπει να ανοίξει ένα στρώμα πακέτων συνδέσεων στοιχείων, το οποίο πρέπει στη συνέχεια να ανοίξει και να σταματήσει ανίχνευση καναλιών στάσεων:

```
module RoutingLayerC {
  provides interface StdControl;
  uses interface StdControl as SubControl;
}

module PacketLayerC {
  provides interface StdControl;
}
```

Listing 3.13: Interfaces in component signatures

Η σύνδεση των προμηθευτών και των χρηστών από κοινού καλείται καλωδίωση (wiring). Παραδείγματος χάριν, ο κώδικας RoutingLayerC έχει λειτουργία κλήσεις στο SubControl.start () και SubControl.stop (). Εκτός αν το SubControl συνδέεται με καλώδιο σε έναν προμηθευτή, αυτές οι λειτουργίες είναι απροσδιόριστα σύμβολα: δεν είναι συνδεδεμένες σε οποιοδήποτε πραγματικό κώδικα. Εντούτοις, εάν το SubControl συνδέεται με καλώδιο στο PacketLayerC του StdControl, έπειτα όταν το RoutingLayerC καλεί το SubControl.start (), θα επικαλεστεί το PacketLayerC του StdControl.start (). Αυτό σημαίνει ότι η αναφορά RoutingLayerC.SubControl.start δείχνει προς τον καθορισμό του PacketLayerC.StdControl.start. Τα δύο συστατικά RoutingLayerC και PacketLayerC

είναι εντελώς αποσυνδεδεμένα, και συνδέονται μόνο με καλώδιο. Αυτό γίνετε κατ' αρχάς, γιατί ο κώδικας χωρίζεται στα συστατικά, ιδιαίτερες μονάδες της λειτουργίας.

Ένα συστατικό μπορεί μόνο να παραπέμψει μεταβλητές από το δικό του τοπικό namespace. Ένα συστατικό δεν μπορεί να ονομάζει μεταβλητές σε ένα άλλο συστατικό. Εντούτοις, ένα συστατικό μπορεί να δηλώσει ότι χρησιμοποιεί μια λειτουργία που καθορίζεται από άλλο συστατικό. Μπορεί επίσης να δηλώσει ότι παρέχει μια λειτουργία που ένα άλλο συστατικό μπορεί να καλέσει. Υλοποίηση ενός nesC προγράμματος περιλαμβάνει τα τμήματα γραψίματος συνδέοντας με καλώδιο τους χρήστες στους προμηθευτές. Επειδή αυτή η σύνθεση εμφανίζεται κατά τη μεταγλώττιση δεν απαιτεί κατανομή χρόνου εκτέλεσης ή αποθήκευση των δεικτών λειτουργίας στο RAM. Επιπλέον, αφού ένα πρόγραμμα δεν έχει αυτά τα επίπεδα indirection, ο μεταγλωττιστής nesC ξέρει την πλήρη γραφική παράσταση κλήσης. Πριν συνδέσει με καλώδιο, ένα συστατικό θα μπορούσε να καλέσει οποιοδήποτε άλλο συστατικό, και έτσι αποσυνδέεται εντελώς από αυτό που καλεί.. Εντούτοις, σε μια εφαρμογή, η κλήση συνδέεται με καλώδιο σε ένα συγκεκριμένο σημείο τέλους, και έτσι ο μεταγλωττιστής nesC μπορεί να βελτιστοποιηθεί πέρα από το όριο κλήσης . **TinyOS** και nesC μπορούν να υιοθετήσουν αυτήν την μέθοδο επειδή, αντίθετα από τους υπολογιστές τελικών χρηστών, που έχουν ανάγκη να είναι σε θέση να φορτώσουν δυναμικά τα νέα προγράμματα, τα δίκτυα αισθητήρων αποτελούνται από ενσωματωμένους υπολογιστές, οι οποίοι έχουν καθορισμένες με σαφήνεια και στενά διευκρινισμένες χρήσεις. Ενώ αυτές μπορούν να εξελιχθούν κατά τη διάρκεια του χρόνου, η εξέλιξη είναι πολύ αργή σε σύγκριση με πόσο συχνά ένα PC φορτώνει τα νέα προγράμματα.

Η λειτουργία split-face

Επειδή οι κόμβοι αισθητήρων έχουν ένα μεγάλο εύρος ικανοτήτων υλικού, ένας από τους στόχους του**TinyOS** είναι να υπάρξει ένα εύκαμπτο όριο υλικού/λογισμικού. Μια εφαρμογή που κρυπτογραφεί τα πακέτα πρέπει να είναι σε θέση να χρησιμοποιεί εναλλακτικές εφαρμογές υλικού ή λογισμικού χρήσης. Το υλικό, εντούτοις, είναι σχεδόν πάντα split-face παρά Blocking . Είναι split-face σε εκείνη την ολοκλήρωση ενός αιτήματος που είναι μια επανάκληση. Παραδείγματος χάριν, για να χρησιμοποιήσει έναν αισθητήρα διαβάζοντας με έναν αναλογικό σε ψηφιακό μετατροπέα (ADC), το λογισμικό γράφει σε μερικούς καταλόγους

διαμόρφωσης στην έναρξη ένα δείγμα. Όταν το δείγμα του ADC ολοκληρωθεί, τα ζητήματα υλικού διακόπτουν, και το λογισμικό διαβάζει την τιμή από έναν κατάλογο στοιχείων.

Το **TinyOS** επομένως υιοθετεί την εξής μέθοδο. Παρά να τα καταστήσει όλα σύγχρονα κατευθείαν με νήματα, διαδικασίες που είναι split-face στο υλικό είναι split-face και στο λογισμικό επίσης. Αυτό σημαίνει πολλές κοινές διαδικασίες, όπως η δειγματοληψία των αισθητήρων και η αποστολή των πακέτων, είναι split-face. Ένα σημαντικό χαρακτηριστικό των διεπαφών διάσπαση-φάσης είναι ότι είναι αμφίδρομες: υπάρχει ένα downcall για να αρχίσει τη λειτουργία, και ένα upcall που δηλώνει τη λειτουργία είναι πλήρες. Σε nesC, downcalls είναι γενικά εντολές, ενώ upcalls είναι γεγονότα. Μια διεπαφή διευκρινίζει και τις δύο πλευρές αυτής της σχέσης. Παραδείγματος χάριν, αυτό είναι η βασική διεπαφή **TinyOS** αποστολής πακέτων, Send:

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

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

Listing 4.1: The split–phase Send interface

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

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

Tasks

Ο σωστός τρόπος να γίνει αυτό είναι με έναν task, μια αναβεβλημένη κλήση διαδικασίας. Μια ενότητα μπορεί να ταχυδρομήσει έναν task στον **TinyOS** χρονοπρογραμματιστής. Σε κάποιο σημείο αργότερα, ο χρονοπρογραμματιστής θα

εκτελέσει το task.Επειδή το task δεν καλείτε αμέσως, δεν υπάρχει καμία επιστροφής αξία. Επίσης, επειδή ένα task εκτελεί στο ονομαζόμενο πλαίσιο ενός συστατικού, δεν παίρνει οποιεσδήποτε παραμέτρους: οποιαδήποτε παράμετρο θέλετε να περάσετε μπορείτε να την αποθηκεύετε στο συστατικό. Taskς, όπως οι λειτουργίες, μπορούν να δηλωθούν εκ των προτέρων μια δήλωση για έναν στόχο μοιάζει με:

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

Listing 4.13: Declaring a task

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

```
post readDoneTask();
```

Listing 4.14: Posting a task

Έτσι το τμήμα φίλτρων στοιχείων μας μοιάζει με εφαρμοσμένο task:

```
module FilterMagC {
  provides interface StdControl;
  provides interface Read<uint16_t>;
  uses interface Timer<TMilli>;
  uses interface Read<uint16_t> as RawRead;
}
module PeriodicReaderC {
  provides interface StdControl;
  uses interface Timer<TMilli>;
  uses interface Read<uint16_t>;
}
implementation {
  uint16_t filterVal = 0;
  uint16_t lastVal = 0;

  task void readDoneTask();

  command error_t StdControl.start() {
    return call Timer.startPeriodic(10);
  }
  command error_t StdControl.stop() {
    return call Timer.stop();
  }
  event void Timer.fired() {
    call RawRead.read();
  }
  event void RawRead.readDone(error_t err, uint16_t val) {
    if (err == SUCCESS) {
      lastVal = val;
      filterVal *=  9;
      filterVal /= 10;
      filterVal += lastVal / 10;
    }
  }

  command error_t Read.read() {
    post readDoneTask();
    return SUCCESS;
  }

  task void readDoneTask() {
    signal Read.readDone(SUCCESS, filterVal);
  }
```

Όταν FilterMagC.Read.read καλείται, το FilterMagC σταματά το treadDoneTask και επιστρέφει αμέσως. Κάποια στιγμή αργότερα το **TinyOS** τρέχει το task το οποίο σηματοδοτεί το Read.readDone.

Τα tasks είναι non-preemptive. Αυτό σημαίνει ότι μόνο ένα task τρέχει οποιαδήποτε στιγμή, και το **TinyOS** δεν διακόπτει ένα task για να τρέξει άλλο/ς. Μόλις αρχίσει ένα task, κανένα άλλο task δεν τρέχει έως ότου ολοκληρωθεί. Αυτό

σημαίνει αυτό τα tasks που οργανώνονται ατομικά όσον αφορά το ένα το άλλο. Εάν ένα συστατικό έχει έναν πολύ μακροχρόνιο υπολογισμό που κάνει, πρέπει να τον σπάσει σε πολλαπλά tasks. Ένα task μπορεί να ποστάρει τον εαυτό του. Παραδείγματος χάριν, στο βασικός βρόχος εκτέλεσης του Mate bytecode ο διερμηνέας είναι ένα task που εκτελεί μερικές οδηγίες ενός νήματος και reposts τον εαυτό του.

Διεπαφές με επιχειρήματα (interfaces and arguments)

Οι διεπαφές μπορούν να πάρουν τους τύπους ως επιχειρήματα. Παραδείγματος χάριν, είναι η Read είναι μια απλή διεπαφή για την ανάγνωση αισθητήρων:

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

Listing 4.2: The Read interface

Τα επιχειρήματα τύπων στις διεπαφές εγκλείονται στις αγκύλες . Η διεπαφή read έχει ένα ενιαίο επιχείρημα, το οποίο καθορίζει τον τύπο της αξίας στοιχείων που παράγει. Παραδείγματος χάριν, ένα τμήμα μαγνητόμετρων που παράγει μια δεκαεξάμπιτη ανάγνωση μοιάσει με αυτό:

```
module MagnetometerC {
  provides interface StdControl;
  provides interface Read<uint16_t>;
}
```

Listing 4.3: Using the Read interface in MagnetometerC

Κατά καλωδίωση των προμηθευτών και των χρηστών των διεπαφών που έχουν τύπους επιχειρήματα, οι τύποι πρέπει να ταιριάζουν. Για παράδειγμα, δεν μπορεί να συνδέσει με καλώδιο τη Read <uint8 τ> με τη Read<uint16 τ>. Μερικές φορές, τα επιχειρήματα χρησιμοποιούνται για να επιβάλουν τον τύπο ελέγχοντας που δεν αναφέρεται πραγματικά στα επιχειρήματα στις εντολές ή τα γεγονότα. Παραδείγματος χάριν, η διεπαφή Timer παίρνει μια ενιαία παράμετρο που δεν είναι σε οποιεσδήποτε από τις λειτουργίες της:

```
interface Timer<precision_tag> {
  command void startPeriodic(uint32_t dt);
  command void startOneShot(uint32_t dt);
  command void stop();
  event void fired();

  command bool isRunning();
  command bool isOneShot();

  command void startPeriodicAt(uint32_t t0, uint32_t dt);
  command void startOneShotAt(uint32_t t0, uint32_t dt);
  command uint32_t getNow();
  command uint32_t gett0();
  command uint32_t getdt();
}
```

Listing 4.4: The Timer interface and its typing

Υλοποίηση Ενοτήτων (Module Implementation)

Κάθε συστατικό έχει μια δομή εφαρμογής μετά από την υπογραφή του. Για τις ενότητες, αυτή η εφαρμογή είναι παρόμοια με ένα αντικείμενο: έχει μεταβλητές και λειτουργίες. Μια ενότητα πρέπει να εφαρμόσει κάθε εντολή των διεπαφών που παρέχει και κάθε γεγονός των διεπαφών που χρησιμοποιεί. Παραδείγματος χάριν, αυτό είναι μια απλή πιθανή εφαρμογή
PeriodicReaderC:

```
module PeriodicReaderC {
  provides interface StdControl;
  uses interface Timer<TMilli>;
  uses interface Read<uint16_t>;
}
implementation {
  uint16_t lastVal = 0;

  command error_t StdControl.start() {
    return call Timer.startPeriodic(1024);
  }
  command error_t StdControl.stop() {
    return call Timer.stop();
  }
  event void Timer.fired() {
    call Read.read();
  }
  event void Read.readDone(error_t err, uint16_t val) {
    if (err == SUCCESS) {
      lastVal = val;
    }
  }
}
```

Listing 4.9: Complete PeriodicReaderC component

Αυτό το συστατικό λαμβάνει δείγματα περιοδικά από τους αισθητήρες και αποθηκεύει την τελευταία έγκυρη ανάγνωση σε μια τοπική μεταβλητή. Σημείωτέον πώς το επιτυγχάνει αυτό με την διάσπαση-φάση. Η κλήση σε StdControl.start θα αρχίσει το χρονόμετρο. Ένα δευτερόλεπτο αργότερα, το Timer.fired επισημαίνεται, το συστατικό καλεί το Read.read και επιστρέφει. Σε κάποιο σημείο αργότερα, ανάλογα με τη λανθάνουσα κατάσταση της λειτουργίας Read, τα σήματα Read.readDone πηγής στοιχείων, περνούν την ανάγνωση ως επιχείρημα.

Διαμορφώσεις και Καλωδίωση(configurations and wiring)

Οι ενότητες διαθέτουν την κατάσταση και εφαρμόζουν την εκτελέσιμη λογική. Εντούτοις, όπως όλα τα συστατικά, μπορούν μόνο να ονομάσουν λειτουργίες και μεταβλητές μέσα στα τοπικά namespaces τους, όπως καθορίζονται από τις υπογραφές τους. Για μια ενότητα να είναι ικανή να καλέσει άλλη, πρέπει να χαρτογραφήσουμε ένα σύνολο ονομάτων από το ένα συστατικό ένα σύνολο ονομάτων σε ένα άλλο συστατικό. Σε nesC, που συνδέει δύο συστατικά κατ' αυτό τον τρόπο καλούνται καλωδίωση. Επιπλέον στις ενότητες, η nesC έχει ένα δεύτερο είδος συστατικού, διαμορφώσεις (configurations), η εφαρμογή των οποίων είναι συστατικό καλωδίωσης. Οι ενότητες εφαρμόζουν τη λογική προγράμματος: οι διαμορφώσεις συνθέτουν τις ενότητες σε μεγαλύτερες αφαιρέσεις.

Σε ένα πρόγραμμα **TinyOS**, υπάρχουν συνήθως περισσότερες διαμορφώσεις από ενότητες. Υπάρχουν δύο λόγοι για αυτό. Κατ' αρχάς, εκτός από τις χαμηλού επιπέδου αφαιρέσεις υλικού, οποιοδήποτε δεδομένο συστατικό χτίζεται πάνω σε ένα σύνολο άλλων αφαιρέσεων, που είναι τοποθετημένες στις διαμορφώσεις. Παραδείγματος χάριν, ένας σωρός δρομολόγησης εξαρτάται από ένα single-hop στρώμα πακέτων, το οποίο είναι μια διαμόρφωση. αυτή η διαμόρφωση single-hop ενώνει την πραγματική εφαρμογή πρωτοκόλλου σε ένα ακατέργαστο στρώμα πακέτων πάνω από το radio.

Ουσιαστικά, η τοποθέτηση μιας αφαίρεσης A σε μια διαμόρφωση σημαίνει ότι μπορεί να είναι έτοιμη προς χρήση: το μόνο που έχουμε να κάνουμε είναι να καλωδιώσουμε στη λειτουργία A. Αντίθετα, εάν ήταν μια ενότητα που χρησιμοποιεί τις διεπαφές, θα ήταν ανάγκη να καλωδιώσουμε επάνω στις εξαρτήσεις A και τις απαιτήσεις. Εκτός από την καλωδίωση ενός συστατικού σε άλλο, οι διαμορφώσεις

πρέπει επίσης να εξαγάγουν τις διεπαφές. Αυτό είναι ένα άλλο είδος καλωδίωσης, εκτός αυτού, παρά να συνδεθούν ο προμηθευτής και ο χρήστης μια εξαγωγή χαρτογραφεί ένα όνομα σε άλλο. Αυτή η ιδέα είναι δύσκολη αρχικά και γίνετε καλυτέρα κατανοητή μετά από μερικά παραδείγματα. Οι διαμορφώσεις φαίνονται πολύ παρόμοιες με τις ενότητες. Έχουν μια προδιαγραφή και μια εφαρμογή. Αυτή είναι η διαμόρφωση LedsC, η οποία παρουσιάζει την αφαίρεση **TinyOS** των περιβόητων 3 LED:

```
configuration LedsC {
  provides interface Init @atleastonce();
  provides interface Leds;
}
implementation {
  components LedsP, PlatformLedsC;
  Init = LedsP;
  Leds = LedsP;

  LedsP.Led0 -> PlatformLedsC.Led0;
  LedsP.Led1 -> PlatformLedsC.Led1;
  LedsP.Led2 -> PlatformLedsC.Led2;
}
```

Listing 5.1: The LedsC configuration

Ταυτοχρονισμός στην NesC

Ο ταυτοχρονισμός (concurrency) παίζει βασικό ρόλο στα στοιχεία της NesC. Τα συμβάντα (ή οι εντολές) μπορούν να σηματοδοτηθούν άμεσα ή έμμεσα από μια διακοπή, κάτι που τα κατατάσσει στον τομέα των ασύγχρονων κωδίκων. Για να χειριστεί αυτόν τον ταυτοχρονισμό, η γλώσσα NesC παρέχει δυο εργαλεία, όπως αναφέραμε και προηγουμένως: τα atomic sections και τις εργασίες (tasks). Στην περίπτωση που ένας ασύγχρονος κώδικας αποκτά πρόσβαση σε μια μεταβλητή x, τότε κάθε πρόσβαση στη μεταβλητή αυτή έξω από μια atomic δήλωση είναι λάθος που χτυπάει κατά τη μεταγλώττιση. Για αυτό, ο προγραμματιστής πρέπει να δηλώσει το τμήμα αυτό ως ‘atomic’ ή να προωθήσει το ανάλογο τμήμα του κώδικα σε μια εργασία (task). Η χρήση ασύγχρονου (async) κώδικα για την απόκριση σε μια διακοπή υλικού πρέπει να γίνεται πολύ προσεκτικά.. Και αυτό γιατί, εκτελώντας ο ασύγχρονος κώδικας μια σχετικά χρονοβόρα διαδικασία επεξεργασίας, αναγκαστικά δεν αφήνει περιθώρια στον επεξεργαστή να χειριστεί αξιόπιστα άλλες διακοπές υλικού εκείνη τη στιγμή, με αποτέλεσμα το όλο σύστημα να χάνει σε ανταπόκριση και αξιοπιστία. Για αυτό το λόγο το μέγεθος της επεξεργασίας που εκτελεί ο

ασύγχρονος κώδικας πρέπει να είναι μικρό και τα atomic sections μέσα στον κώδικα πρέπει να αποφεύγουν την απευθείας κλήση εντολών ή σηματοδοσία συμβάντων όσο το δυνατόν περισσότερο.

Στην περίπτωση που ένας ασύγχρονος κώδικας έχει να εκτελέσει ένα μεγάλο επεξεργαστικό φόρτο μπορεί να θέσει μια εργασία η οποία θα αναλαμβάνει την επεξεργασία αυτή. Με αυτό τον τρόπο μεταφέρεται ο έλεγχος από ένα ασύγχρονο πλαίσιο εφαρμογής σε ένα σύγχρονο πλαίσιο. Η συγκεκριμένη εργασία θα εκτελεστεί σύγχρονα και, ενώ μπορεί να μην έχει ήδη ολοκληρωθεί, υπάρχει περίπτωση μια άλλη διακοπή υλικού να εμφανιστεί η οποία θα διαχειριστεί άμεσα (καθώς υπερτερεί της εργασίας) και με τον ίδιο τρόπο θα δημιουργηθεί μια νέα εργασία η οποία θα εκτελεστεί μόλις τελειώσει η προηγούμενη (καθώς η μια εργασία δεν μπορεί να προηγηθεί της άλλης).

Παραμετροθετημένες διεπαφές (Parameterized interfaces)

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

Σε ένα module, οι υλοποιήσιμες εντολές και τα συμβάντα μιας parameterized διεπαφής λαμβάνουν επιπλέον παραμέτρους που προσδιορίζουν την επιλεγμένη διεπαφή και επίσης επιλέγουν μια συγκεκριμένη διεπαφή όταν καλούν μια εντολή ή ένα συμβάν σε μια parameterized διεπαφή. Έτσι, για παράδειγμα, μπορούμε σε μια εφαρμογή να χρησιμοποιήσουμε πολλαπλούς χρονομετρητές (timers), καθένας από τους οποίους μπορεί να διαχειριστεί ανεξάρτητα. Μπορεί δηλαδή σε μια εφαρμογή ένα στοιχείο να χρειάζεται έναν timer που θα παίρνει τιμές από έναν αισθητήρα κάθε

δευτερόλεπτο, ενώ παράλληλα ένα άλλο στοιχείο θα θέλει έναν timer ο οποίος θα χρονομετρεί με διαφορετικό ρυθμό ώστε να διαχειρίζεται την ασύρματη μετάδοση.

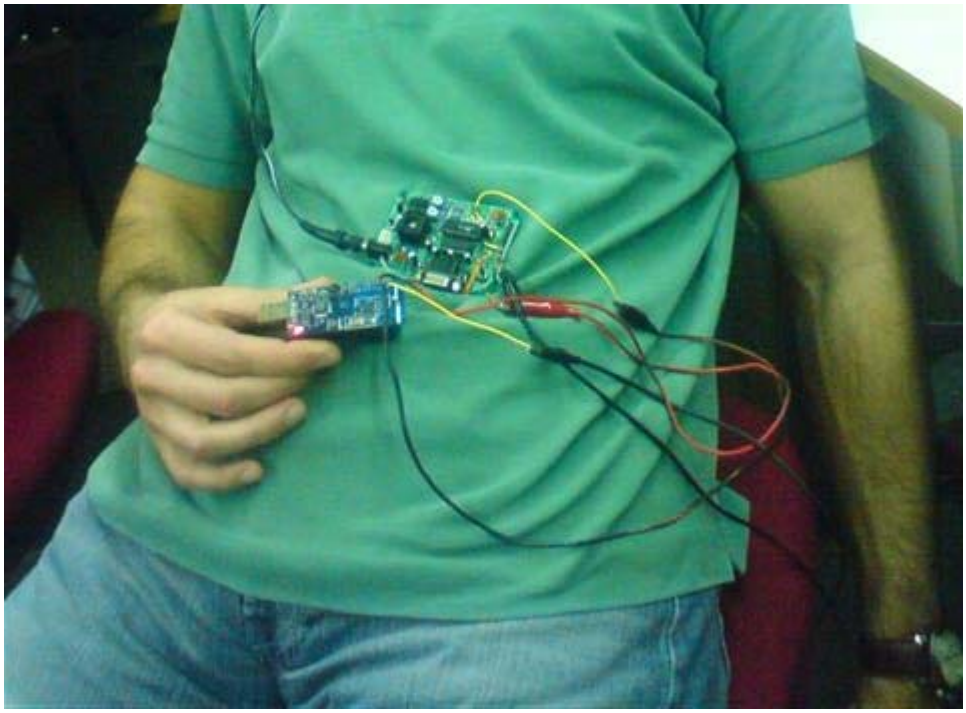
Συνδέοντας (wiring) τη διεπαφή Timer καθενός από αυτά τα στοιχεία σε διαφορετική υπόσταση της διεπαφής Timer, που παρέχεται από τον TimerC, μπορούμε να δώσουμε σε κάθε στοιχείο το δικό του ‘προσωπικό’ Timer.

Κεφάλαιο 4

Σχεδιασμός και υλοποίηση της Εφαρμογής

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μη επεμβατική παρατήρηση και επεξεργασία του βιοσήματος της ανθρώπινης αναπνοής. Η εφαρμογή αυτή φυσικά γίνεται στα πλαίσια της γενικότερης ιδέας της απομακρυσμένης παρακολούθησης ασθενών μέσω BSN τα οποία επεκτείνονται εκτός από το σήμα αναπνοής στη καταγραφή του ηλεκτροκαρδιογραφήματος, της θερμοκρασίας, της πίεσης, της κίνησης και διαφόρων άλλων σημάτων όπως αναφέραμε σε προηγούμενο σημείο.

Η μέτρηση του σήματος της αναπνοής γίνετε με της χρήση ενός αναλογικού επιταχυνσιόμετρου δύο διαστάσεων (MEMSIC MXD6020G/H/M/N με πλατφόρμα αξιολόγησης την MXEB-S-003) το οποίο καταγράφει της μετατοπίσεις στην κοιλιακή χώρα και το στήθος σαν αποτέλεσμα των εισπνοών και εκπνοών. Το μετρούμενο βιοσήμα δειγματοληπτείται εν συνεχεία από την ασύρματη μονάδα Tmote sky που λειτουργεί σαν κόμβος και εκπέμπεται προς μια άλλη μονάδα Tmote που λειτουργεί σαν σταθμός βάσης και προωθεί τα δεδομένα προς ένα υπολογιστή μέσω θύρας USB με τη βοήθεια κατάλληλων εργαλείων και εφαρμογών του **TinyOS**-2-x. που θα αναλυθούν στη συνέχεια.



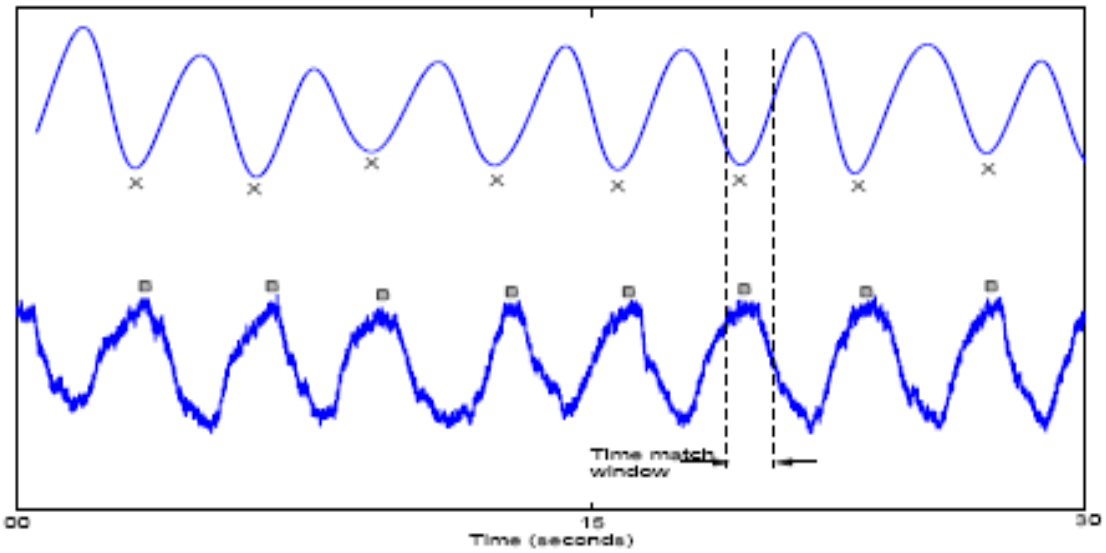
Εικόνα 42-Φωτογραφία κατα τη διάρκεια των μετρήσεων.Φαίνεται το evaluation Board και το mote.

Το βιοσήμα της αναπνοής.

Το βιοσήμα της αναπνοής είναι ένα σήμα υψίστης σημασίας .Σε συνδυασμό με άλλα βιοσήματα όπως το ηλεκτροκαρδιογράφημα και η αρτηριακή πίεση μπορούν να δώσουν μια πλήρη εικόνα της κατάστασης του υπό παρακολούθηση ασθενούς.

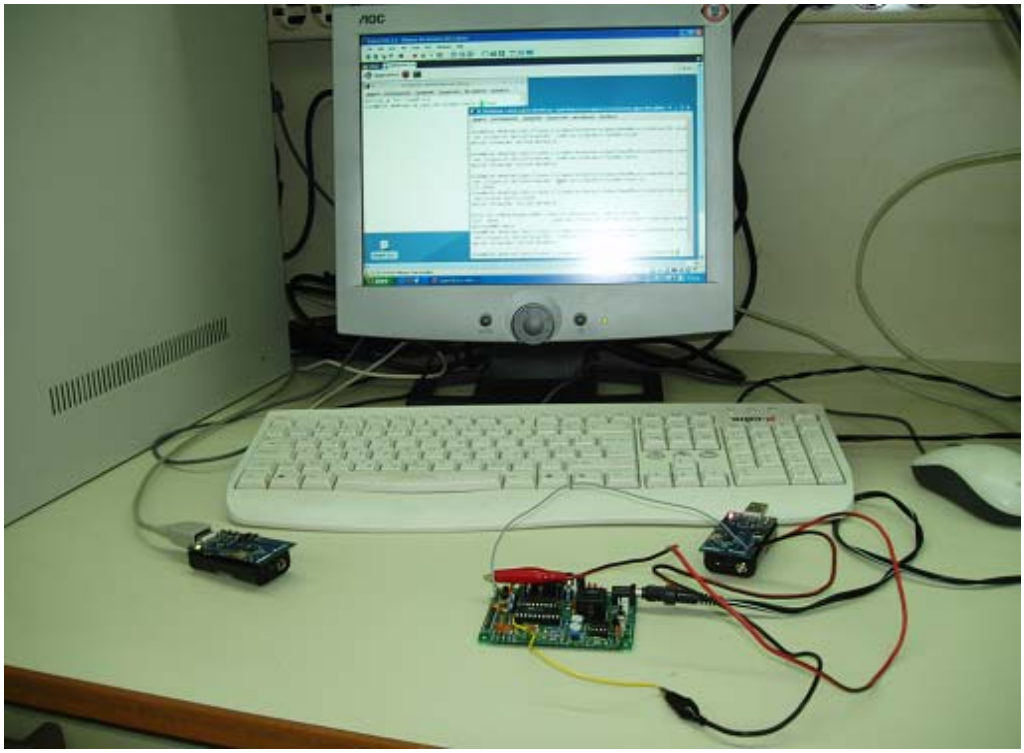
Γενικά υπάρχουν αρκετοί λόγοι για να παρακολουθείται ιατρικά η αναπνοή όπως η κατανόηση ορισμένων καρδιακών αρρυθμιών , χρησιμεύει στο συγχρονισμό ορισμένων εξετάσεων όπως το MRI (μαγνητική τομογραφία)σε σχέση με την καρδιά και το θώρακα. Σε περιπτώσεις μετεγχειρητικής ανάρρωσης τα αναλγητικά φάρμακα μπορούν να καταπιέσουν και να δυσχεραίνουν την αναπνοή του ασθενούν η οποία πρέπει να παρακολουθείται σε σταθερή βάση.

Σε συνδυασμό με την παρακολούθηση της κίνησης του ασθενούς εξάγονται συμπεράσματα από την αύξηση του ρυθμού αναπνοής και την ένταση της όσο αφορά την κατάσταση του. Επιπλέον ιδιαίτερο ενδιαφέρων παρουσιάζει το φαινόμενο της άπνοιας το οποίο επηρεάζει μεγάλο μέρος του πληθυσμού.



Εικόνα 43-Ενδεικτικό σήμα αναπνοής που λαμβάνεται από άμεση μέτρηση της αναπνοής με Nasal thermistor που χρησιμοποιείται σαν σήμα αναφοράς

Το σήμα της αναπνοής είναι ένα σύνθετο σήμα με χαμηλές συχνότητες. Αποτελείται κατά κύριο λόγω από δύο συνιστώσες μια χαμηλή (0-5 Hz) και μια πιο ψηλή (2-40Hz). Συσπάσεις και κινήσεις των μυών και του διαφράγματος εμπεριέχονται στο σήμα αυτό ειδικά κατά τη λήψη του μέσω επιταχυνσιομέτρου.
Το σήμα αυτό θεωρείται στοχαστικό και μη στατικό κάτι που θα μας απασχολήσει στο στάδιο της επεξεργασίας.



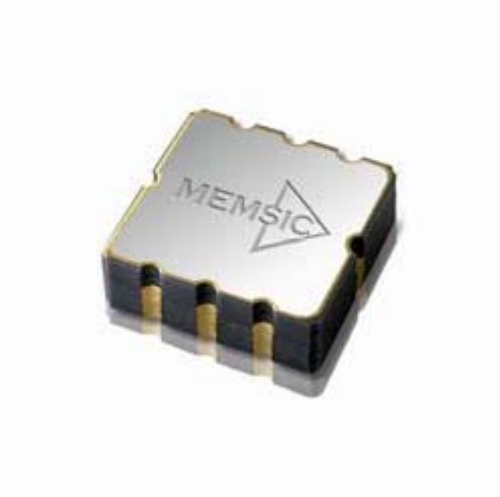
Εικόνα 44-Η διάταξη και ο εξοπλισμός που χρησιμοποιήθηκαν για τις μετρήσεις στο εργαστήριο. Το evaluation board με το accelerometre συνδέονται με καλώδια στο mote που επικοινωνεί ασύρματα με το basestation.

Το επιταχυνσιόμετρο

Για την μη παρεμβατική παρακολούθηση του σήματος της αναπνοής χρησιμοποιήθηκε ένα αναλογικό επιταχυνσιόμετρο 2 αξόνων από την Memsic που τοποθετήθηκε στην κατάλληλη βάση του evaluation board.

Αρχή λειτουργίας.

Τα επιταχύνσιόμετρα MEMSIC είναι πλήρη συστήματα μέτρησης κινήσεων δύο αξόνων σε έναν μονολιθικό ολοκληρωμένο κύκλωμα CMOS. Η αρχή της λειτουργίας του βασίζεται στη μεταφορά θερμότητας από φυσική μετατόπιση. Οι συσκευές αυτές μετρούν τις εσωτερικές αλλαγές στη μεταφορά θερμότητας που προκαλείται από την επιτάχυνση. Οι συσκευές είναι λειτουργικά παρόμοιες με τα παραδοσιακά επιταχυνσιόμετρα ανίχνευσης -μάζας.

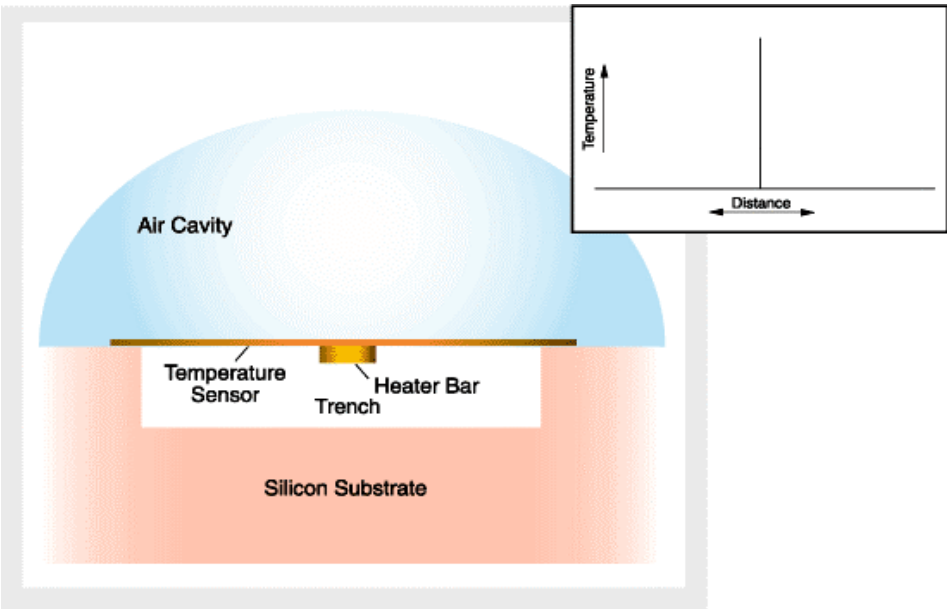


Εικόνα 45-Το επιταχυνσιόμετρο Memsic.

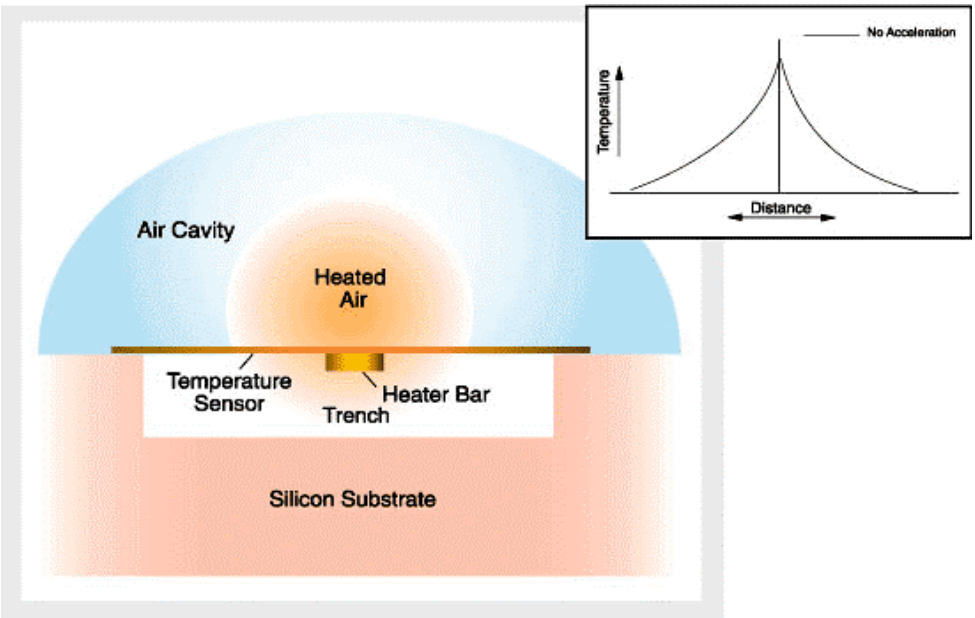
Η μάζα ανίχνευσης στον αισθητήρα MEMSIC είναι ένα αέριο. Η ανίχνευση αεριώδης μάζας παρέχει μεγάλα πλεονεκτήματα σε σχέση με τη χρήση της παραδοσιακής στερεάς μάζας απόδειξης. Η συσκευή δεν επιδέχεται προβλήματα μόλυνσης σωματιδίων που συνδέονται με τις ανταγωνιστικές συσκευές και παρέχει ανοχή κλονισμού μέχρι 50,000g που οδηγεί σε σημαντικά χαμηλότερα ποσοστά αποτυχίας και χαμηλότερη απώλεια κατά το χειρισμό και τη διάρκεια της συναρμολόγησης.

Μια ενιαία πηγή θερμότητας, που τοποθετείται στο κέντρο του τσιπ πυριτίου είναι ανασταλμένη σε μια κοιλότητα. Σε ίσες αποστάσεις θερμοζεύγη αργιλίου/πολυπυρίτιων βρίσκονται κατανεμημένα σε ίσες αποστάσεις και στις τέσσερις πλευρές της πηγής θερμότητας (διπλός άξονας). Σε μηδενική επιτάχυνση, η κλίση θερμοκρασίας είναι συμμετρική με την πηγή θερμότητας, έτσι ώστε η θερμοκρασία είναι ίση και στα τέσσερα θερμοζεύγη , που παράγουν την ίδια τάση (σχήμα 1).

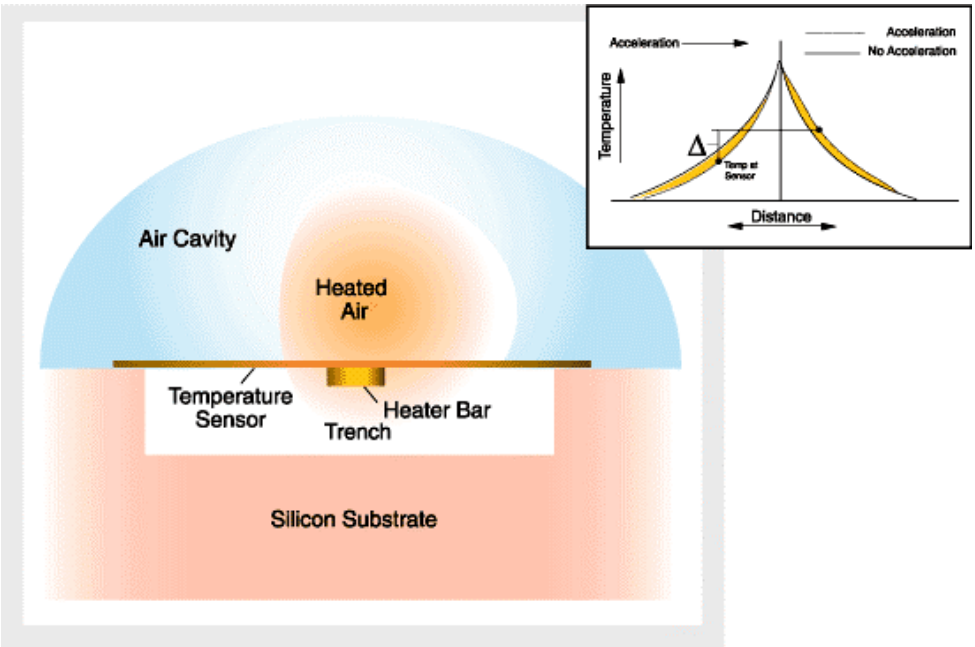
Η επιτάχυνση σε οποιαδήποτε κατεύθυνση θα διαταράξει το διάγραμμα θερμοκρασίας, λόγω της ελεύθερης μεταφοράς θερμότητας , που γίνεται ασύμμετρη. Η θερμοκρασία, και ως εκ τούτου η παραγωγή τάσης από τα τέσσερα θερμοζεύγη διαφοροποιείται. Η παράγωγος της τάσης ως προς την έξοδο στα θερμοζεύγη είναι ανάλογη προς την επιτάχυνση. Υπάρχουν δύο ίδιες πορείες σημάτων επιτάχυνσης στη συσκευή, μια για να μετράει την επιτάχυνση στον Χ-άξονα και μια μετράει επιτάχυνση Υ-άξονα.



Εικόνα 46-Εικόνα της αρχής λειτουργίας του επιταχυνσιομέτρου.



Εικόνα 47-Αρχή λειτουργίαςς του επιταχυνσιόμετρου. Η θέση του θερμού αέρα δηλώνει τις μεταβολές της επιτάχυνσης.

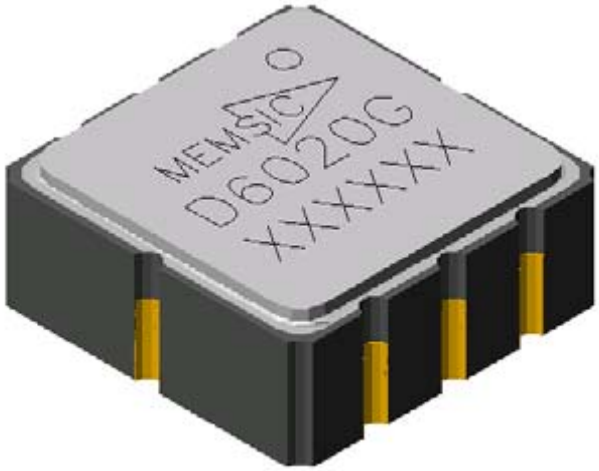


Εικόνα 48-Σύγκριση της κατακόρυφης μετατόπισης του θερμού αέρα στα θερμοζεύγοι.

Εύρος μετρήσεων και έξοδοι επιταχυνσιομέτρων

Οι συσκευές MEMSIC είναι ικανές να μετρήσουν επιταχύνσεις με μια τάξη μεγέθους κάτω από ± 1.0g και πάνω από ±100g. Οι συσκευές αυτές μπορούν να μετρήσουν τη δυναμική επιτάχυνση (π.χ. δόνηση) και στατική επιτάχυνση (π.χ. βαρύτητα).

Οι συσκευές μπορούν να παρέχουν αναλογική ή ψηφιακή τάση. Οι αναλογικές τάσεις εξόδου είναι διαθέσιμες με απόλυτο και ποσοστιαίο τρόπο. Η απόλυτη τάση είναι ανεξάρτητη από την τάση τροφοδοσίας , ενώ η ποσοστιαία τάση εξόδου είναι ανάλογη προς την τάση τροφοδοσίας. Οι ψηφιακές έξοδοι είναι σήματα με Duty cycle (αναλογία του πλάτους παλμού στην περίοδο) που ποικίλλει ανάλογα με την επιτάχυνση.



Εικόνα 49-Το επιταχυνσιόμετρο Memsic.

Περιγραφές ακίδων

Vdd- Η τάση τροφοδοσίας που εισάγεται για τα ψηφιακά κυκλώματα και τη θερμάστρα αισθητήρων. Η dc τάση πρέπει να είναι μεταξύ 2,70V και 5,.5V.

Vda- αυτό είναι η παροχή ηλεκτρικού ρεύματος που εισάγεται για τους αναλογικούς ενισχυτές.

Gnd - Γείωση

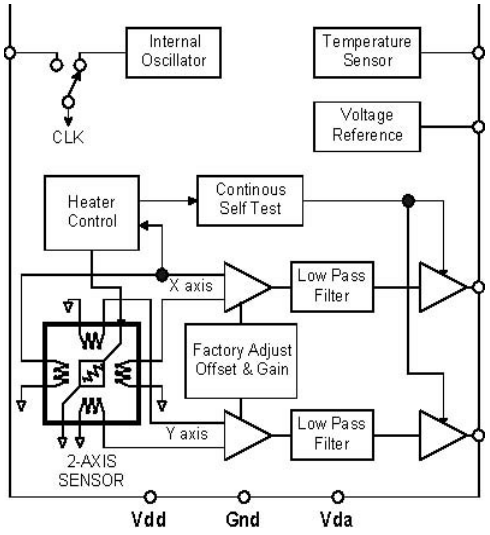
Aout- Έξοδος για το X-άξονα

Aout- Έξοδος για το Y-άξονα

T out- Ένδειξη της εσωτερικής θερμοκρασίας

Sck - η τυποποιημένη έξοδος παραδίδεται με εσωτερική επιλογή ρολογιών (800kHz)..

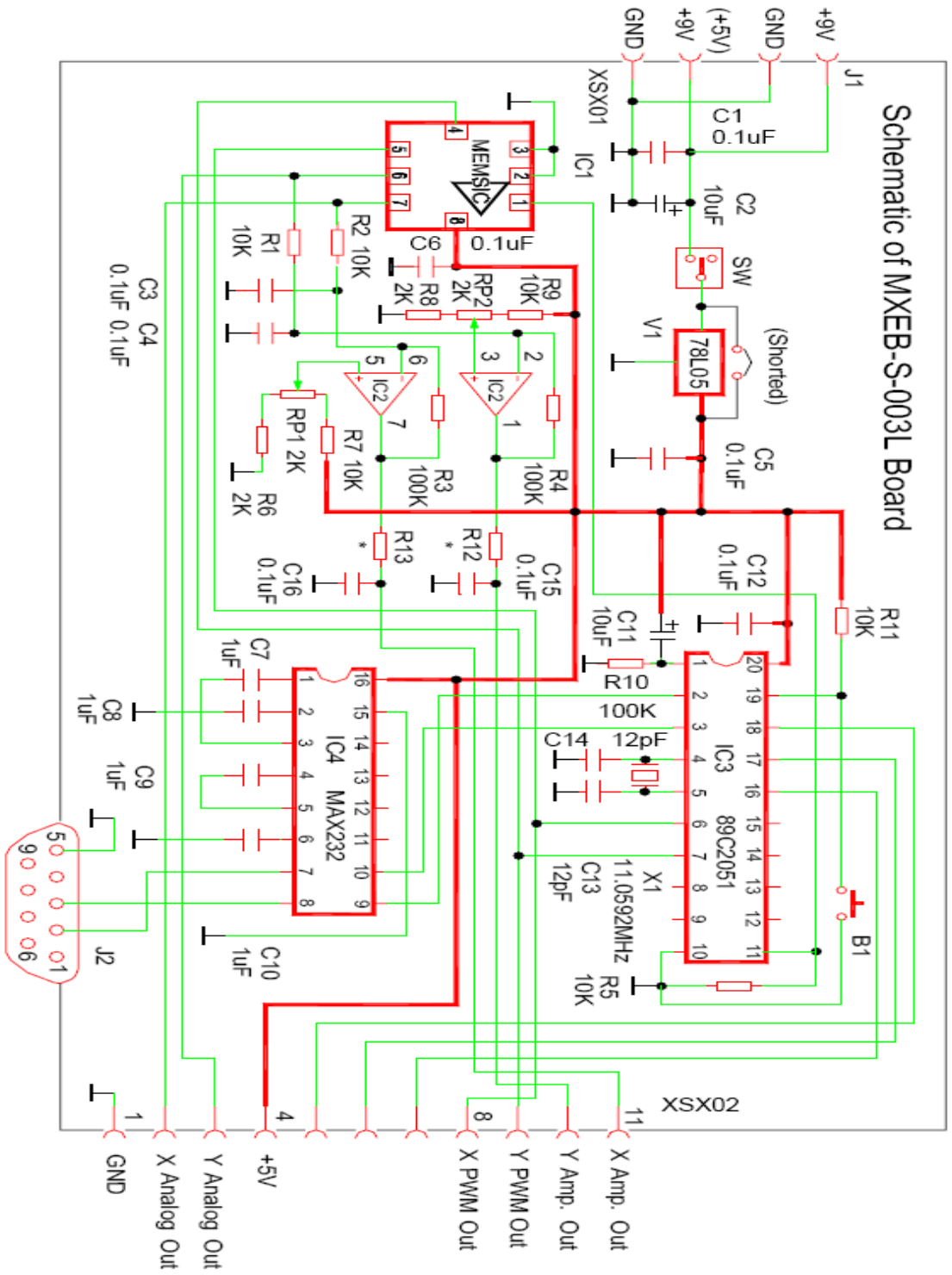
Vref- Τάση αναφοράς



Pin Description: LCC-8 Package		
	Nam	Description
	TOU	Temperature (Analog Voltage)
	AOU	Y-Axis Acceleration Signal
	Gnd	Ground
	VDA	Analog Supply Voltage
	AOU	X-Axis Acceleration Signal
	Vref	2.5V Reference
	Sck	Optional External Clock
	VDD	Digital Supply Voltage

Πλατφόρμα Αξιολόγησης (Evaluation Board)

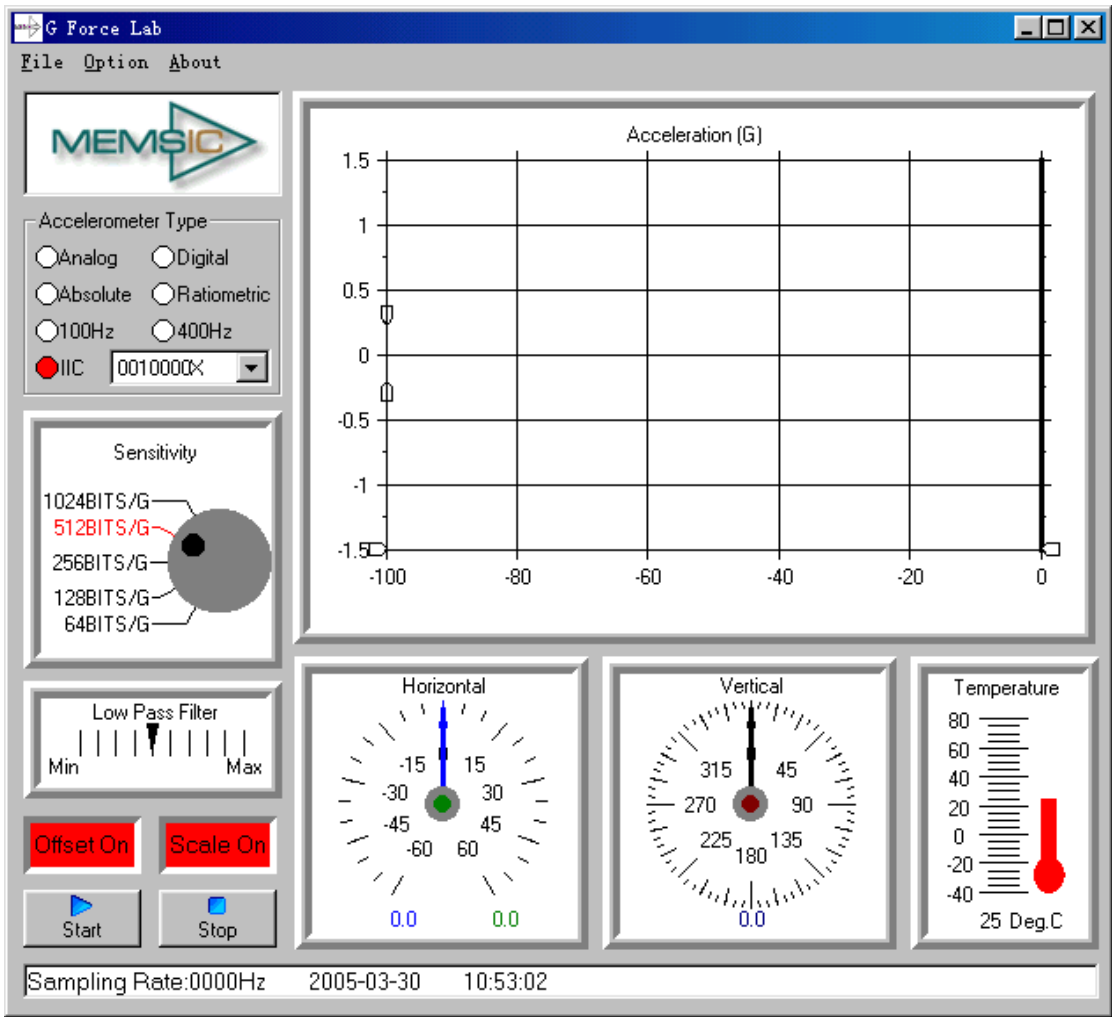
Εδώ φαίνεται ολόκληρο το διάγραμμα της πλατφόρμας αξιολόγησης στην οποία τοποθετείτε το αναλογικό ή ψηφιακό επιταχυνσιόμετρο και λαμβάνονται οι ανάλογες έξοδοι από τις κατάλληλες ακίδες



Εικόνα 50-Το evaluation board

Gforse Lab

Στην περίπτωση του ψηφιακού επιταχυνσιόμετρου το σήμα μπορεί να παρασταθεί άμεσα στον υπολογιστή με το ειδικό πρόγραμμα Force Lab.Στην περίπτωση αυτή η πλατφόρμα συνδέετε με τον υπολογιστή μέσω της σειριακής θύρας.



Εικόνα 51-Το πρόγραμμα Gforse lab για την καταγραφή του σήματος ψηφιακού επιταχυνσιόμετρου.

Η εφαρμογή Ecosensory.

Όπως αναφέρθηκε και αλλού στην παρούσα διπλωματική εργασία ένας ασύρματος κόμβος αισθητήρων mote εκτελούσε δειγματοληψία του αναλογικού σήματος που έμπαινε σαν είσοδος στις θύρες του ADC. Το mote είχε ως αποστολή την δειγματοληψία του σήματος αυτού με μια συχνότητα που καθορίστηκε ανάλογα με τον τύπο του σήματος και στη συνέχεια αφού έφτιαξε τα δεδομένα σε πακέτα τα στέλνει ασύρματα προς μια άλλη ασύρματη μονάδα αισθητήρων που λειτουργεί σαν σταθμός βάσης. Η βασική πρόκληση της όλης εργασίας ήταν η πολυκαναλλική δειγματοληψία που επιτεύχθηκε με την κατάλληλη χρήση της εφαρμογής Ecosensory και των προγραμμάτων ReadMoistureSensors .

Ακολουθεί μια περιληπτική εξήγηση κάποιων βασικών στοιχείων του κώδικα. Το ‘σύνολο του σχετικού κώδικα με όλα τα σχετικά αρχεία μπορεί να αναζητηθεί στο παράρτημα.

Το Πρόγραμμα ReadMoisotureSensorsP.nc

Το πρόγραμμα αυτό έχει αναφορά στις βιβλιοθήκες Timer.h , ReadMoistureSensors.h, Msp430Adc12.h και a2d12ch.h. Το πρόγραμμα αυτό εκτελεί τις βασικές λειτουργίες της εφαρμογής. Κατά την πυροδότηση των γεγονότων(event) timer εκτελείται διαδοχικά η όλη διαδικασία.

Κατά την πυροδότηση του timer0 προετοιμάζεται το σύστημα για την λήψη των μετρήσεων. Οι χρόνοι προθέρμανσης και τερματισμού ορίζονται στο αρχείο ReadMoistureSensors.h. Ακόμα προετοιμάζεται η πρώτη τράπεζα δεδομένων (bank1) να δεχτεί τα δεδομένα. Στο γεγονός timer0 καλείται και η πυροδότηση των γεγονότων timer1και timer3 για την λήψη των μετρήσεων.

```
event void timer0.fired() {   //From Timer<TMilli>
//    timerend = call  timer0.getNow();  // time senvdd1 was turned on.  del a.
//    call timer3.startOneShotAt(timerend, ALWAYS_SHUTOFF_MILLI);    del a.
    call timer1.startOneShot(ECH2O_WARMUP_MILLI);
    call timer3.startOneShot(ALWAYS_SHUTOFF_MILLI);
    atomic  { bank1 = TRUE;  // next data ready goes in bank1 data slots.
    }
    call a2dmuxdisable.clr();  //timer1 -->  LO to choose mux enabled.
    call a2dbankselect.clr();  //timer0 -->   LO to choose SENSIG1 muxed.
    call a2dsenvdd1drv.set();  //timer0 -->  set HI a2dsenvdd1drv.
    atomic {
      timestamp++;   //a simplistic timestamp value to put in data slot for now.
```

```
    }
   //   to let the ECH2O sensors get setup time done.
  }
```

Με την πυροδότηση του timer1 τα δεδομένα από τον ADC τοποθετούνται στη δομή που ορίζει το ReadMoistureSensors.h και καλείται το timer2.

```
event void timer1.fired() {   //timer1 dt = ECH2O_WARMUP_MILLI
//    timerend = call  timer1.getNow();  // time senvdd2 was turned on.  del a.
    call timer2.startOneShot(ECH2O_WARMUP_MILLI);
    call a2dsenvdd2drv.set();  //timer1 -->  set HI a2dsenvdd2drv.
    atomic  {
      if (!busy) {       //AMSend not in progress
        call Resource.request();  // Resource --> adc12multichannel
    // 08jan08JG  get the outgoing packet payload pointer, (and default len).
    //09dec07jg    //cast result of .getPayload to pointer rmspkt.
  //  rmspkt = (MoistureSensorsMsg*) (call AMSend.getPayload(&pkt, pktlen));
 rmspkt = (MoistureSensorsMsg*) (call AMSend.getPayload(&pkt));
    rmspkt->nodeid = TOS_NODEID;  //  outgoing packet struct empty of a2d data.
      }
    }
  }
```

Το γεγονός timer2 προετοιμάζει το σύστημα για την επόμενη μέτρηση και ορίζει την bank2 να δεχτεί τα δεδομένα και το γεγονός timer3 εκτελει την μέτρηση από τονADC.

```
event void timer2.fired() {   //timer2 dt = ECH2O_WARMUP_MILLI
    call a2dmuxdisable.clr();  //timer2 -->  LO to choose mux enabled.
    call a2dbankselect.set();  //timer1 -->   HI to choose SENSIG2 muxed.
    atomic  {
    bank1 = FALSE;  // next data ready goes in bank2 data slots.
      if (!busy) {       //AMSend not in progress
        call Resource.request();  // Resource --> adc12multichannel
    // use the same outgoing packet payload pointer, (and default len).
      }
    }
  }
```

Η διαχείριση των 2 τραπεζών δεδομένων γινεται από το ασύγχρονο γεγονός a2d12ch.dataReady

```
async event void a2d12ch.dataReady(uint16_t *readybuf, uint16_t readybuflen)
  {
    atomic {
    bufferlen = readybuflen;  //probably can just not use *readybuf
    }
    //sort adc read values into nx_struct MoistureSensorsMsg.
    if (bank1 == TRUE) {       // async handling of banks.
      rmspkt->timestamp = timestamp;  // 2nd bank will have delta from stamp.
      post MoistureSensorsMsgBank1();
    }
    else {
      post MoistureSensorsMsgBank2();
// do something to send message now
    post MoistureSensorsMsgSend();
    }
  }
```

Η λήψη των μετρήσεων από τον ADC γινεται κατά την πυροδότηση των γεγονότων timer1 και timer3 που καλούν το γεγονός Resource.granted()

```
event void Resource.granted()  //resource meaning is "just the a2d channel"
  {
    const msp430adc12_channel_config_t* a2d12chconfig = call
AdcConfigure.getConfiguration();  //to get setup from a2d12ch.
       // start the adc read, ( use a2d12chconfig from above).
    atomic {
    if ( call a2d12ch.configure(a2d12chconfig, memctl, numMemctl, buffer, bufferlen,
jiffies) == SUCCESS){
      call Leds.led0On();  //debug aid
      call a2d12ch.getData();  // returns buffer[], bufferlen integer
    }
```

Ο τρόπος που τα δεδομένα από το κάθε κανάλι θα τοποθετηθούν στο buffer με βάση την δομή που έχει οριστεί για τις 2 banks γίνεται στα 2 tasks που φαίνονται πιο κάτω και τα οποία τροποποιήθηκαν κατάλληλα για τις ανάγκες της παρούσας εφαρμογής ώστε το σύστημα να εκτελεί δειγματοληψία μόνο στα κανάλια ADC0, ADC1, ADC2.

```
task void MoistureSensorsMsgBank1()
  {
    //assign buffer[] elements to rmspkt  nx_struct elements.
    atomic
    {
      rmspkt->adc00 = buffer[0];
      rmspkt->adc01 = buffer[1];
      rmspkt->adc02 = buffer[2];
```

```
    // rmspkt->adc03 = buffer[3];
   //  rmspkt->adc04 = buffer[4];
  //   rmspkt->adc05 = buffer[5];
   }
   //rmspkt struct is now filled with new bank1 data.
     //  shut off bank1 related.
     call Resource.release();
     call a2dsenvdd1drv.clr();  //dataReady -->  a2dsenvdd1drv = LO.
  }

  task void MoistureSensorsMsgBank2()
  {
    atomic
    {
      rmspkt->adc10 = buffer[0];
      rmspkt->adc11 = buffer[1];
      rmspkt->adc12 = buffer[2];
    //  rmspkt->adc13 = buffer[3];
    //  rmspkt->adc14 = buffer[4];
   //   rmspkt->adc15 = buffer[5];
    }
      //  shut off bank2 related.
     call a2dmuxdisable.set();  //data stored -->  HI disables mux.
     call a2dsenvdd2drv.clr();  //data stored -->  a2dsenvdd2drv = LO.
     call Resource.release();
  }
```

Όταν η μνήμη γεμίσει, τα δεδομένα αποστέλλονται μέσω του ασύρματου συστήματος στο σταθμό βάσης. Τη διαδικασία χειρίζεται το γεγονός MoistureSensorsMsgSend()

```
task void MoistureSensorsMsgSend()
  {

    call Leds.led1On();  //debug aid
    if (call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(MoistureSensorsMsg)) == SUCCESS) {
      call Leds.led2On();  //debug aid
      busy = TRUE;  //the data is queued to go out.
    }
  }
```

Το πρόγραμμα ReadMoistureSensorsC.nc

Περιλαμβάνει της βιβλιοθήκες ReadMoistureSensors.h και Msp430Adc12ch.h. Το πρόγραμμα αυτό λαμβάνει μετρήσεις για τις 2 τράπεζες δεδομένων του sensorboard a2d12ch. Η διαχείριση των τραπεζών γίνετε όπως είδαμε και πιο πάνω στο ReadMoistureSensorsP μέσω του HplMsp430GeneralIO.

Το module a2d12ch έχει ως βασική λειτουργία την πολυκαναλλική δειγματοληψία και όχι τις λειτουργίες εισόδου εξόδου. Το interface multichannel.getData επιστρέφει τα 2 προσωρινά buffers σαν ένα .

```
#include <Timer.h>
#include "ReadMoistureSensors.h"
#include "Msp430Adc12.h"

configuration ReadMoistureSensorsC {
}
implementation {
  components MainC, LedsC;
  components ActiveMessageC;
  components new AMSenderC(AM_MOISTURESENSORSMSG);
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;
  components new TimerMilliC() as Timer3;
  components ReadMoistureSensorsP;
  ReadMoistureSensorsP.Boot -> MainC;
  ReadMoistureSensorsP.Leds -> LedsC;
  ReadMoistureSensorsP.timer0 -> Timer0;
  ReadMoistureSensorsP.timer1 -> Timer1;
  ReadMoistureSensorsP.timer2 -> Timer2;
  ReadMoistureSensorsP.timer3 -> Timer3;
  ReadMoistureSensorsP.AMSend -> AMSenderC;
  ReadMoistureSensorsP.AMRadioOn -> ActiveMessageC;
  components a2d12chP, a2d12chC;
  ReadMoistureSensorsP.a2d12ch -> a2d12chP.a2d12ch;
//Msp430Adc12MultiChannel
  ReadMoistureSensorsP.AdcConfigure -> a2d12chP.AdcConfigure;
//Msp430Adc12MultiChannel
  ReadMoistureSensorsP.Resource -> a2d12chP.Resource; //ResourceRVG (first
thing to do)

  components HplMsp430GeneralIOC;
  ReadMoistureSensorsP.a2dmuxdisable -> HplMsp430GeneralIOC.Port21;
  ReadMoistureSensorsP.a2dbankselect -> HplMsp430GeneralIOC.Port23;
  ReadMoistureSensorsP.a2dsenvdd1drv -> HplMsp430GeneralIOC.ADC7;
  ReadMoistureSensorsP.a2dsenvdd2drv -> HplMsp430GeneralIOC.ADC6;
  ReadMoistureSensorsP.a2dterm2drv -> HplMsp430GeneralIOC.Port17;
```

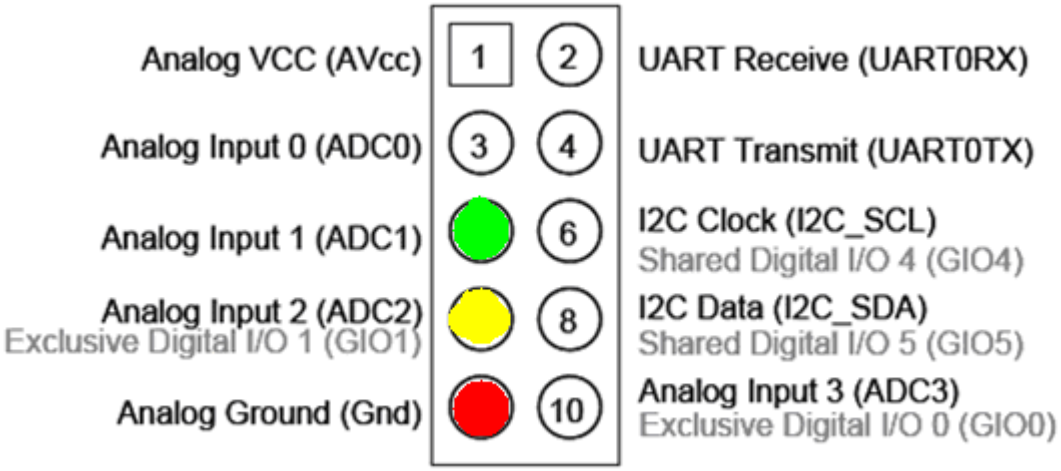
Το αρχείο ReadMoistureSensors.h

Το αρχείο αυτό ορίζει τους χρόνους προθέρμανσης του συστήματος και την περίοδο λειτουργίας για κάθε σειρά μετρήσεων. Επιπρόσθετα ορίζει τη δομή των δεδομένων.

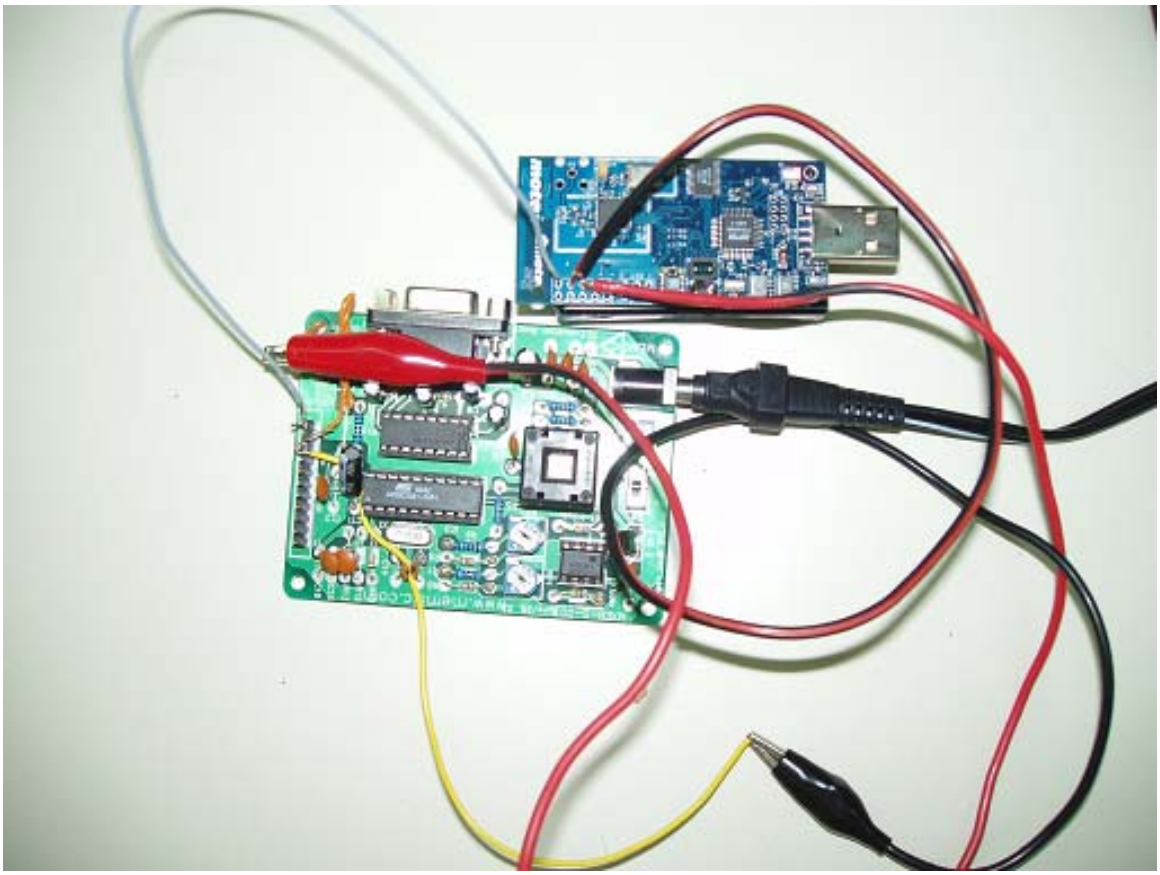
```
#ifndef READMOISTURESENSORS_H
#define READMOISTURESENSORS_H

enum {
  ALWAYS_SHUTOFF_MILLI = 30,    // a2d reading delta t after sensors powered.
  ECH2O_WARMUP_MILLI = 10,    // a2d reading delta t after sensors powered.
  TIMER_PERIOD_MILLI = 50,   // a2d reading period.
//    AM_MOISTURESENSORSMSG is an active message type.  Depends on Makefile
//  BUILD_EXTRA_DEPS=MoistureSensorsMsg.class
//  MoistureSensorsMsg.class:  MoistureSensorsMsg.java
//  and MoistureSensorsMsg.java:
//  see tos/types/AM.h for other basic definitions related.
  AM_MOISTURESENSORSMSG = 3,
  TOS_NODEID = 22
};
typedef nx_struct MoistureSensorsMsg {
  nx_uint16_t nodeid;
  nx_uint16_t adc00; // bank 0, channel 0
  nx_uint16_t adc01; // bank 0, channel 1
  nx_uint16_t adc02; // bank 0, channel 2
  //nx_uint16_t adc03; // bank 0, channel 3
  //nx_uint16_t adc04; // bank 0, channel 4
  //nx_uint16_t adc05; // bank 0, channel 5
  nx_uint16_t adc10; // bank 1, channel 0
  nx_uint16_t adc11; // bank 1, channel 1
  nx_uint16_t adc12; // bank 1, channel 2
  //nx_uint16_t adc13; // bank 1, channel 3
  //nx_uint16_t adc14; // bank 1, channel 4
  //nx_uint16_t adc15; // bank 1, channel 5
  nx_uint16_t timestamp;
} MoistureSensorsMsg;
```

Περιγραφή εφαρμογής

Για τη μέτρηση του βιοσήματος της αναπνοής με μη επεμβατικό τρόπο μέσω WSN με τη χρήση αναλογικού επιταχυνσιομέτρου τοποθετήθηκε η πλατφόρμα αξιολόγησης (evaluation board) στο σημείο μέτρησης του σώματος του ασθενούς και συνδέθηκε με την ασύρματη πλατφόρμα tmote με καλώδια. Κατακρίβεια συνδέθηκαν οι αναλογικές εξόδοι 1 και 2 της πλατφόρμας αξιολόγησης με τις αναλογικές εισόδους ADC1 και ADC2 του mote αντίστοιχα, οι οποίες εκτελούν δειγματοληψία του αναλογικού σήματος με συχνότητα δειγματοληψίας που καθορίστηκε στον κώδικα και την γείωση με την είσοδο 9 του mote δηλαδή την αναλογική γείωση. Εν συνεχεία το mote επεξεργάζεται τα δεδομένα σε πακέτα και τα στέλνει ασύρματα προς τον σταθμό βάσης. Ο σταθμός βάσης είναι ένα άλλο mote στο οποίο εγκαταστάθηκε η εφαρμογή Basestation και το οποίο είναι συνδεδεμένο σε USB θύρα υπολογιστή. Το λειτουργικό σύστημα **TinyOS** μέσω του εργαλείου Serial Forwarder προωθεί τα πακέτα στον υπολογιστή τα οποία στη συνέχεια μπορούν να οπτικοποιήθούν στην οθόνη σε πραγματικό χρόνο (real time) ή να αποθηκεύονται σε αρχεία .txt με χρήση των εργαλείων Listen και Mig .



Εικόνα 52--Ο συνδετήρας επέκτασης των 10 pins.Στις θύρες ADC1 και ADC2 συνδέσαμε τις εξόδους X και Y του evaluation Board καθώς και τη γείωση.



Εικόνα 53-Φωτογραφεία του mote συνδεδεμένου με το evaluation Board κατα την πειραματική διαδικασία.

Επίπεδο κώδικα

Σε ότι αφορά το λογισμικό της εφαρμογής η εργάσία έγινε στο **TinyOS**.2-x που λειτουργεί μέσο της διανομής Linux Xubuntu μιας βελτιωμένης σε σχέση με το Cygwin έκδοσης. Η εφαρμογή Basestation που χρησιμοποιήθηκε για το σταθμό βάσης που προωθεί δεδομένα προς τον υπολογιστή προσφέρεται έτοιμη και συνδυάζεται με πληθώρα άλλων εφαρμογών που υπάρχουν στο **TinyOS**-2.x.

Βασικό πρόβλημα στην εφαρμογή ήταν η ανάπτυξη κώδικα σε NesC για να δειγματοληπτεί τουλάχιστον δύο από τις εξωτερικές θύρες του συνδετήρα επέκτασης ώστε να τροφοδοτεί σαν είσοδο τον κάθε άξονα του επιταχυνσιομέτρου ξεχωριστά . Αυτό γίνεται με κατάλληλη χρήση του interface Multichannel επίπεδο προσαρμογής υλικού HAL hardware adaptation Layer .Σαν βάση της εφαρμογής χρησιμοποίηθηκε ο κώδικας της εφαρμογής Ecosensory που βρίσκετε στην ιστόθεση http://nesc.sourceforge.net . Ο κώδικας αυτός διαβάζει για 2 τράπεζες δεδομένων με το a2d12ch module. Τις τράπεζες δεδομένων διαχειρίζεται το component

HplMsp430GeneralOI στο ReadMoistureSensorsP.nc. Στην ουσία ο ADC δειγματοληπτεί τα έξι κανάλια του συνδετήρα επέκτασης δύο φορές σε μια περίοδο με διαφορά χρόνου dt , γεμίζοντας έτσι στη κάθε ανάγνωση και μια από τις 2 τράπεζες δεδομένων. Ακολούθως τα δεδομένα πακετοποιούνται με την δομή που ορίζει ο κώδικας και αποστέλλονται προς τον σταθμό βάσης μέσω του δικτύου με συχνότητα αποστολής ίδια με τη συχνότητα δειγματοληψίας του ADC όπως αυτή ορίζεται στο αρχείο βιβλιοθήκης του

TinyOS-2.x ReadMoistureSensors.h .

Οι βασικές μετατροπές που έγιναν στον κώδικα αφορούσαν το μέγεθος των πακέτων δεδομένων .Το επιταχυνσιόμετρο είναι διαξονικό οπότε χρειάζονται μόνο 2 θύρες στο mote για να ληφθεί το σήμα εξόδου. Οι υπόλοιπες 4 δεσμεύουν χώρο για δεδομένα αλλά μένουν κενές. Έτσι τροποποίηθηκε ο κώδικας ώστε να δειγματοληπτεί και να στέλνει στη βάση δεδομένα μόνο από τις θύρες ADC0, ADC1 και ADC2 και πάλι αφήνοντας την ADC0 κενή με την προοπτική επέκτασης με τριαξονικό επιταχυνσιόμετρο.

Η συχνότητα δειγματοληψίας και κατεπέκταση ο ρυθμός δεδομένων στο δίκτυο καθοριστήκαν στο αρχείο ReadMoistureSensors.h καθώς και λοιπές παράμετροι του προγράμματος όπως ο χρόνος προθέρμανσης των αισθητήρων, και ο χρόνος dt, ο χρόνος που μεσολαβεί στα γεμίσματα των δύο τραπεζών δεδομένων.

```
enum {
  ALWAYS_SHUTOFF_MILLI = 30,    // a2d reading delta t after sensors powered.
  ECH2O_WARMUP_MILLI = 10,    // a2d reading delta t after sensors powered.
  TIMER_PERIOD_MILLI = 50,   // a2d reading period.
```

Ο χρόνος εδώ είναι σε milliseconds και αντιστοιχεί σε περίοδο δειγματοληψίας
Fs = (1/50)*1000 = 20 Hz.

Ακόμα στο ίδιο αρχείο καθορισαμε τη δομή των πακέτων δεδομένων.

```
typedef nx_struct MoistureSensorsMsg {
  nx_uint16_t nodeid;
  nx_uint16_t adc00; // bank 0, channel 0
  nx_uint16_t adc01; // bank 0, channel 1
  nx_uint16_t adc02; // bank 0, channel 2

  nx_uint16_t adc10; // bank 1, channel 0
```

```
nx_uint16_t adc11; // bank 1, channel 1
nx_uint16_t adc12; // bank 1, channel 2
nx_uint16_t timestamp;
```

Εργαλεία Listen και Mig

Για την παρουσίαση σε παρόντα χρόνο(real time) και αποθήκευση των μετρήσεων του συστήματος χρησιμοποιήθηκαν τα εργαλεία Listen και Mig. Φυσικά τα εργαλεία αυτά δουλεύουν παράλληλα με την εφαρμογή JAVA τον Serial Forwarder που στέλνει τα δεδομένα από το σταθμό βάσης που φτάνουν μέσω του ασύρματου δικτύου στον υπολογιστή.

Με την εντολή

```
java net.tinyos.sf.SerialForwarder -comm.serial@/dev/ttyUSB0:tmote
```

ενεργοποιείται ο Serial Forwarder και παρουσιάζεται το παράθυρο που προωθεί τα πακέτα δεδομένων (εικόνα 54).

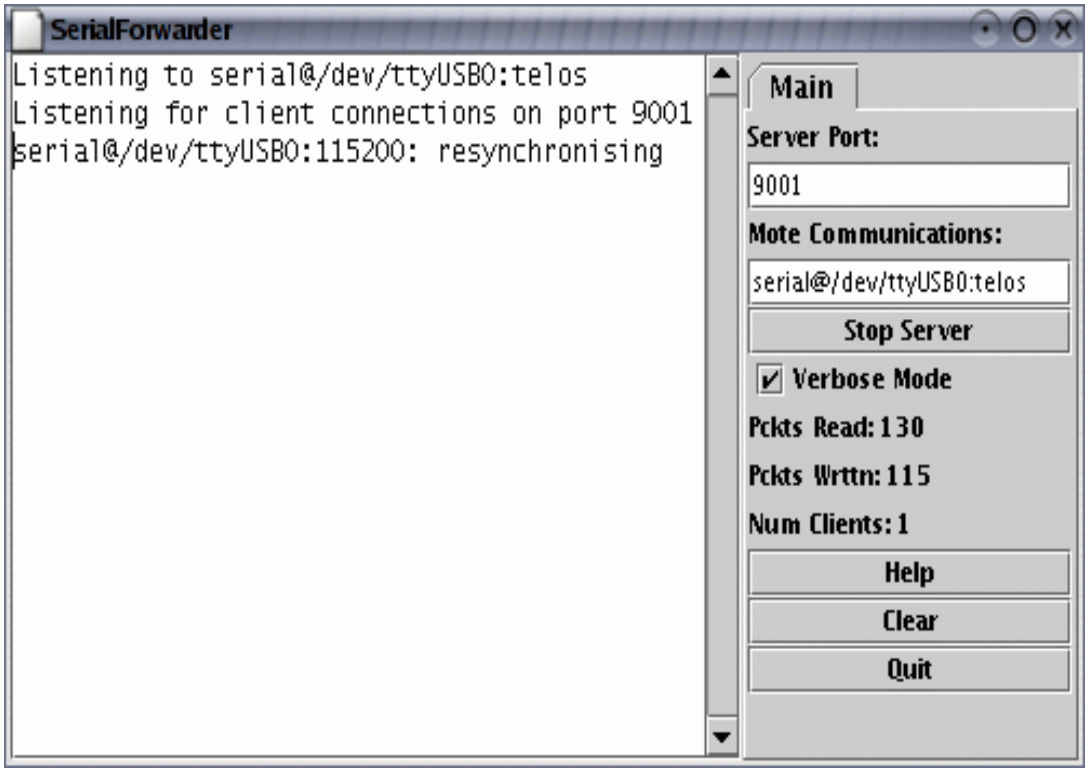
Ο Serial Forwarder είναι ένα πρόγραμμα που χρησιμοποιείται για να διαβάζει πακέτα από μια σειριακή θύρα και να τα προωθεί μέσα από μια TCP/IP σύνδεση, έτσι ώστε και άλλα προγράμματα να μπορούν να επικοινωνούν με το δίκτυο αισθητήρων μέσα από το δίκτυο αυτό. Το όρισμα –comm ορίζει στον SerialForwarder να επικοινωνήσει με τη θύρα COM0. Προσδιορίζει επίσης την προέλευση των πακέτων που προωθούνται μέσω του εργαλείου αυτού. Σε αντίθεση με τα περισσότερα προγράμματα, ο SerialForwarder δεν κάνει χρήση της μεταβλητής MOTECOM αλλά χρησιμοποιεί το όρισμα –comm για την πηγή προέλευσης του πακέτου.

Η παράμετρος ‘tmote’ ορίζει στον SerialForwarder να επικοινωνήσει με την ταχύτητα που υποστηρίζει η πλατφόρμα tmote, δηλαδή στην περίπτωσή αυτή με τον ρυθμό 57600 baud.

Ο SerialForwarder δεν απεικονίζει το ίδιο το πακέτο, αλλά ανανεώνει τον μετρητή πακέτων στην δεξιά γωνία του παραθύρου. Καθώς τρέχει, αναμένει για συνδέσεις πελάτη δικτύου σε μία δοσμένη θύρα (port) – 9001 είναι η προκαθορισμένη θύρα – και απλά προωθεί πακέτα **TinyOS** από τη σειριακή θύρα στη σύνδεση πελάτη και αντίστροφα. Εδώ αξίζει να αναφερθεί ότι, στον SerialForwarder, μπορούν να συνδεθούν ταυτόχρονα περισσότερες της μια

εφαρμογές και όλες θα λαμβάνουν ένα αντίγραφο του μηνύματος από το δίκτυο αισθητήρων.

Άξιο αναφοράς είναι το γεγονός ότι η υλοποίησης του Serial Forwarder στο **TinyOS**-2.x κρίνεται πολύ αναξιόπιστη καθώς λειτουργεί και χωρίς προφανή λόγο σχεδόν πάντα κολλούσε δυσχεραίνοντας την διαδικασία των μετρήσεων.



Εικόνα 54-Ο Serial Forwarder

Listen

Ακολούθως με την εντολή

```
$ java net.tinyos.tools.Listen
```

φαίνονται τα πακέτα κωδικοποιημένα σε παρόντα χρόνο να καταγράφονται στην οθόνη του υπολογιστή. Με την εντολή

```
$ java net.tinyos.tools.Listen > resp_Listen.txt
```

τα δεδομένα αποθηκεύονται σε ένα αρχείο κειμένου και με την κατάλληλη επεξεργασία εξάγονται οι μετρήσεις σε περιβάλλον Matlab με μια διαδικασία που θα αναλύσουμε στη συνέχεια.

00 FF FF FF FF 10 00 03 00 00 05 07 07 EB 06 0B 05 1F 07 E7 05 FF AC 4B
00 FF FF FF FF 10 00 03 00 00 05 13 07 E7 05 FB 05 23 07 E7 05 EF AC 4C
00 FF FF FF FF 10 00 03 00 00 05 17 07 E7 05 F7 05 23 07 DB 05 F7 AC 4D
00 FF FF FF FF 10 00 03 00 00 05 17 07 EB 05 FF 05 23 07 F3 05 FB AC 4E
00 FF FF FF FF 10 00 03 00 00 05 0B 07 EB 05 F7 05 1F 07 EB 05 F7 AC 4F
00 FF FF FF FF 10 00 03 00 00 05 0B 07 EB 05 F7 05 0B 07 DB 05 FB AC 4F
00 FF FF FF FF 10 00 03 00 00 04 EB 07 EF 05 FB 05 17 07 E7 05 FB AC 51
00 FF FF FF FF 10 00 03 00 00 04 EB 07 EF 05 FB 05 13 07 F7 05 F3 AC 51

Στη δομή αυτή του πακέτου έχουμε τα εξής πεδία

- Destination address (2 bytes)
- Link source address (2 bytes)
- Message length (1 byte)
- Group ID (1 byte)
- Active Message handler type (1 byte)
- Payload (up to 28 bytes):
    - source mote ID (2 bytes)
    - sample counter (2 bytes)

00.	FF	FF	FF	FF	10.	00.	03.	00.	00.
	destination add		source addre		Msg leng	Group II	handler	mote ID	

05.	07.	07.	EB	06.	0B	05.	1F	07.	E7	05.	FF	AC	4B
ADC		ADC		ADC		ADC		ADC		ADC		Timestamp	

Δομή πακέτου listen.

.

Να σημειώσουμε ότι τα δεδομένα στέλνονται από το mote σε big-Indian μορφή δηλαδή το

01 02 είναι 258*(256*1 + 2)

MIG (Message Interface Generator)

Το εργαλείο MIG δίνει τα δεδομένα με μια πιο ξεκάθαρη και ευανάγνωστη μορφή. Με την εντολή

```
java net.tinyos.tools.MsgReader MoistureSensors.Msg
```

έχουμε και πάλι τα πακέτα στην οθόνη του υπολογιστή και με την εντολή

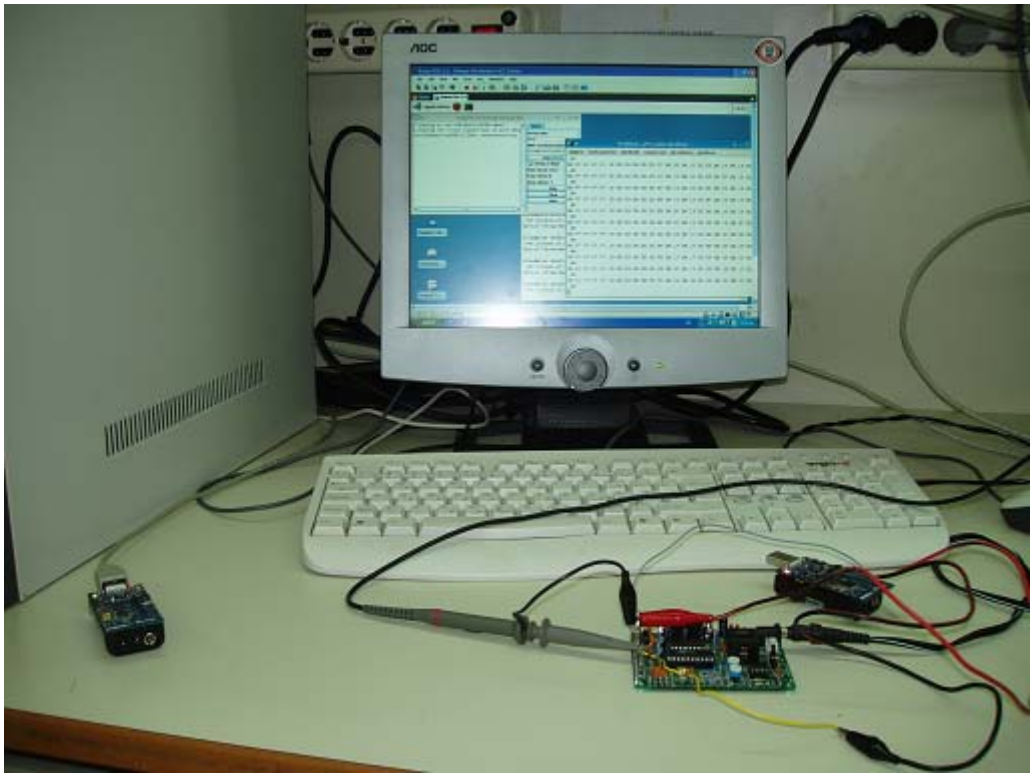
```
java net.tinyos.tools.MsgReader MoistureSensors.Msg>resp_MIG.txt
```

αποθηκεύονται σε αρχείο κειμένου για επεξεργασία.

```
1206131028096: Message <MoistureSensorsMsg>
  [nodeid=0x0]
  [adc00=0x54f]
  [adc01=0x8fb]
  [adc02=0x59b]
  [adc10=0x52f]
  [adc11=0x8f3]
  [adc12=0x597]
  [timestamp=0x2f8f]

1206131028196: Message <MoistureSensorsMsg>
  [nodeid=0x0]
  [adc00=0x55b]
  [adc01=0x8eb]
  [adc02=0x59b]
  [adc10=0x52f]
  [adc11=0x8eb]
  [adc12=0x59b]
  [timestamp=0x2f90]
```

Στην περίπτωση του μηνύματος MIG έχουμε στην πρώτη γραμμή τον αριθμό του μηνύματος και τον τύπο. Στη δεύτερη έχουμε την ταυτότητα του mote που το στέλνει και μετά ακολουθούν τα δεδομένα από το κάθε κανάλι ξεχωριστά. Στη τελευταία γραμμή του μηνύματος βρίσκετε η χρονοσφραγίδα του κάθε μηνύματος.

Να σημειωθεί ότι στον κώδικα του αρχείου MoistureSensorsMsg.class έγιναν οι κατάλληλες τροποποιήσεις ώστε το μήνυμα να περιέχει πληροφορία μόνο από 3 κανάλια τα ADC0 ,ADC1 και ADC2 και όχι και από τα 12 όπως ήταν αρχικά σχεδιασμένο .Ο κώδικας του εν λόγω αρχείου φαίνεται στο σύνολο του στο παράρτημα.

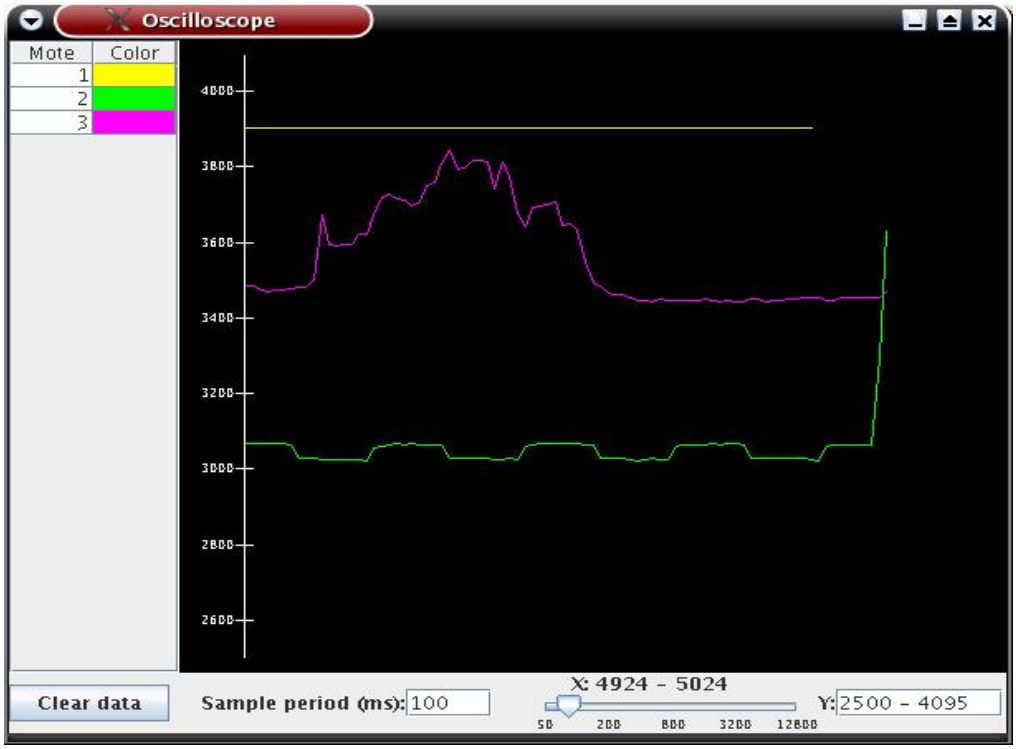


Εικόνα 55-Η πειραματική διάταξη.Φαίνονται το evaluation board συνδεδεμένο στο mote καθώς και το Basestation συνδεδεμένο στον υπολογιστή που καταγράφει δεδομένα μέσω του Serial Forwarder.

Oscilloscope application

Στην προσπάθεια να βρεθεί ο κατάλληλος κώδικας για ανάγνωση από τον αναλογικό σε ψηφιακό μετατροπέα με εξωτερικές εισόδους (κατ'επέκταση άξονες επιταχυνσιομέτρου) έγιναν πολλές και διάφορες μετατροπές στην εφαρμογή Oscilloscope η οποία εκτός από την λήψη μετρήσεων από ένα αισθητήρα και ασύρματη εκπομπή των μηνυμάτων στο δίκτυο μπορεί να υποστηρίξει το java GUI, δηλαδή το γραφικό περιβάλλον της Java και να επεκταθεί σε multihop δίκτυα .Επεμβαίνοντας στο HAL(hardware adaptive layer) καθορίστηκε το σήμα εισόδου και αντί του Demosensor δειγματοληπτήθηκε το κανάλι ADC0. Το πρόβλημα όμως με την εφαρμογή αυτή ήταν ότι δεν υποστηρίζει πολυκαναλική ανάγνωση και θα έπρεπε να χρησιμοποιήθούν 2 motes ένα για κάθε άξονα του επιταχυνσιομέτρου λύση που κρίθηκε ακατάλληλη αφού δεν έκανε σωστή διαχείριση των πόρων.

Στη συνέχεια η εφαρμογή Ecosensory και ο κώδικας ReadMoistureSensors κριθήκαν πιο κατάλληλα για τις ανάγκες της παρούσας εργασίας. Έγινε προσπάθεια να προσαρμοστεί το γραφικό μέρος της εφαρμογής Oscilloscope στην εφαρμογή Ecosensory αλλά δεν ολοκληρώθηκε καθώς η δομή των μηνυμάτων διαφέρει μεταξύ των 2 εφαρμογών.



Εικόνα 56-Το γραφικό μέρος της εφαρμογής Ooscilloscope.

Ασύρματα δίκτυα αισθητήρων – Επεξεργασία σήματος και αποτελέσματα

Όπως εξήγηθηκε αναλυτικά πιο πριν με την χρήση των κατάλληλων εργαλείων και εφαρμογών ληφθηκαν και αποθηκεύτηκαν τα μετρούμενα σήματα στον υπολογιστή.

Όλες οι μετρήσεις έγιναν στο εργαστήριο Κινητών Επικοινωνιών του ΕΜΠ. Μέσα στο περιβάλλον του εργαστηρίου υπήρχαν πολλές πηγές θορύβου και παρεμβολών. Διάφορες κεραίες που εκπέμπουν σε διάφορες ζώνες συχνοτήτων, μηχανήματα μετρησης πεδίου, αναλυτές φάσματος, παλμογράφοι, υπολογιστές και πολλά άλλα πιθανόν να επηρέαζαν το περιβάλλον των μετρήσεων. Για το λόγο αυτό είναι επιτακτική η ανάγκη κατάλληλης επεξεργασίας του σήματος το οποίο είναι σύνθετο και περιέχει συνιστώσες που οφείλονται σε κινήσεις του ασθενούς ,φτερνίσματα ,χασμουρητά ,μετατοπίσεις της πλατφόρμας αξιολόγησης τα οποία καταγράφονται από το επιταχυνσιόμετρο.

Επεξεργασία

Το σήμα της αναπνοής είναι σύνθετο και αποτελείται από δύο βασικές συνιστώσες. Μία χαμηλή 0- 4 Hz και μια πιο ψηλή 4- 40 Hz. Οι συχνότητες όμως αυτές αφορούν την αναπνοή ως βιοσήμα. Στην παρούσα εργασία αυτό που ενδιαφέρει είναι η ανίχνευση και η μέτρηση των αναπνοών που υπάρχουν στο δείγμα του σήματος και όχι συνιστώσες που μπορεί να προέρχονται από συσπάσεις του διαφράγματος ή των διαφόρων μυών της κοιλιακής χώρας και του στήθους.

Με γνώμονα το πιο πάνω σκεπτικό τα διάφορα σήματα πέρασαν από μια σειρά φιλτραρίσματος με σκοπό να καθαριστεί κατά το δυνατό ο θόρυβος και να τονιστεί όσο πιο καθαρά γίνεται η μορφή της αναπνοής σαν κύμα με ελάχιστα και μέγιστα που να αντιστοιχούν σε αναπνοές. Ύστερα από μια σειρά δοκιμών και πειραματισμών πάνω σε διαφόρων τύπων φίλτρα, βαθυπερατά ή υψιπερατά, με τη βοήθεια του εργαλείου Signal Processing
(sptool) του Matlab και σημάτων που πήραμε από την εφαρμογή μας, καταλήξαμε στην επιλογή του φίλτρου ‘κινούμενου μέσου όρου’ (moving average filter), ως του πιο αποτελεσματικού στην περίπτωσή μας. Και αυτό γιατί το φίλτρο αυτό αποδείχθηκε ιδιαίτερα αποτελεσματικό στην εξάλειψη των έντονων διακυμάνσεων του σήματος μας εξαιτίας του θορύβου, διαθέτοντας ταυτόχρονα και συμπεριφορά

βαθυπερατού FIR φίλτρου. Παρακάτω παρουσιάζουμε μια σύντομη θεωρία του συγκεκριμένου τύπου φίλτρου, καθώς και την υλοποίηση του στην εφαρμογή μας.

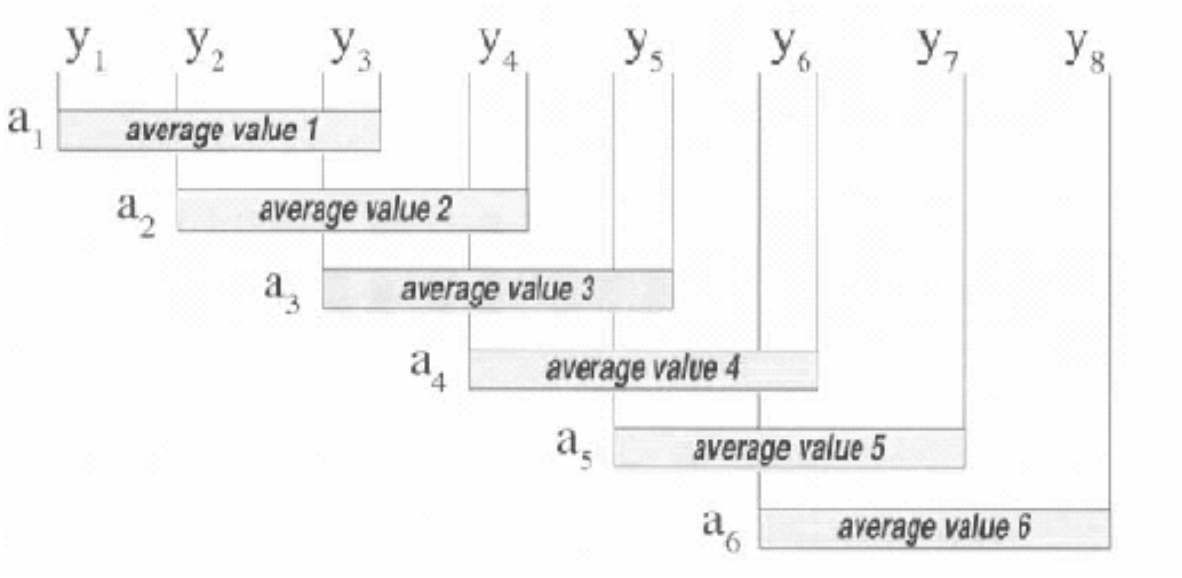
Το φίλτρο 'Κινούμενου Μέσου Όρου' (Moving Average Filter)

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

Θεωρία φίλτρου κινούμενου μέσου όρου

Ο αλγόριθμος του κινούμενου μέσου όρου επιτρέπει μεγάλη ευελιξία σε εφαρμογές φιλτραρίσματος κυματομορφών. Μπορεί να χρησιμοποιηθεί ως βαθυπερατό φίλτρο για να εξασθενήσει το θόρυβο που ενυπάρχει σε κυματομορφές διαφόρων τύπων, ή ως υψιπερατό φίλτρο που εξαφανίζει τη μετατόπιση βασικής γραμμής, λόγω παρεμβολής από άλλο υψίσυχνο σήμα. Η διαδικασία που χρησιμοποιεί ο αλγόριθμος για να καθοριστεί το μέγεθος του φιλτραρίσματος, περιλαμβάνει τη χρήση ενός παράγοντα εξομάλυνσης (παράθυρο s σημείων). Αυτός ο παράγοντας μπορεί να αυξηθεί ή να ελαττωθεί ανάλογα με τον αριθμό των πραγματικών τιμών της κυματομορφής ή των δειγμάτων που θα χρησιμοποιήσει ο αλγόριθμος. Κάθε περιοδική κυματομορφή μπορεί να θεωρηθεί σαν μια σειρά ή συλλογή από τιμές δεδομένων. Ο αλγόριθμος υπολογίζει τον κινούμενο μέσο όρο παίρνοντας 2 ή περισσότερες τιμές της κυματομορφής, προσθέτοντας τις, διαιρώντας το άθροισμά τους με το συνολικό αριθμό των προστιθέμενων τιμών, αντικαθιστώντας την πρώτη τιμή με το μέσο όρο που μόλις υπολογίστηκε και επαναλαμβάνοντας τα βήματα με τη δεύτερη, τρίτη κ.ο.κ. τιμή, μέχρι την τελευταία τιμή της κυματομορφής.

Το αποτέλεσμα είναι μια παραγόμενη κυματομορφή που συνίσταται από τις μέσες τιμές των δεδομένων και έχει τον ίδιο αριθμό σημείων με την αρχική κυματομορφή.



Εικόνα 57-Αρχή λειτουργίας του αλγόριθμου κινούμενου μέσου όρου.

Παράδειγμα εφαρμογής κινούμενου μέσου όρου με παράθυρο 3.

Η Εικόνα παρουσιάζει πώς εφαρμόζεται ο αλγόριθμος του κινούμενου μέσου όρου στα σημεία y μιας κυματομορφής. Με y σημειώνονται τα συνεχόμενα σημεία της κυματομορφής ώστε να φανεί πώς υπολογίζεται ο κινούμενος μέσος όρος. Στην περίπτωση αυτή, εφαρμόστηκε ο παράγοντας εξομάλυνσης τριών σημείων, που σημαίνει ότι 3 συνεχόμενα σημεία της αρχικής κυματομορφής προστέθηκαν, το άθροισμά τους διαιρέθηκε δια του 3 και το πηλίκο αυτό αποτέλεσε το πρώτο σημείο της παραγόμενης κυματομορφής. Η διαδικασία επαναλαμβάνεται για το δεύτερο, το τρίτο κ.ο.κ. σημείο της αρχικής κυματομορφής, μέχρι το τελευταίο σημείο της. Είναι αξιοσημείωτη η επικάλυψη που συμβαίνει κατά τον υπολογισμό του κινούμενου μέσου όρου. Αυτή η τεχνική της επικάλυψης, μαζί με μια ειδική μεταχείριση του αρχικού και τελικού σημείου, δίνει τον ίδιο αριθμό σημείων στην παραγόμενη κυματομορφή, με αυτόν της αρχικής.

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

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

όπου α = η μέση τιμή

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

Η ευελιξία του αλγορίθμου βασίζεται στο μεγάλος εύρος επιλογών που υπάρχουν για τον παράγοντα εξομάλυνσης. Ο παράγοντας αυτός καθορίζει πόσα πραγματικά σημεία θα ληφθούν υπόψη για την εξαγωγή του μέσου όρου. Κάθε θετικός παράγοντας εξομάλυνσης έχει ως αποτέλεσμα ένα βαθυπερατό φίλτρο ενώ κάθε αρνητικός, προσομοιώνει ένα υψιπερατό φίλτρο. Δεδομένης της απόλυτης τιμής του παράγοντα εξομάλυνσης, οι υψηλές τιμές δίνουν μεγαλύτερη εξομάλυνση στην παραγόμενη κυματομορφή ενώ οι χαμηλές τιμές παρέχουν μικρότερη εξομάλυνση.

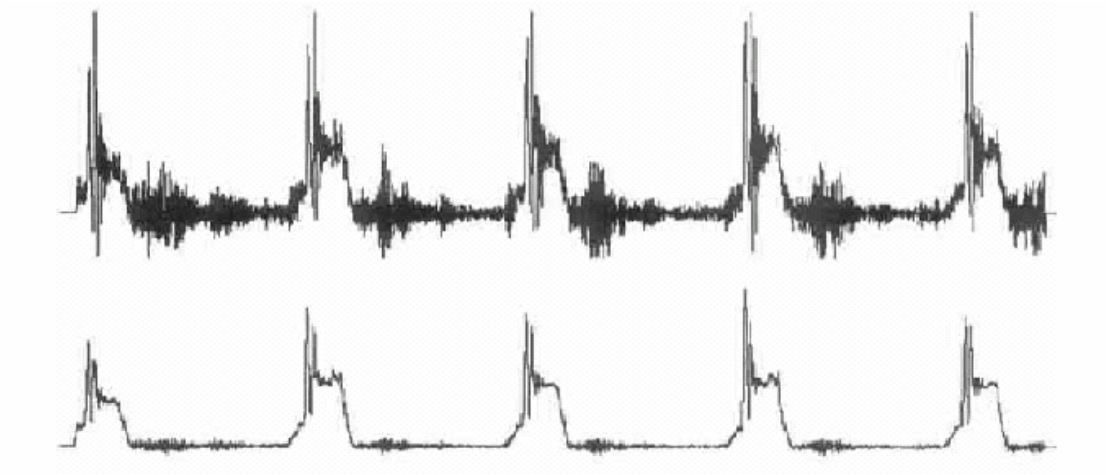
Εφαρμόζοντας τον κατάλληλο παράγοντα εξομάλυνσης, ο αλγόριθμος μπορεί να χρησιμοποιηθεί για την εξαγωγή της μέσης τιμής μιας περιοδικής κυματομορφής. Μεγαλύτεροι θετικοί παράγοντες εξομάλυνσης συνήθως εφαρμόζονται για την παραγωγή μεσαίων τιμών κυματομορφής.

Εφαρμόζοντας τον αλγόριθμο του κινούμενου μέσου όρου

Μια σημαντική ιδιότητα του αλγορίθμου είναι ότι, αν χρειάζεται, μπορεί να εφαρμοστεί πολλές φορές στην ίδια κυματομορφή, έτσι ώστε να παραχθεί το επιθυμητό αποτέλεσμα φιλτραρίσματος. Το αποτέλεσμα αυτό είναι υποκειμενικό. Το μόνο που μπορεί να κριθεί είναι αν ο αριθμός των σημείων που επιλέχθηκαν ήταν πολύ μεγάλος, πολύ μικρός ή ο ιδανικός. Η ευελιξία του αλγορίθμου επιτρέπει την προσαρμογή του παράγοντα εξομάλυνσης (παράθυρο) και την επανάληψη του αλγορίθμου όταν τα αποτελέσματα της πρώτης προσπάθειας δεν είναι ικανοποιητικά.

Μείωση θορύβου

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



Εικόνα 58-Παράδειγμα εφαρμογής του moving average.

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

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

Ανάγνωση δεδομένων

Η καταγραφή και αποθήκευση των δεδομένων στον υπολογιστή έγινε με τα Εργαλεία Listen και MIG όπου κάθε ένα δίνει μια διαφορετική δομή στα δεδομένα από την οποία πρέπει με τον κατάλληλο τρόπο να εξάχθεί η ζητούμενη πληροφορία, δηλαδή μια τιμή τάσης που να αντιστοιχεί στις εκάστοτε μεταβολές της επιτάχυνσης ώστε να διακρίνεται στο πεδίο του χρόνου την κυματομορφή της αναπνοής. Τα προγράμματα ανάγνωσης δεδομένων ονομάστηκαν **readListen3Channels.m** και **readMIG3Channels.m** και οι αντίστοιχοι κώδικες φαίνονται στο παράρτημα.

Listen

Πιο πίσω ανάλυθηκε η μορφή του μηνύματος Listen η οποία έχει ως εξής

00 FF FF FF FF 10 00 03 00 00 05 07 07 EB 06 0B 05 1F 07 E7 05 FF AC 4B
00 FF FF FF FF 10 00 03 00 00 05 13 07 E7 05 FB 05 23 07 E7 05 EF AC 4C
00 FF FF FF FF 10 00 03 00 00 05 17 07 E7 05 F7 05 23 07 DB 05 F7 AC 4D
00 FF FF FF FF 10 00 03 00 00 05 17 07 EB 05 FF 05 23 07 F3 05 FB AC 4E

Στη δομή αυτή του πακέτου έχουμε τα εξής πεδία

- Destination address (2 bytes)
- Link source address (2 bytes)
- Message length (1 byte)
- Group ID (1 byte)
- Active Message handler type (1 byte)
- Payload (up to 28 bytes):
    - source mote ID (2 bytes)
    - sample counter (2 bytes)

Ενδιαφέρει η τιμή που αντιστοιχεί σε κάθε κανάλι του mote δηλαδή οι τιμές που φαίνονται στη συνεχεία και το αντίστοιχο κανάλι.

05 07	**07 EB**	**06 0B**	**05 1F**	**07 E7**	**05 FF**
ADC0	ADC1	ADC2	ADC10	ADC11	ADC12

Οι τιμές αυτές είναι σε δεκαεξαδική μορφή . Με χρήση της εντολής **hex2dec(data)** στον κώδικα Matlab μετατρέπονται σε δεκαδικό. Ακολούθως η τιμή αυτή πολλαπλασιάζετε επί 3 και διαιρείται με 4096 για να προκύψει η ακριβής τιμή της τάσης όπως καταγράφεται από το επιταχυνσιόμετρο. Αυτό γίνεται γιατί ο A/D converter του μικροεπεξεργαστή MSP430 που χρησιμοποιείται στην εφαρμογή μας είναι 12-dits-από όπου προκύπτει και το 4096(2^12) – και μετατρέπει την τάση σύμφωνα με τον τύπο

$$DV_{cc} = \frac{ADCCounts}{4096} \times 2V_{ref}$$

Το Vref αναφέρεται στην τάση αναφοράς από όπου προκύπτει και το 3.

Σε προηγούμενη ενότητα εξήγηθηκε πως η εφαρμογή ReadMoistureSensors παίρνει δείγματα από τις θύρες ADC0 έως ADC5 και γεμίζει σειριακά 2 τράπεζες

δεδομένων στην ίδια περίοδο. Υπάρχουν δηλαδή δυο μετρήσεις από κάθε κανάλι στην ίδια περίοδο ADC0-ADC10 , ADC1 –ADC11, ADC2-ADC12. Για τον λόγω αυτό στον κώδικα τοποθετούνται διαδοχικά τα δείγματα με την ίδια σφραγίδα χρονισμού που αντιστοιχούν στο ίδιο κανάλι και κατ'επέκταση στον ίδιο άξονα του επιταχυνσιόμετρου. Έτσι τα δεδομένα τοποθετούνται σε ένα πίνακα για κάθε άξονα και αναπαρισταναι γραφικά , φιλτράρονται και επεξεργαζονται αναλόγως.

Η πρόσβαση στο αρχείο κειμένου όπου αποθήκευτηκε η κάθε μέτρηση γίνεται με την εντολή **fopen()** αφού προσπελάστεί το κατάλληλο αρχείο στο μονοπάτι(path) που έχει ορίστει.

MIG

Η βάση της λογικής του κώδικα για τα μηνύματα MIG παραμένει η ίδια με του Listen .Με τον ίδιο τρόπο προσπελαύνονται τα δεδομένα στα αρχεία κειμένου από το κατάλληλο μονοπάτι, και πάλι τοποθετούνται τα δεδομένα που αφορούν την ίδια θύρα με την ίδια χρονοσφραγίδα διαδοχικά. Η διαφορά έγκειται στην μορφή του μηνύματος.

```
1206131028096: Message <MoistureSensorsMsg>
  [nodeid=0x0]
  [adc00=0x54f]
  [adc01=0x8fb]
  [adc02=0x59b]
  [adc10=0x52f]
  [adc11=0x8f3]
  [adc12=0x597]
  [timestamp=0x2f8f]
```

Παρατηρούμε οτι τα δεδομένα στο μήνυμα είναι σε δεκαεξαδική μορφή και πρέπει να τα μετατραπούν σε δεκαδική με την εντολή **hex2dec(data).** Στη συνέχεια και πάλι για να βρούμε την ακριβής τιμή της τάσης πολλαπλασιάζονται με 3 και διαιρούνται με 4096. Με την ίδια διαδικασία που ακολουθήσαμε στο Listen τα δεδομένα τοποθετούνται σε πίνακα από όπου μπορούν να παρασταθούν γραφικά και να τύχουν της κατάλληλης επεξεργασίας.

Επεξεργασία

Σκοπός της εργασίας αυτής είναι η ανίχνευση και καταμέτρηση των αναπνοών που καταγράφοντα από το επιταχυνσιόμετρο μέσω του ασύρματου δικτύου αισθητήρων. Προς αυτή την κατεύθυνση χρησιμοποιήθηκαν δύο μεθόδοι που στη συνέχεια τα αποτελέσματα τους θα συγκριθούν. Και στις 2 μεθόδους τα σηματα που ορίστικαν ως είσοδοι είχαν φιλτραριστεί με αρκετά μεγάλη αυστηρότητα με το moving_average.

Πρώτη μέθοδος ανίχνευσης αναπνοών

Η υλοποίηση του αλγόριθμου έγινε σε περιβάλλον Matlab. Συγκεκριμένα υλοποιήθηκε η συνάρτηση *resp_counter(decvalue_x,decvalue_y)* με ορίσματα τις τιμές των δειγμάτων του φιλτραρισμένου σήματος (decvalue_χ,decvalue_y) και τις αντίστοιχες τιμές του χρόνου στις οποίες λήφθηκαν τα δείγματα αυτά. Η έξοδος της συνάρτησης μας δίνει τον αριθμό των συμπλεγμάτων RESP και τον αναπνευστικό ρυθμό. Ο αλγόριθμος έχει ως εξής:

- Αρχικά, πραγματοποιεί μερικούς στατιστικούς υπολογισμούς πάνω στο σήμα ώστε να προκύψει κάποια τιμή κατωφλίου την οποία και θα χρησιμοποιήσει σαν κριτήριο για την επιλογή των πιθανών τιμών που αποτελούν το σύμπλεγμα RESP. Έτσι υπολογίζει τη μέση τιμή (mean value) και την τυπική απόκλιση του σήματος (stdev). Το κατώφλι που θέτει για το σήμα μας είναι ίσο με:

```
mean1 = mean(decvalue_x);
n = length(decvalue_x);
stdev = sqrt(sum((decvalue_x-mean1).^2/n));
y_up = mean1 + 0.5*stdev;
```

Θεωρούμε σαν πιθανά RESP complexes οποιαδήποτε τιμή του σήματος μας ξεπερνά το παραπάνω κατώφλι. Η υλοποίηση αυτή έγινε, ώστε ο αλγόριθμος να υπολογίζει για κάθε σήμα το αντίστοιχο κατώφλι με βάση τα στατιστικά μεγέθη του, χωρίς να απαιτείται να εισάγωγή κάθε φορά της τιμής του κατωφλίου.

```
possibleRESP = find(decvalue_x > y_up);
```

- Οι τιμές που ικανοποιούν το παραπάνω κριτήριο και βρίσκονται σε πιθανά συμπλέγματα RESP, αποθηκεύονται σε ένα πίνακα possibleRESP. Στον πίνακα αυτό δεν αποθηκεύονται οι ίδιες οι τιμές, αλλά οι δείκτες των τιμών αυτών, που δείχνουν στην αντίστοιχη θέση του πίνακα decvalue_χ στην οποία έχουν αποθηκευτεί. Στον πίνακα decvalue_χ, όπως αναφέραμε και προηγουμένως, έχουν αποθηκευτεί όλες οι τιμές τάσης των δειγμάτων μας. Έτσι, στην περίπτωση που θέλουμε να αναφερθούμε στις πραγματικές τιμές τάσης, πρέπει να αναφερθούμε ως decvalue_χ(possibleRESP(i)). Το ίδιο πρέπει να κάνουμε αν θέλουμε να αναφερθούμε στην αντίστοιχη χρονική στιγμή που έγινε η λήψη του συγκεκριμένου δείγματος (τιμή τάσης). Δηλαδή πρέπει να καλέσουμε την xdata(possibleRESP(i)) , όπου xdata ο πίνακας με τις χρονικές στιγμές λήψης όλων των δειγμάτων.
- Από τις τιμές που έχουν αποθηκευτεί σαν πιθανές να βρίσκονται μέσα σε σύμπλεγμα RESP πρέπει να διαχωρίσουμε ποιες από αυτές βρίσκονται μέσα στο ίδιο σύμπλεγμα RESP ή δεν βρίσκονται καθόλου, γιατί απλά μπορεί να αποτελούν κορυφές από τη διακύμανση του σήματος ή από το θόρυβο, και οι οποίες δεν κατάφεραν να αποκοπούν με βάση το κατώφλι που ορίσαμε. Θεωρούμε ότι εάν βρισκόμαστε σε ένα σύμπλεγμα RESP, τότε οι τιμές possibleRESP(i) θα έχουν σίγουρα συνεχόμενη αρίθμηση.

```
while(index <= length(possibleRESP))

   temp = index;
   i=index;
   while (i+1 <=length(possibleRESP) && abs(possibleRESP(i)-
possibleRESP(i+1))==1)
      %temp = index;
      i = i+1;
   end
   index = i+1;
```

Αυτό εξηγείται από το γεγονός ότι, κατά τη διάρκεια της σειριακής αναζήτησης, εάν κάποια τιμή που βρίσκεται μέσα στο σύμπλεγμα RESP και συγκεκριμένα το κύμα του RESP, βρεθεί πάνω από το κατώφλι που έχουμε ορίσει, τότε το δείγμα (ο δείκτης που δείχνει στην αντίστοιχη θέση του πίνακα των δειγμάτων) αυτόματα αποθηκεύεται στον πίνακα possibleRESP και τα επόμενα

δείγματα, εφόσον ανήκουν και αυτά στο κύμα RESP, θα βρίσκονται σίγουρα πάνω από το κατώφλι και θα αποθηκεύονται κατά σειρά στον πίνακα αυτόν. Αυτό θα συνεχιστεί μέχρι το επόμενο δείγμα να βρεθεί πάλι κάτω από το κατώφλι. Οποιαδήποτε τιμή, η οποία θα ξεπερνά το κατώφλι, εμφανιστεί ξανά κατά την αναζήτηση, τότε είτε θα ανήκει στο επόμενο σύμπλεγμα RESP είτε θα ανήκει σε αιχμή θορύβου που ξεπέρασε το κατώφλι. Για να ανήκει στο επόμενο σύμπλεγμα RESP πρέπει πάλι να ισχύει αυτό που περιγράψαμε και πρίν. Δηλαδή, τελικά, με αυτόν τον τρόπο διακρίνονται τα πραγματικά συμπλέγματα RESP που έχουν αποθηκευτεί μέσα στον πίνακα possibleRESP. Έτσι, στον κώδικά μας, αυτό που γίνεται είναι η αναζήτηση και οριοθέτηση των πραγματικών συμπλεγμάτων RESP. Κάθε φορά που εντοπίζεται ένα πεδίο τιμών που πιθανόν να ορίζει σύμπλεγμα RESP τότε αυτό οριοθετείται με δείκτη την πρώτη και την τελευταία τιμή του δείγματος στον πίνακα possibleRESP (temp, index-1) Εάν η διαφορά μεταξύ των δυο αυτών δεικτών είναι μεγαλύτερη του 20 τότε θεωρούμε ότι έχουμε πραγματικά σε αυτή την περιοχή μια αναπνοή, οπότε αυξάνεται και ο μετρητής RESP_count κατά ένα.

```
if (index - temp >20)
      resp_count = resp_count + 1;
      temp_max = 0;
```

Η επιλογή του αριθμού 20 προέκυψε ύστερα από δοκιμές και θεωρήθηκε ότι είχε τα καλυτερα αποτελέσματα, θεωρώντας ότι υπήρχε περίπτωση μια συνιστώσα θορύβου που είχε καταφέρει να περάσει το κατώφλι θα μπορούσε να είχε στη σειρά περισσότερα από ένα δείγματα τα οποία θα ήταν και αυτά πάνω από το κατώφλι, οπότε στην περίπτωση αυτή θα είχαμε λανθασμένη εκτίμηση συμπλέγματος RESP. Για τα σήματα που εξετάσαμε κρίθηκε ότι η επιλογή του αριθμού αυτού ήταν επαρκής, αφού έδωσε σωστά αποτελέσματα. Βέβαια σε περιπτώσεις άλλων σημάτων με περισσότερο θόρυβο ή εφόσον έχει γίνει λήψη με μεγαλύτερο ρυθμό δειγματοληψίας από αυτόν που επιλέξαμε εμείς (20Hz) ο αριθμός αυτός καλό θα ήταν να είναι μεγαλύτερος.

- Στη συνέχεια, για κάθε σωστά οριοθετημένο σύμπλεγμα RESP, βρίσκει τη μέγιστη τιμή από τις τιμές του πίνακα decvalue_χ που αντιστοιχούν στους αποθηκευμένους δείκτες στον πίνακα possibleRESP. Η μέγιστη αυτή τιμή αποτελεί την κορυφή του κύματος στο σύμπλεγμα RESP και η χρονική στιγμή

στην οποία παρουσιάζεται βρίσκεται μέσω του xdata για το συγκεκριμένο δείγμα. Οι μέγιστες τιμές τάσης και οι αντίστοιχοι χρόνοι στους οποίους εμφανίζονται αποθηκεύονται στους πίνακες peak_values και peak_time_values.

```
peak_values = [peak_values max(decvalue_x(possibleRESP(temp:index-1)))];
peak_time_values = [peak_time_values possibleRESP(index_max)];
```

Δεύτερη μέθοδος ανίχνευσης αναπνοών

Στη μέθοδο αυτή χρησιμοποιήθηκε ο κλασικός τρόπος εντοπισμού των ακροτάτων μιας συνάρτησης μέσω της παραγώγου της. Βασιζόμενοι στο γεγονός ότι σε μια περίοδο υπάρχουν 20 δείγματα το φιλτραρισμένο σήμα χωρίζεται σε παράθυρα των 100 δειγμάτων.

```
start = 1;
end_index =100;
max_value_x = [];
Index_max_x = [];
 for i = 1:fix(length(filtered_x)/100)
    [max1 Index] = max(filtered_x(start:end_index));
    max_value_x = [max_value_x max1];
    Index_max_x = [Index_max_x ((i-1)*100+Index)];
    start = end_index+1;
    end_index = end_index + 100;
 end
```

Το κομμάτι αυτό του κώδικα χωρίζει το φιλτραρισμένο πια σήμα εν προκειμένων του X άξονα σε παράθυρα των 100 δειγμάτων υπολογίζοντας παράλληλα το μέγιστο δείγμα σε κάθε παράθυρο.

```
diff_fltx = diff(filtered_x);
Υπολογίζω την παράγωγο του φιλτραρισμένου σήματος.
while(j<=length(Index_max_x))
    if ((j+1)>length(Index_max_x))
        break;
    else
        if (Index_max_x(j+1)-Index_max_x(j)) <=60;
            counter=counter+1;
```

```
        j=j+1;
        index_array_x = [index_array_x Index_max_x(j)];
    else
        if  diff_fltx(Index_max_x(j))==0;
            counter=counter+1;
            index_array_x = [index_array_x Index_max_x(j)];
```

Γίνεται η υποθέση ότι σε κάθε παράθυρο υπάρχει μία αναπνοή που αντιστοιχεί σε ένα τοπικό μέγιστο άρα στο δείγμα με την μέγιστη τιμή μέσα στο παράθυρο και με μια σειρά από κριτήρια γίνεται προσπάθεια να επαληθευτεί. Εάν το δείγμα αυτό με την μέγιστη τιμή έχει και μηδενική πρώτη παράγωγο τότε αυτόματα πρόκειται για μια αναπνοή.

Αν η πρώτη παράγωγος του δείγματος είναι αρνητική και ταυτόχρονα του αμέσως προηγούμενου δείγματος θετική τότε πρόκειται για ακρότατο και αναπνοή.

```
else
        if diff_fltx(Index_max_x(j))<0 &              diff_fltx(Index_max_x(j)-1)>0
                counter=counter+1;
                index_array_x = [index_array_x Index_max_x(j)];
```

Αν η πρώτη παράγωγος του αμέσως προηγούμενου δείγματος είναι θετική και ταυτόχρονα των 2 επόμενων δειγμάτων αρνητική τότε είναι αναπνοή.

```
else
                if  diff_fltx(Index_max_x(j)-1)>0  &  diff_fltx(Index_max_x(j)+1)<=0  &
diff_fltx(Index_max_x(j)+2)<0;
                    counter=counter+1;
                    index_array_x = [index_array_x Index_max_x(j)];
                end
```

Ο κώδικας όπως διακρίνεται ολοκληρωμένος στο παράρτημα παρουσιάζει επί της οθόνης τα γραφήματα τόσο του αρχικού σήματος σε σχέση με το φιλτραρισμένο και τοποθετεί με τις μετρούμενες από τον αλγόριθμο αναπνοές.

```
disp('Number of respirations at Y axis =');disp(counter);
disp('-------------------------------------------------------------------------------------------
-----');
subplot(4,1,1);plot(X);title('X');hold
on;plot(index_array_x,filtered_x(index_array_x),'r*');
subplot(4,1,2);plot(filtered_x);title('filtered_x');hold
on;plot(index_array_x,filtered_x(index_array_x),'r*');
```

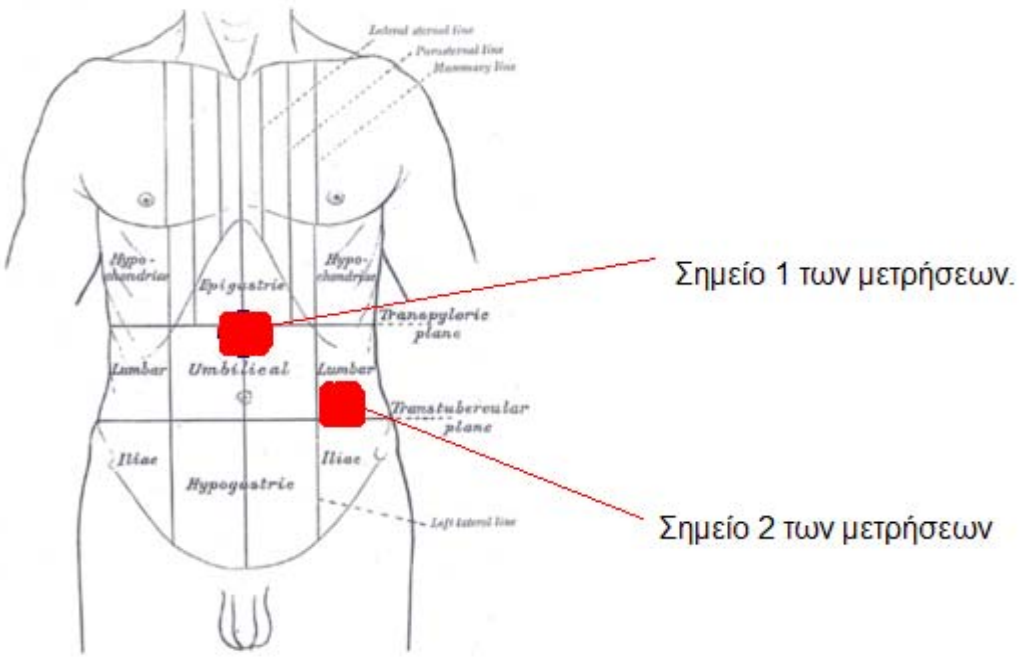
```
subplot(4,1,3);plot(Y);title('Y');hold
on;plot(index_array_y,filtered_y(index_array_y),'r*');
subplot(4,1,4);plot(filtered_y);title('filtered_y');hold
on;plot(index_array_y,filtered_y(index_array_y),'r*');
```

Κεφάλαιο 5

Μετρήσεις και Παρουσίαση αποτελεσμάτων

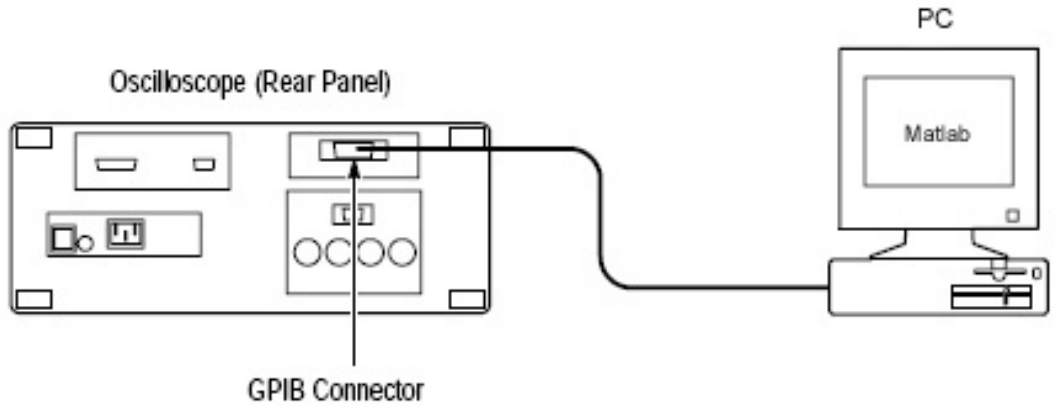
Το επόμενο βήμα είναι μια σειρά μετρήσεων για αξιολόγηση της διάταξης και έλεγχο των μετρήσεων με παλμογράφο. Κατά τη διαδικασία αυτή τροφοδοτήθηκαν με γνωστή τάση Dc ή ημιτονική σταθερής συχνότητας οι θύρες ADC0 και ADC1 του mote παίρνοντας δείγματα στον υπολογιστή από το Basestation μέσω του Serial Forwarder . Εν συνεχεία στο mote συνδέθηκε και η πλατφόρμα αξιολόγησης με το επιταχυνσιόμετρο του οποίου τις εξόδους μετρούσαμε ταυτόχρονα και στον παλμογράφο.

Οι μετρήσεις της αναπνοής έγιναν σε δύο διαφορετικά σημεία της κοιλιακής χώρας και του θώρακα. Μερικές μετρήσεις έγιναν με συχνότητα δειγματοληψίας 50 Hz αλλά ως επί τω πλείστον με συχνότητα 20 Hz. Κατά τη διάρκεια των μετρήσεων ταυτόχρονα με το σύστημα καταγραφής αναπνοής που αναπτύχθηκε μετρούνταν και πόσες αναπνοές εκτελούνταν στο χρονικό διάστημα της μέτρησης.



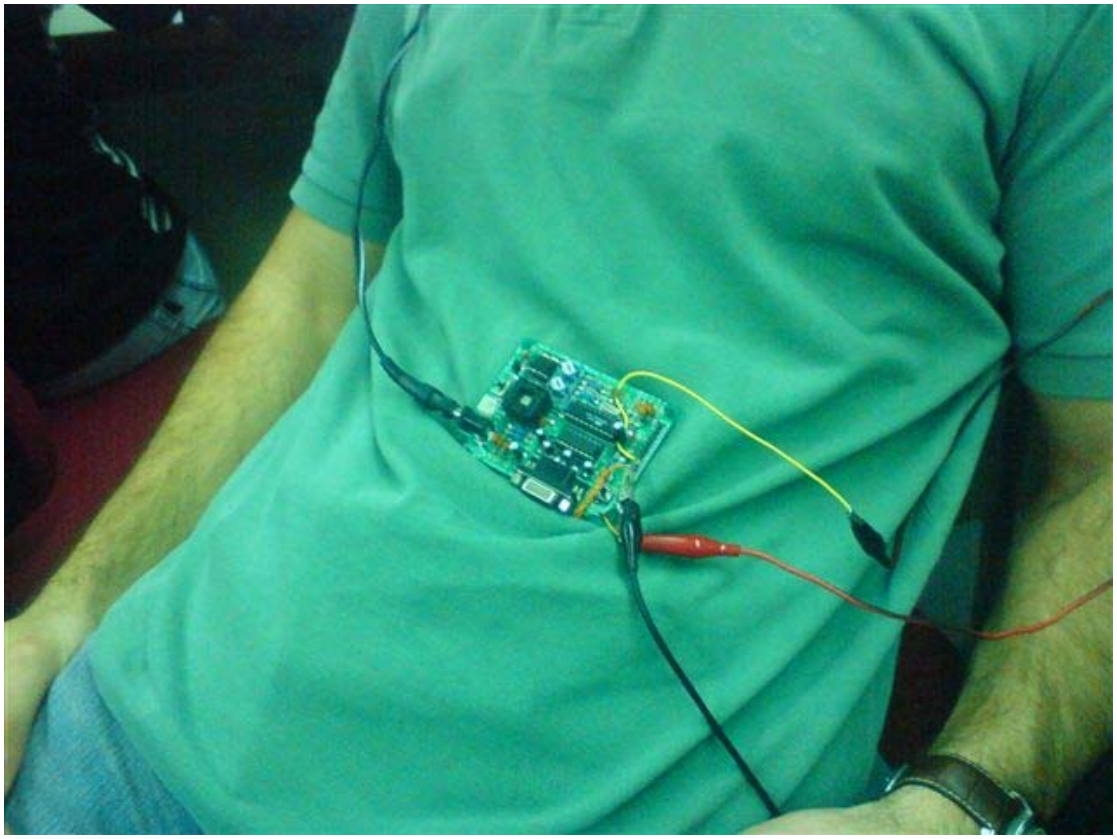
Εικόνα 59-Τα σημεία του σώματος στα οποία τοποθετήσαμε το επιταχυνσιόμετρο κατά την διεξαγωγή των μετρήσεων.

Στη πρώτη σειρά μετρήσεων ταυτόχρονα με τη διάταξη μας μέσω του ασύρματου δικτύου αισθητήρων μετρήθηκε και η έξοδος του επιταχυνσιομέτρου απευθείας στον υπολογιστή στο περιβάλλον Matlab μέσω του GPIB του ψηφιακού παλμογράφου.



Εικόνα 60-Σύνδεση του παλμογράφου με τον υπολογιστή.

Σαν συμπληρωματικές μετρήσεις έγιναν και μερικές μετρήσεις άπνοιας .Στη συνέχεια φαίνονται αναλυτικά οι γραφικές αναπαραστάσεις των μετρούμενων σημάτων και των φιλτραρισμένων με το moving_average όπως και η μέτρηση των αναπνοών όπως εκτελείται από τον κάθε αλγόριθμο ξεχωριστά.

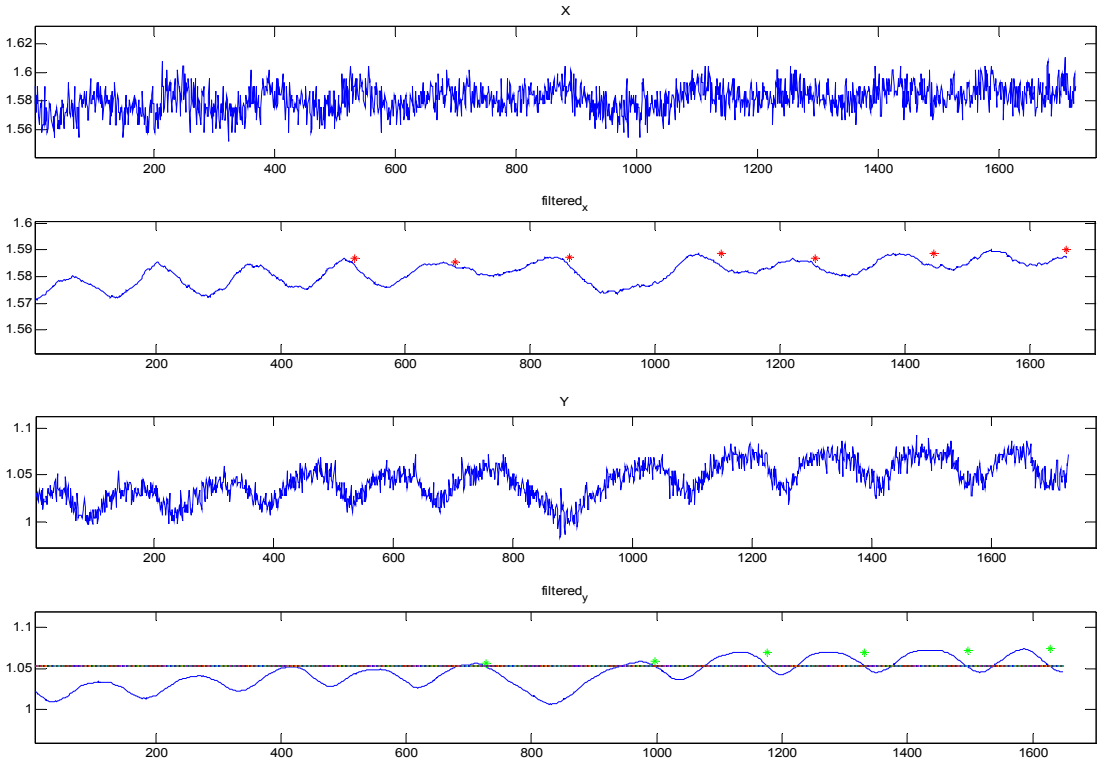


Εικόνα 61-Μέτρηση της αναπνοής με επιταχυνσιόμετρο στη θέση 1.

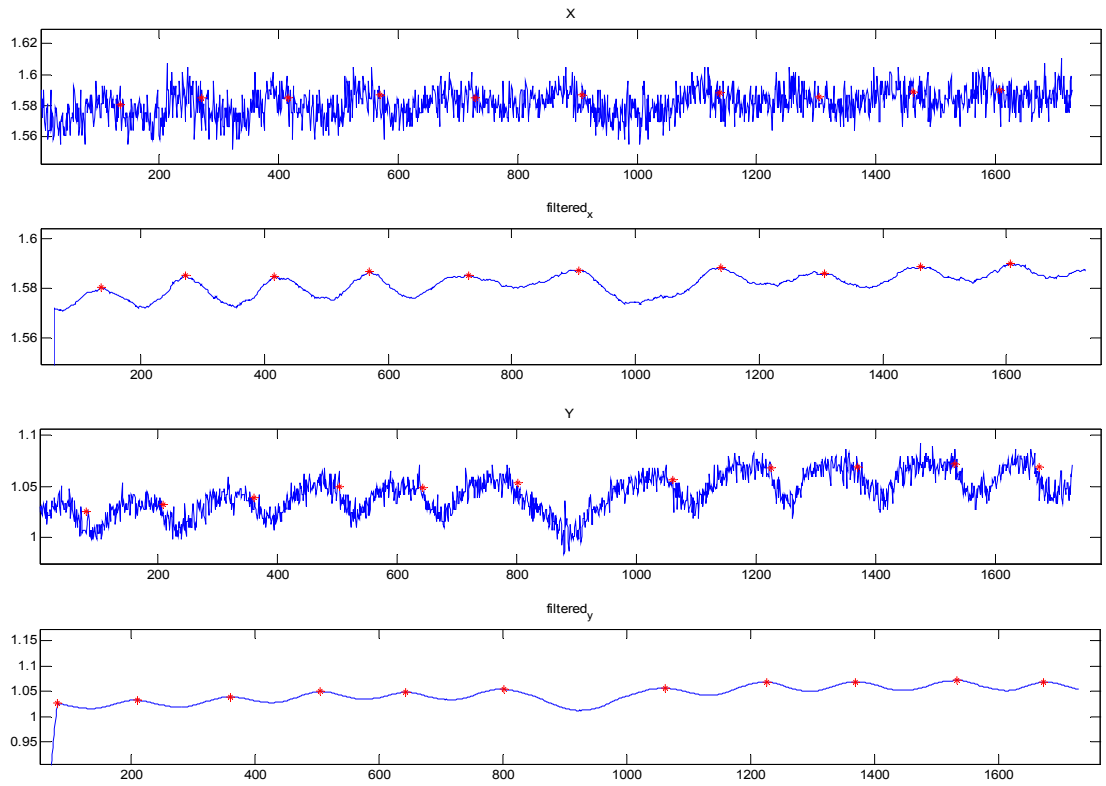
Πρώτη σειρά μετρήσεων .

Μετρήσεις με το εργαλείο Listen.

Η μέτρηση resp4_listen.txt .φαίνεται ο άξονας Χ όπως καταγράφεται από τη διάταξη και από κάτω το φιλτραρισμένο με τις αναπνοές όπως ανιχνεύονται από τη μέθοδο 1 και από κάτω ο άξονας Υ. Το επόμενο γράφημα απεικονίζει το ίδιο σήμα και τις μετρούμενες αναπνοές από τη μέθοδο 2. Το σήμα μετρήθηκε με συχνότητα δειγματοληψίας 20 HZ. Το σήμα περιέχει 11 αναπνοές. Όπως φαίνεται στο πιο κάτω γράφημα η πρώτη μέθοδος χάνει τις πρώτες 3 αναπνοές στον άξονα των Χ και τις πρώτες 4 στον άξονα Υ αφού βρίσκονται κάτω από το decvalue που ορίζει ο αλγόριθμος. Αντίθετα η δεύτερη μέθοδος φαίνεται να δουλεύει πιο καλά. Αξιοσημείωτο είναι ότι κατά την 6 αναπνοή ο ασθενής χασμουρήθηκε.



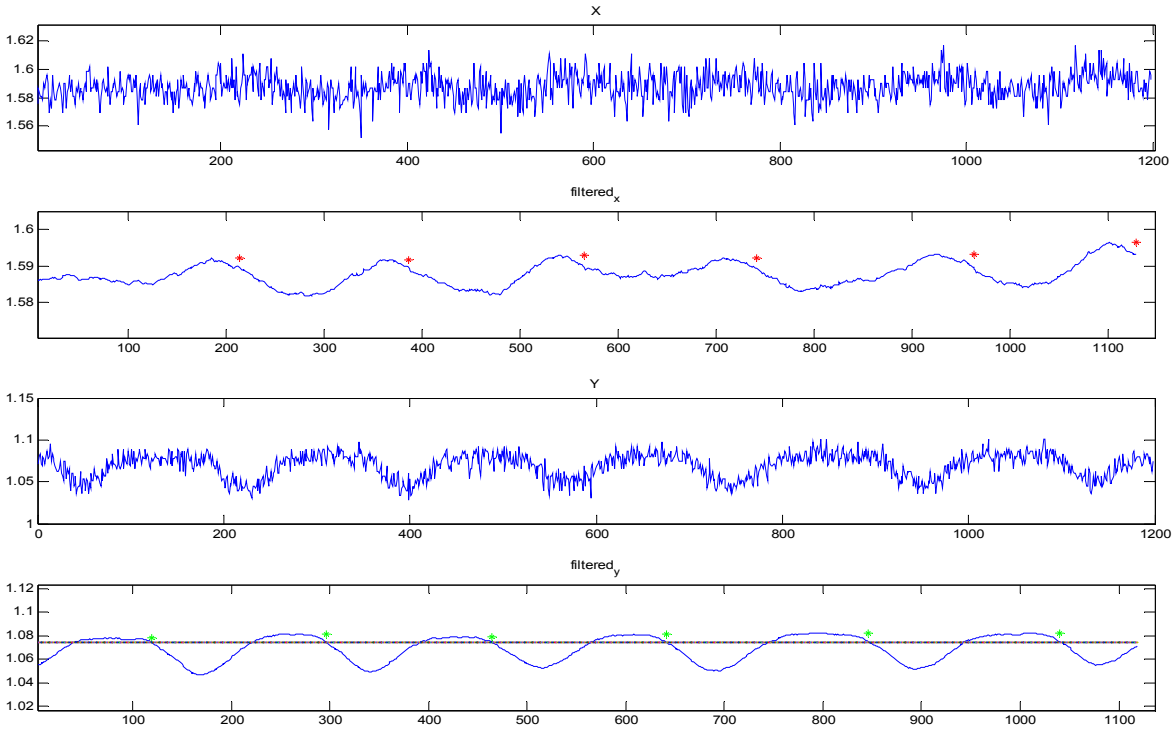
Εικόνα 62-Σήμα resp4_listen.txt.Πάνω παρατηρούμε αφιλτράριστο το σήμα του άξονα X και αμέσως από κάτω το φιλτραρισμένο όπου τα κόκκινα αστεράκια δηλώνουν τις αναπνοές που ανίχνευσε η πρώτη μέθοδος επεξεργασίας. Ακολουθεί το σήμα για τον άξονα Y.



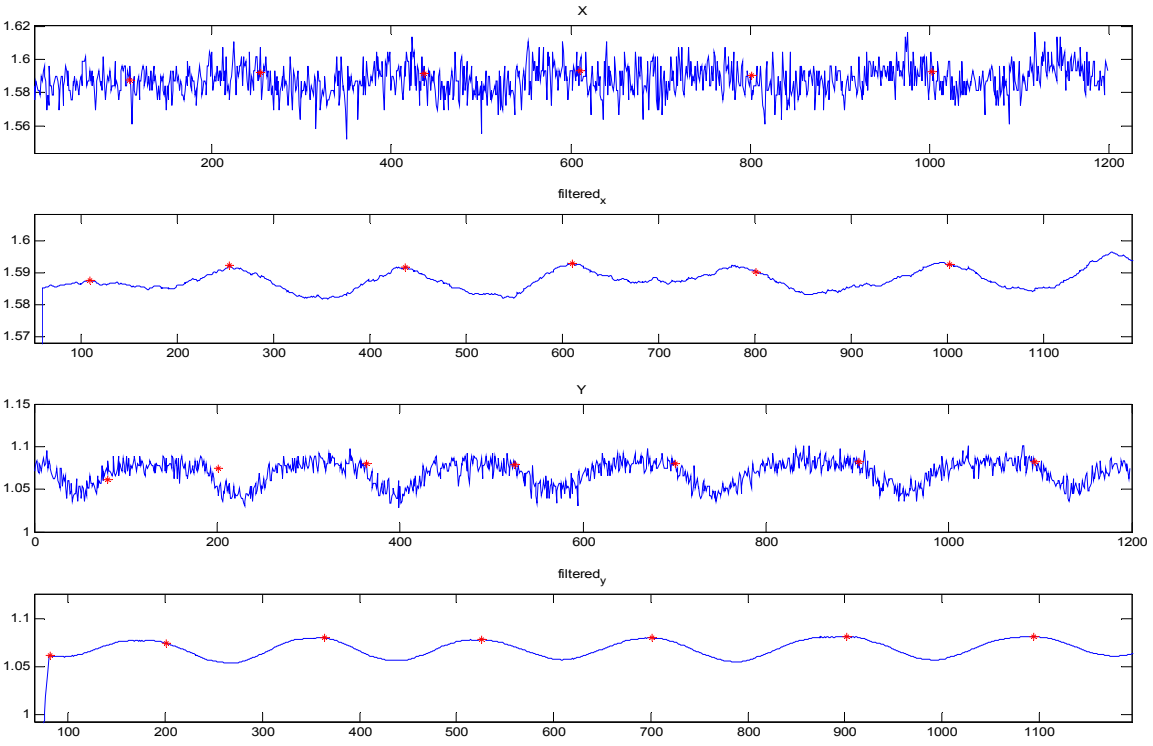
Εικόνα 63- Σήμα resp4_listen.txt.Εδώ εφαρμόζεται η μέθοδος 2 για την ανίχνευση των αναπνοών. Τα πρώτα 2 σήματα αφορούν το αφιλτράριστο και το φιλτραρισμένο άξονα Χ και τα άλλα 2 τον άξονα Υ. Οι κόκκινοι αστερίσκοι αντιστοιχούν στις αναπνοές που εντόπισε η μέθοδος.

.

Μέτρηση resp6_listen.txt με συχνότητα δειγματοληψίας 50 Hz και εφαρμογή των 2 μεθόδων διαδοχικά. Στο σήμα αυτό περιέχονται 7 αναπνοές. Το σήμα είναι αρκετά στρωτό και οι δυο μέθοδοι δουλεύουν αρκετά καλά. Όσον αφορά την πρώτη μέθοδο παρατηρούμε ότι όλες οι κορυφές του σήματος(αναπνοές) βρίσκονται πάνω από το deckvalue.

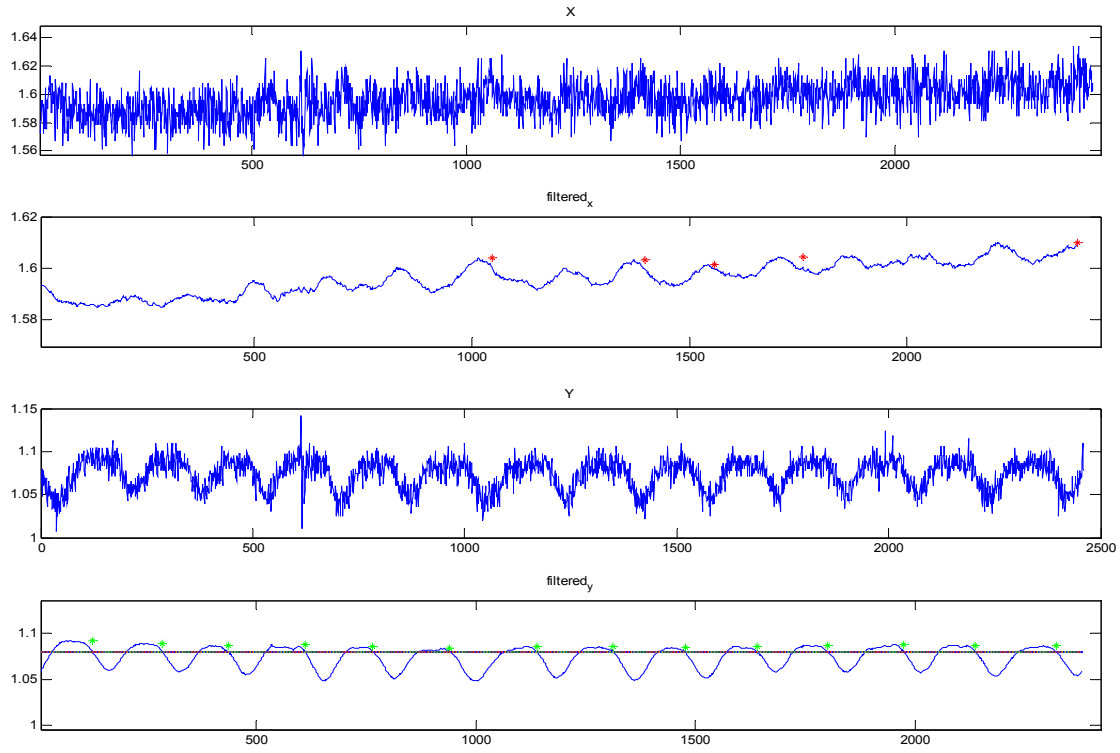


Εικόνα 64-Σήμα resp6_listen.txt.Εφαρμογή της πρώτης μεθόδου στους δύο άξονες X και Y.

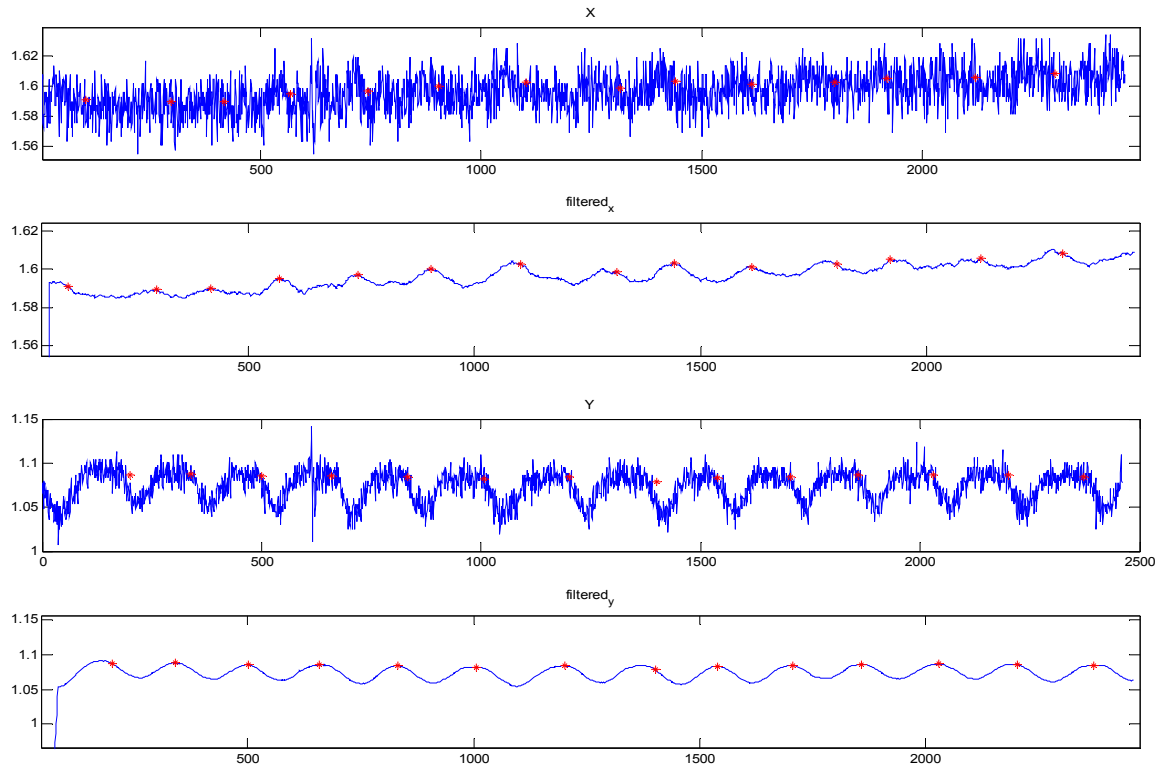


Εικόνα 65-Σήμα resp6_listen.txt.Εφαρμογή της δεύτερης μεθόδου και στους 2 άξονες.

Μέτρηση resp11_listen.txt με συχνότητα δειγματοληψίας 20Hz και εφαρμογή των 2 μεθόδων διαδοχικά. Στο σήμα αυτό περιέχονται 15 αναπνοές. Παρατηρούμε ότι το σήμα του άξονα X δεν είναι ιδιαίτερα ξεκάθαρο και ότι παρουσιάζει μιαν κλίση προς τα πάνω. Αυτό οφείλεται σε πιθανό γλίστρημα της πλατφόρμας αξιολόγησης κατά τη διάρκεια της διαδικασίας . Όπως διακρίνουμε το γλίστρημα αυτό δεν επηρέασε τον άξονα Y και η μέθοδος έδωσε καλά αποτελέσματα. Όσον αφορά τη δεύτερη μέθοδο φαίνεται να μην επηρεάζεται ιδιαίτερα από την κακή ποιότητα του σήματος στον άξονα X ούτε και από την θετική κλίση του σήματος δίνει αρκετά καλά αποτελέσματα κα στους 2 άξονες.



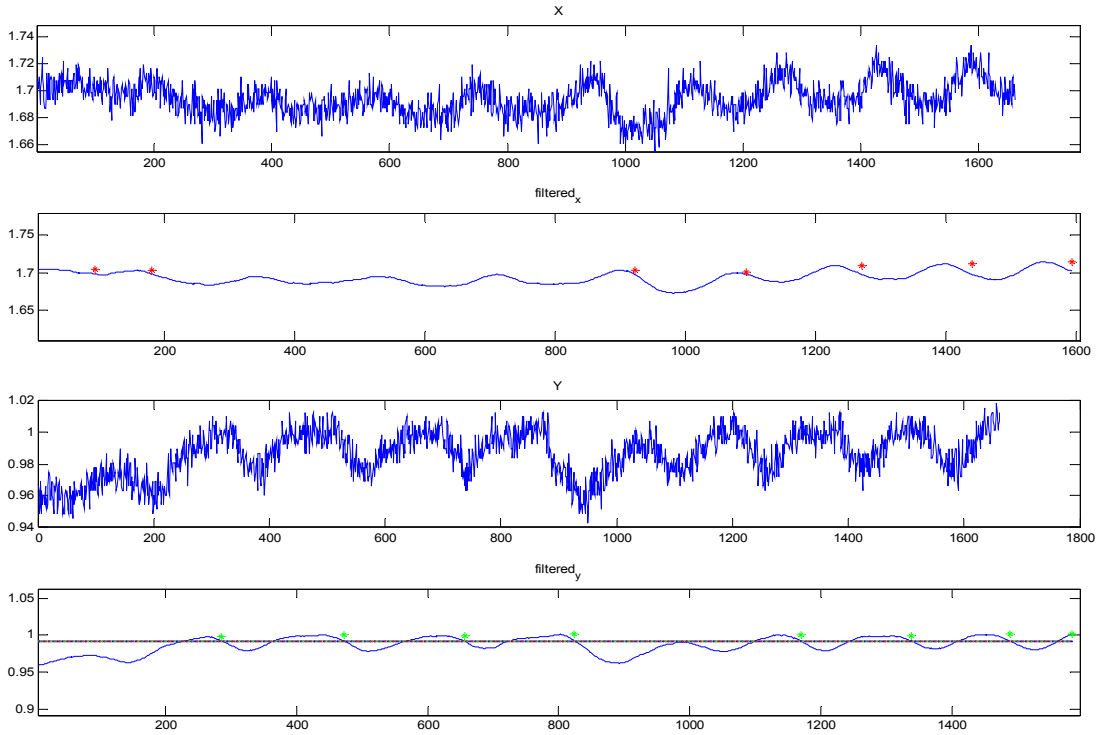
Εικόνα 66-Σήμα resp11_listen.txt.Εφαρμογή της πρώτης μεθόδου για την ανιχνευση αναπνοών.



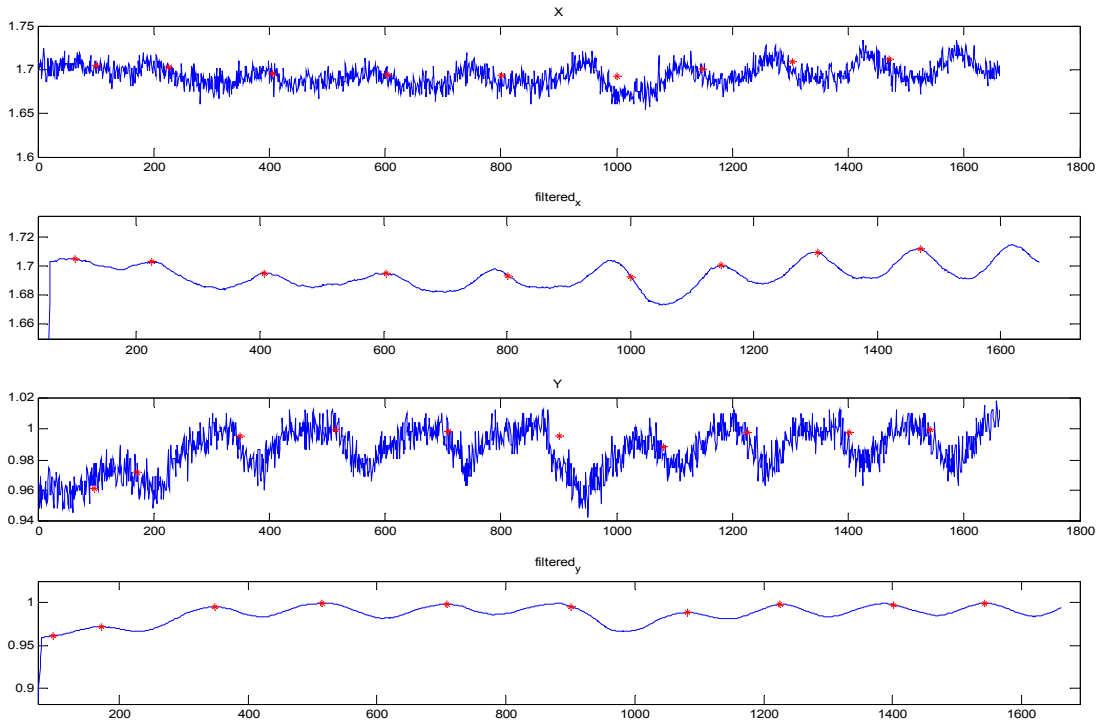
Εικόνα 67-Σήμα resp11_listen.txt.Εφαρμογή της δεύτερης μεθόδου για ανίχνευση των αναπνοών.

Μετρήσεις με το εργαλείο MIG

Μέτρηση resp6_mig.txt σε συχνότητα δειγματοληψίας 20 Hz και εφαρμογή των 2 μεθόδων διαδοχικά. Το σήμα περιέχει 10 αναπνοές .Η πρώτη μέθοδος χάνει σημαντικό μέρος του σήματος στον Χ-άξονα ενώ στον Υ άξονα χάνει μόνο 2 αναπνοές που η κορυφή τους είναι κάτω από το decvalue. Η δεύτερη μέθοδος χάνει την τελευταία κορυφή στον Χ-άξονα μάλλον λόγω παραθύρου και στον Υ άξονα μετρά την πρώτη 2 φορές.

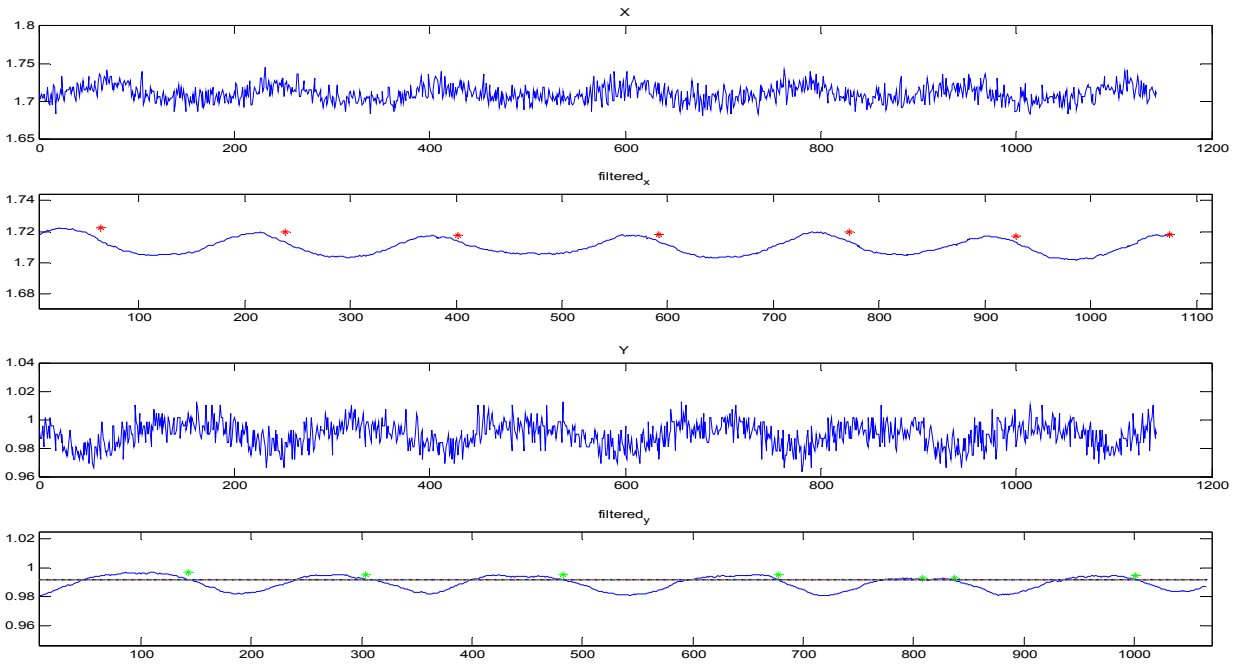


Εικόνα 68-Σήμα resp6_mig.txt. Εφαρμογή της πρώτης μεθόδου ανίχνευσης αναπνοής.

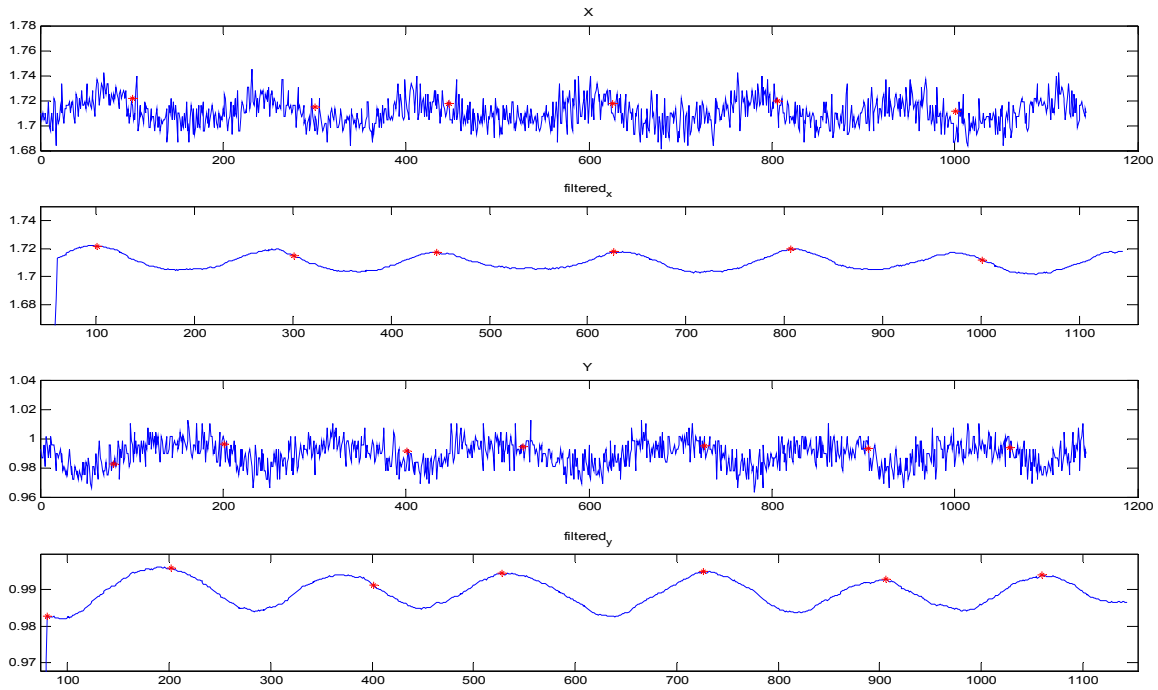


Εικόνα 69-Σήμα resp6_mig.txt. Εφαρμογή της δεύτερησ μεθόδού ανίχνευσης αναπνοής.

Η μέτρηση resp7_mig.txt με συχνότητα δειγματοληψίας 20Hz και εφαρμογή των μεθόδων ανίχνευσης. Το σήμα περιέχει 7 αναπνοές. Παρατηρώ καλή συμπεριφορά και από τις δύο μεθόδους που χρησιμοποιούνται.

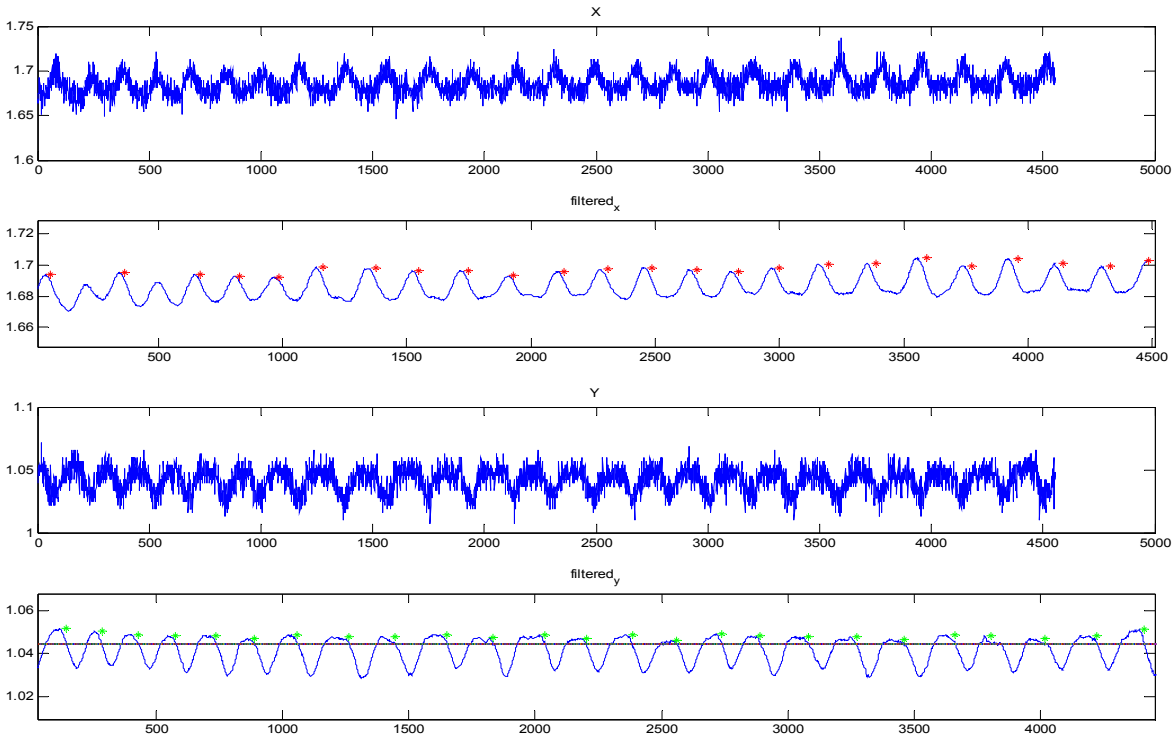


Εικόνα 70-Σήμα resp7_mig.txt.Εφαρμογή της μεθόδου 1

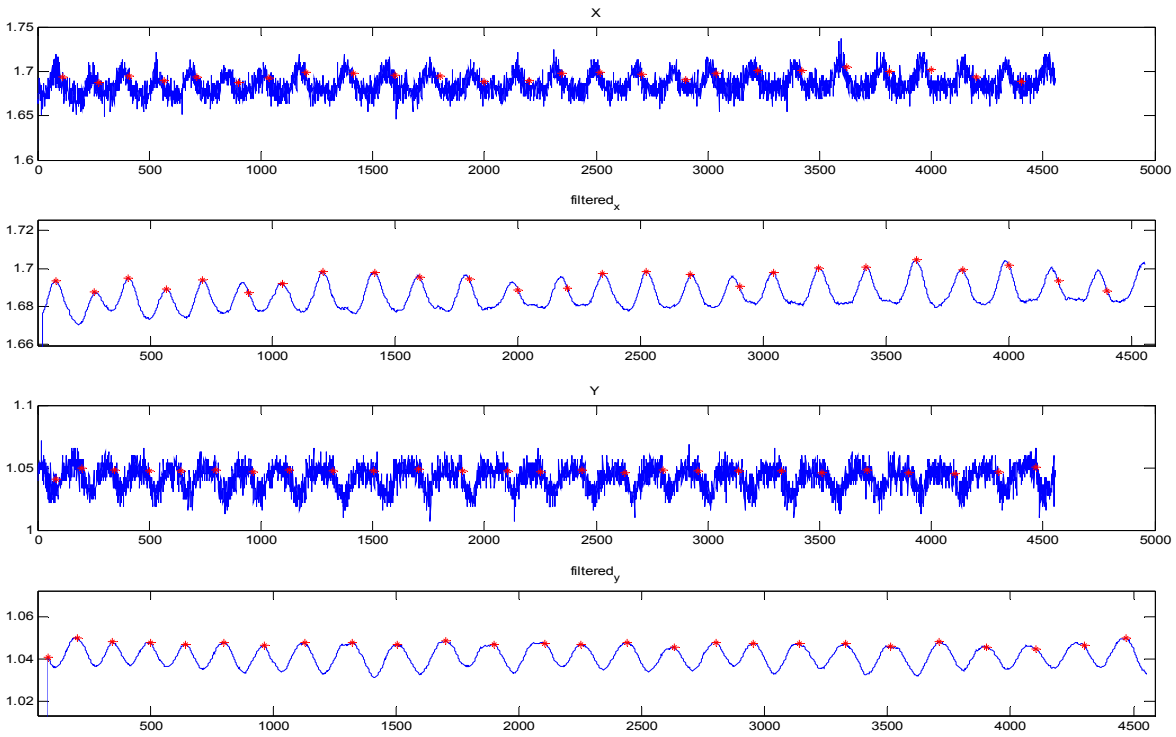


Εικόνα 71-Σήμα resp7_mig.txt.Εφαρμογή της μεθόδου 2

Μέτρηση resp9_mig.txt με συχνότητα δειγματοληψίας 20 Hz και εφαρμογή των 2 μεθόδων. Το σήμα περιέχει 26 αναπνοές. Η πρώτη μέθοδος δίνει καλά στοιχεία αν και χάνει 2 κορυφές στον άξονα X κάτι που μπορεί να βελτιωθεί εύκολα με πιο ελαστικό κριτήριο. Η δεύτερη μέθοδος δίνει ακόμα καλύτερα αποτελέσματα.



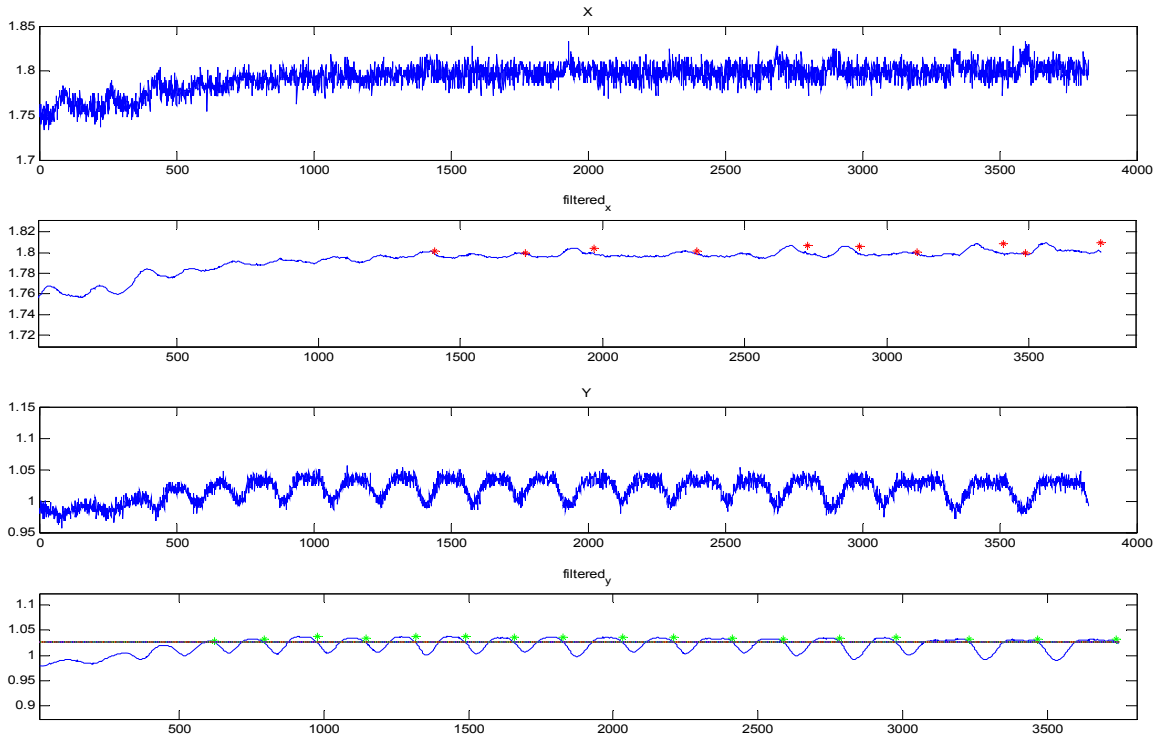
Εικόνα 72-Σήμα resp9_mig.txt.Εφαρμογή της μεθόδου 1



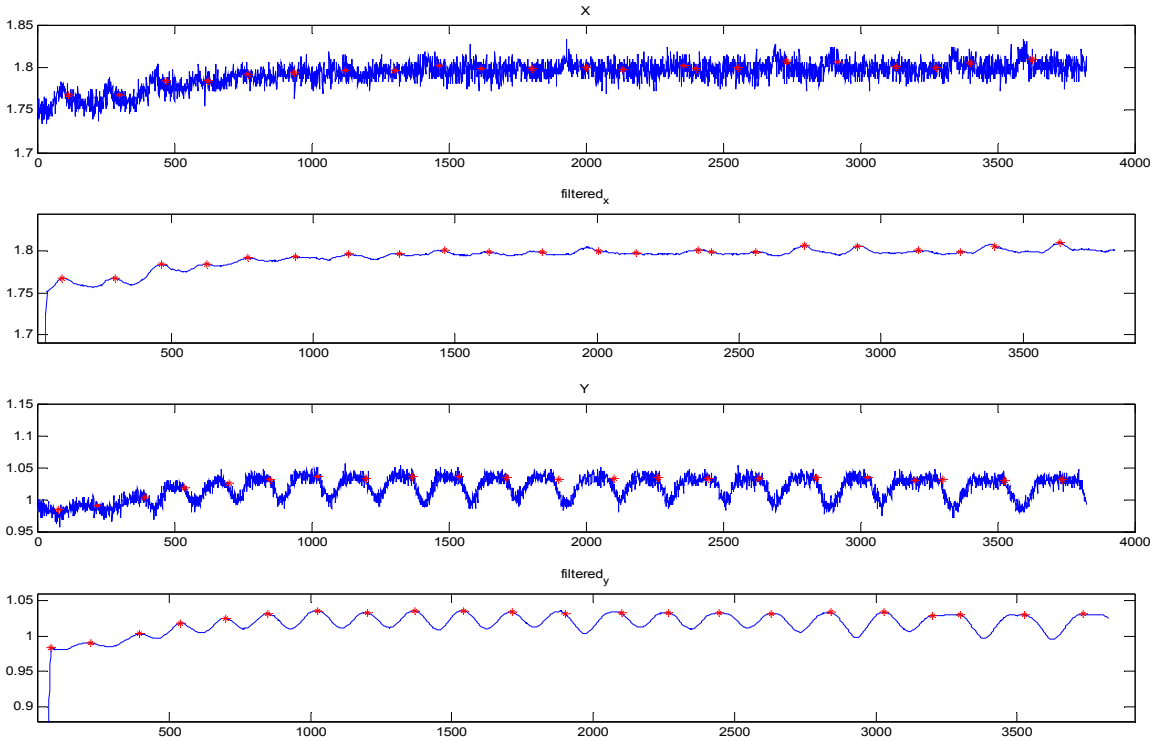
Εικόνα 73-Σήμα resp9_mig.txt.Εφαρμογή της μεθόδου 2

Μέτρηση resp10_mig.txt με συχνότητα δειγματοληψίας 20 Hz και εφαρμογή των 2 μεθόδων. Το σήμα αυτό περιέχει 20 αναπνοές. Διακρίνουμε ότι το σήμα του άξονα X δεν είναι καθαρό και δεν ξεχωρίζουν οι κορυφές –αναπνοές ούτε μετά το φιλτράρισμα. Ακόμα στην αρχή το σήμα έχει μια κλίση η οποία επιδρά στην πρώτη μέθοδο. Η πρώτη μέθοδος καταγράφει μόνο 10 αναπνοές στον X άξονα και χάνει τις πρώτες 3 στον Y. Η δεύτερη μέθοδος αν και φαίνεται να έχει καλυτέρα αποτελέσματα μετρά διπλή μια κορυφή και στους 2 άξονες

.



Εικόνα 74--Σήμα resp10_mig.txt. Εφαρμογή της μεθόδου 1.



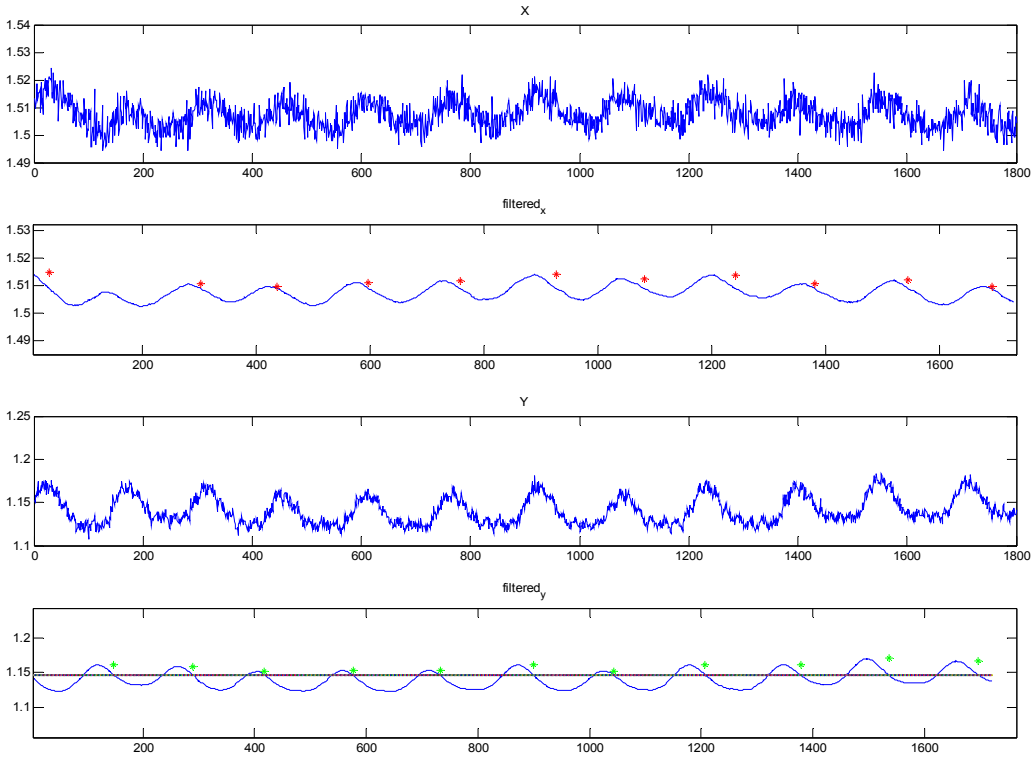
Εικόνα 75--Σήμα resp10_mig.txt.Εφαρμογή της μεθόδου 2.

Δεύτερη σειρά μετρήσεων

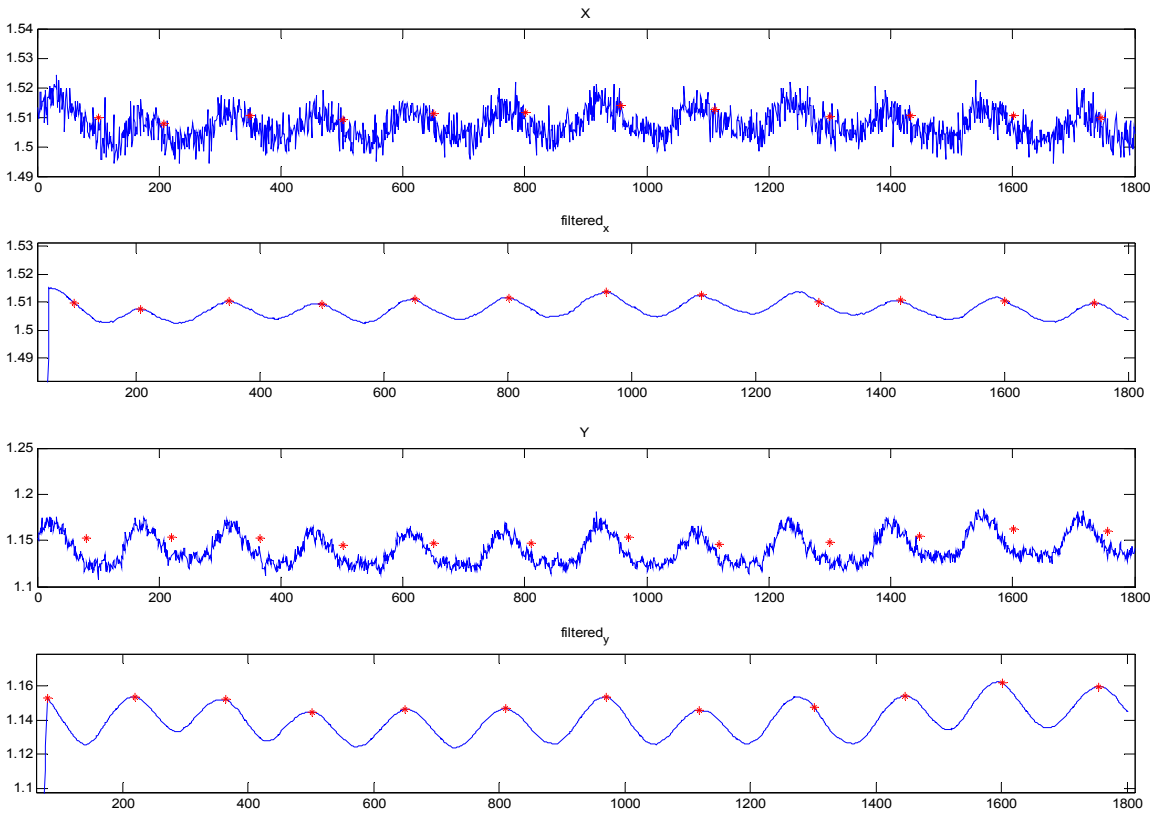
Μετρήσεις στο σημείο 1 με το εργαλείο Listen

Μετρήσεις στο σημείο 1 με το εργαλείο Listen .Αυτή η σειρά μετρήσεων έγινε στο σημείο 1 όπως φαίνεται και στο σχετικό διάγραμμα. Έγιναν μετρήσεις διαδοχικά με το εργαλείο Listen και Mig και με συχνότητα δειγματοληψίας 20 Hz.

Το σήμα που ακολουθεί περιέχει 12-13 αναπνοές. Παρατηρώ ότι η πρώτη μέθοδος χάνει μια αναπνοή στον άξονα X και στον άξονα Y μετρώντας μόνο 11. Η δεύτερη μέθοδος φαίνεται να δουλεύει καλύτερα μετρώντας 12 αναπνοές σε κάθε άξονα.

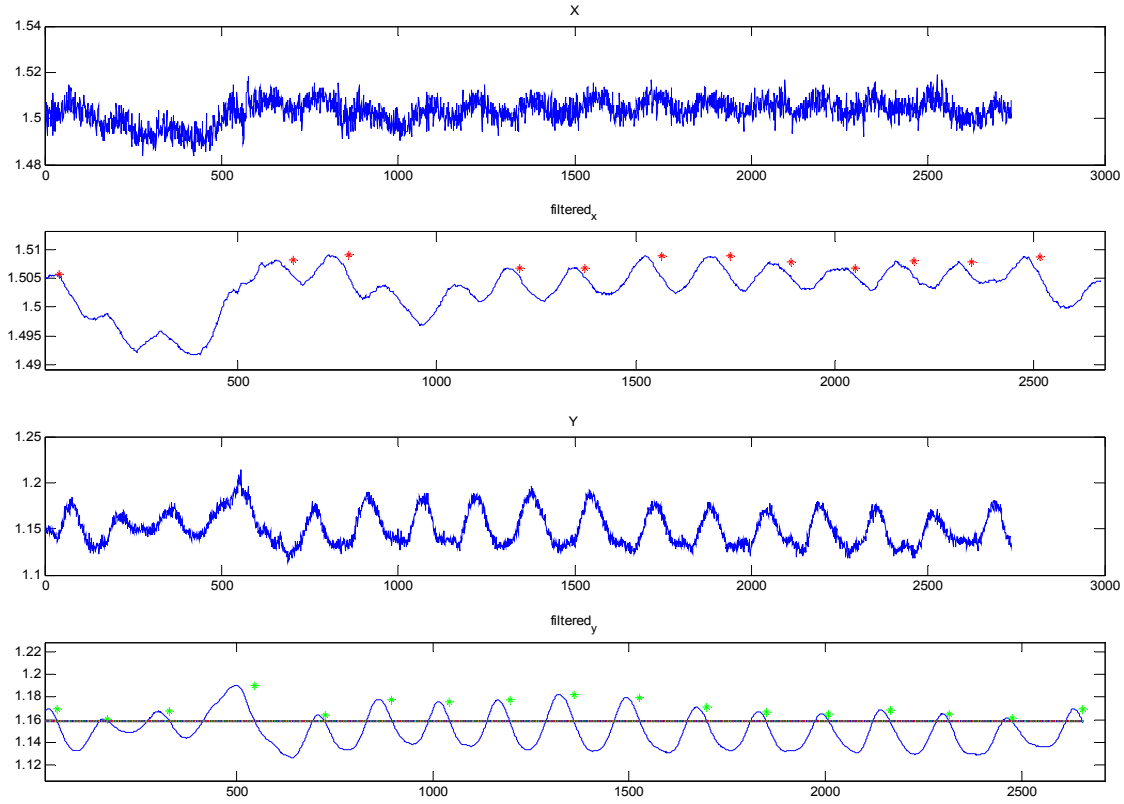


Εικόνα 76-Σήμα resp1-1_listen.txt.Εφαρμογή της μεθόδου 1.

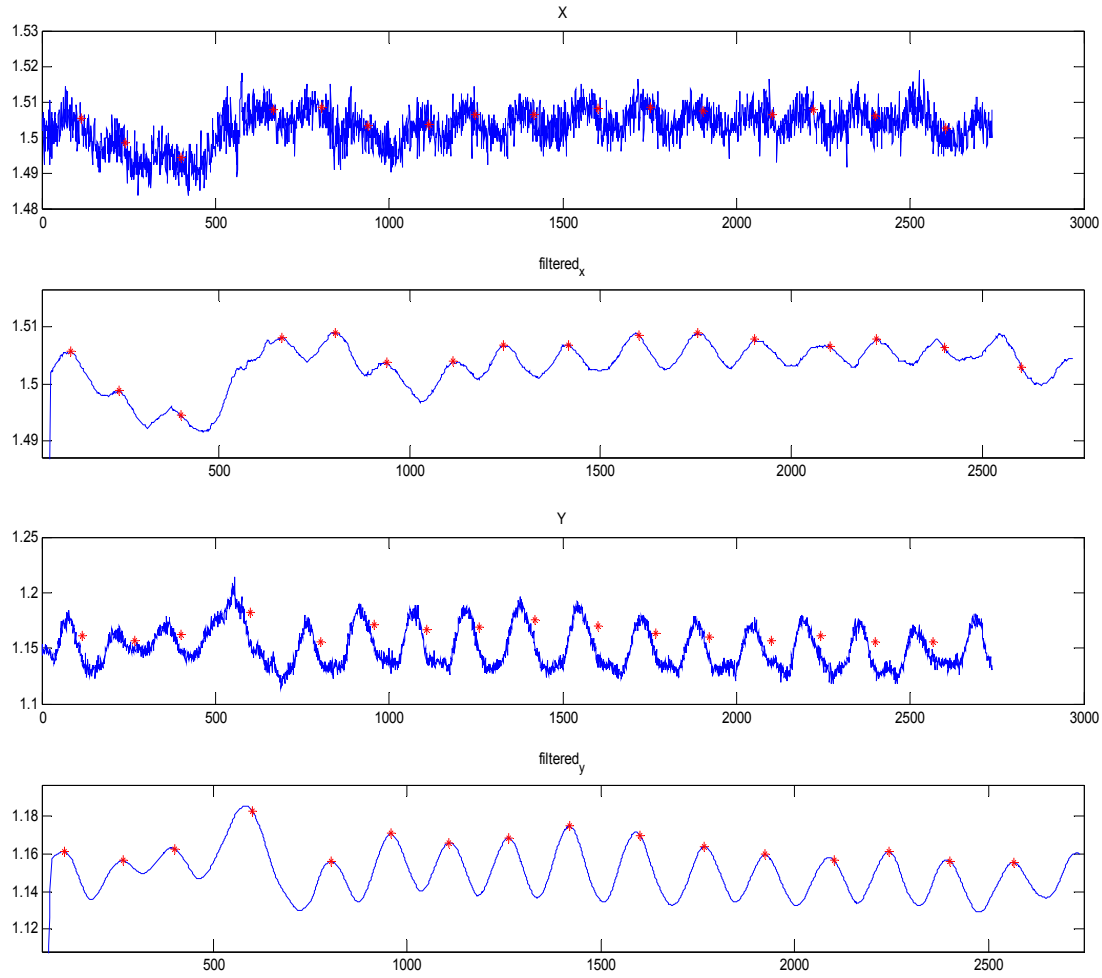


Εικόνα 77-Σήμα resp1_1_listen.txt.Εφαρμογή της μεθόδου 2.

Το σήμα resp1_3_listen.txt έχει δειγματοληπτηθεί με συχνότητα 20 και περιέχει 17 αναπνοές. Το σήμα στον άξονα X δεν είναι τόσο καθαρό όσο το σήμα στον Y άξονα. Για τον λόγω αυτό η πρώτη μέθοδος ανιχνεύει μόνο 12 αναπνοές στον άξονα X ενώ στον Y τις ανιχνεύει όλες. Σαφώς καλύτερα δουλεύει η δεύτερη μέθοδος.

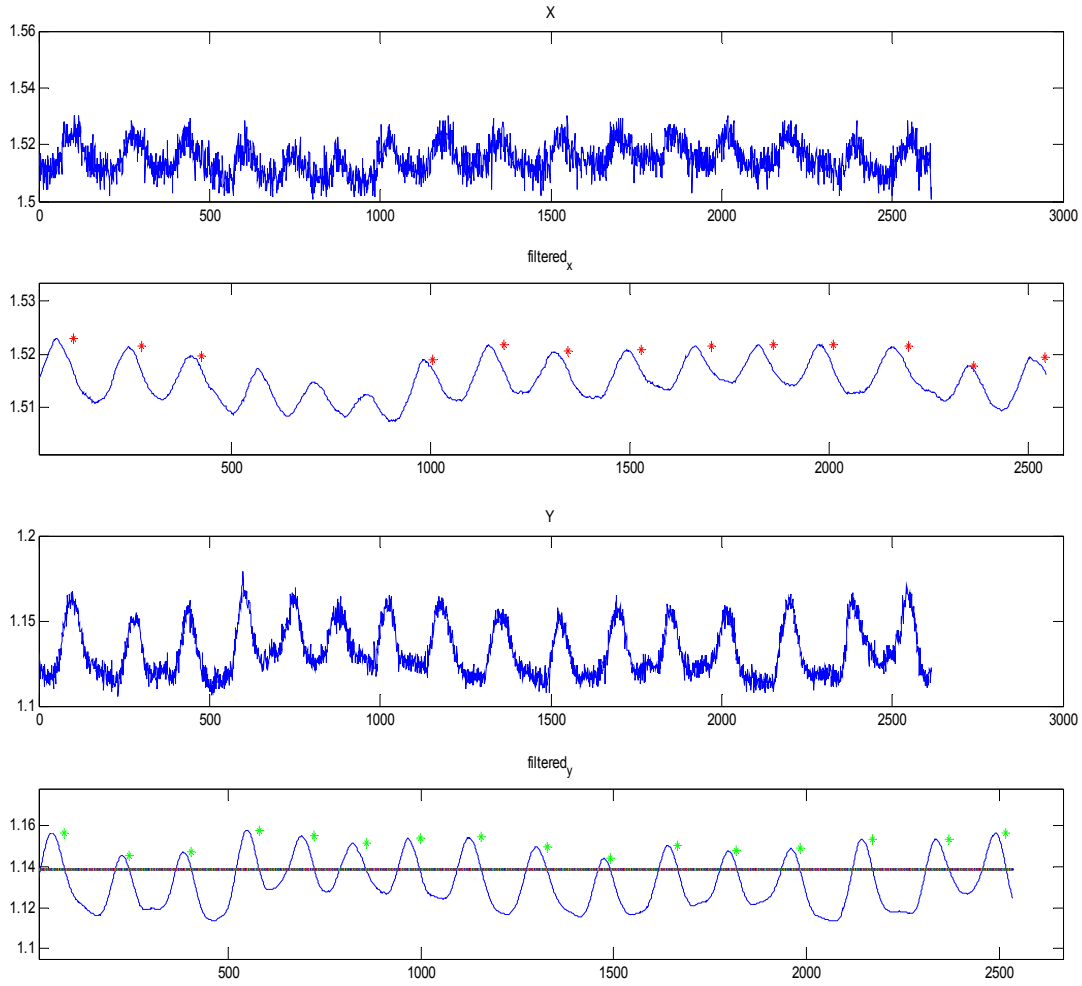


Εικόνα 78-Σήμα resp1_3_listen.Εφαρμογή της μεθόδου 1.

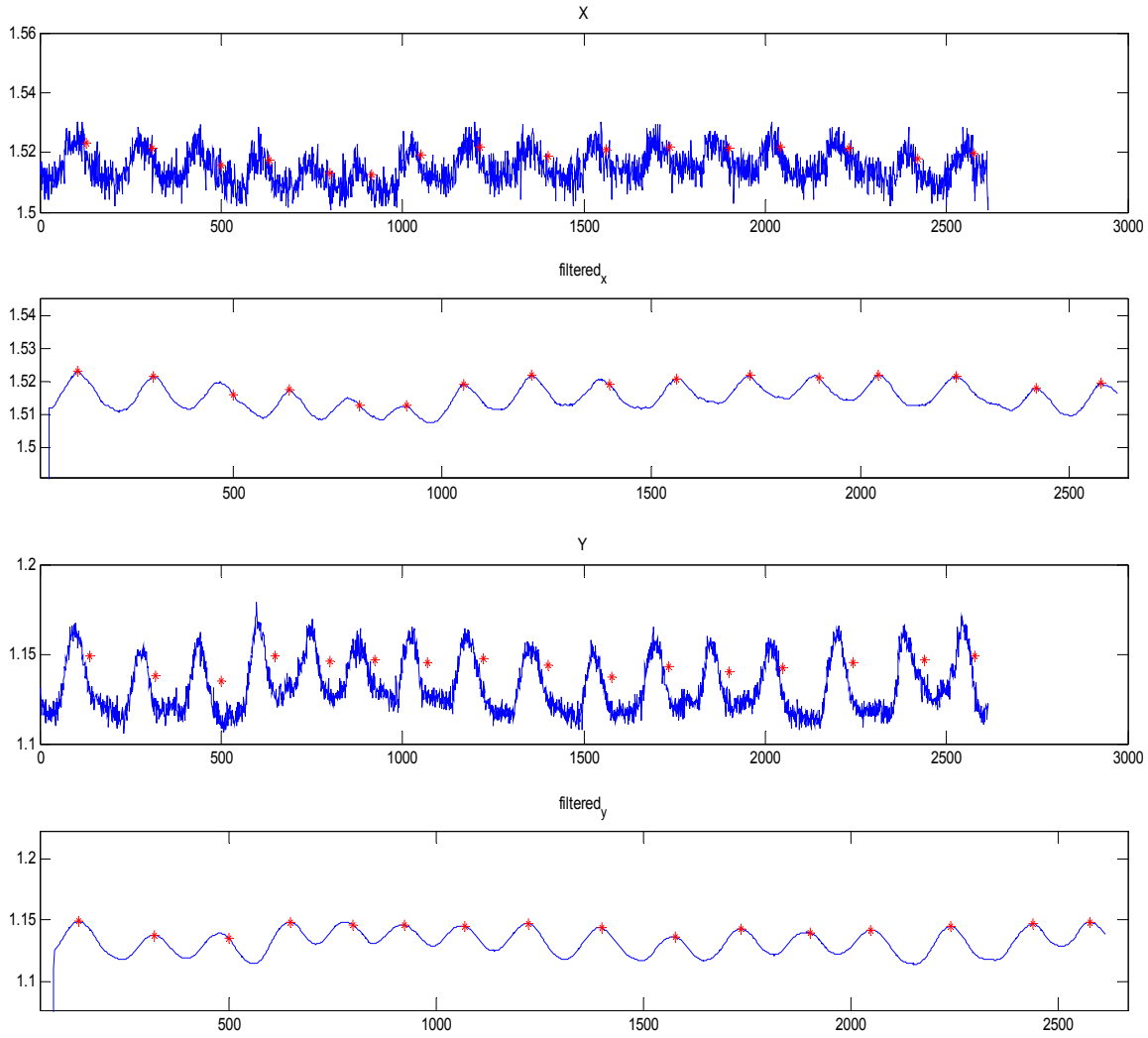


Εικόνα 79-Σήμα resp1_3_listen.txt.Εφαρμογή της μεθόδου 2

Το σήμα resp1_5_listen.txt έχει δειγματοληπτηθεί με συχνότητα 20Hz και περιέχει 17 αναπνοές. Το σήμα στον άξονα X δεν είναι τόσο καθαρό όσο το σήμα στον Y άξονα. Για τον λόγω αυτό η πρώτη μέθοδος ανιχνεύει μόνο 13 αναπνοές στον άξονα X και χάνει όσες είναι κάτω από το decvalue ενώ στον Y τις ανιχνεύει όλες. Η δεύτερη μέθοδος δουλεύει μετράει 16 αναπνοές και στους 2 άξονες.



Εικόνα 80-Σήμα resp1_5_listen.txt.Εφαρμογή της μεθόδου 1.

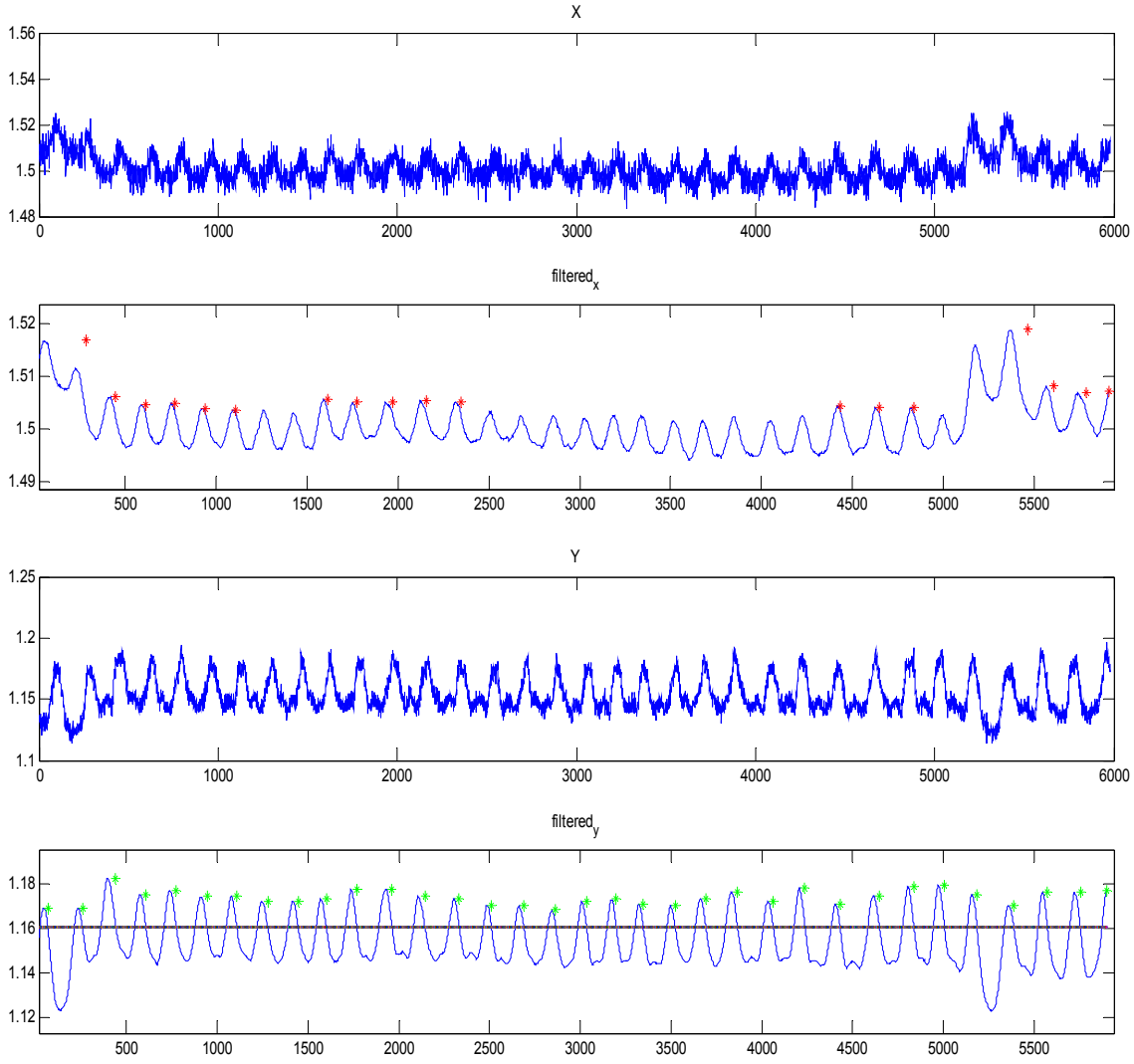


Εικόνα 81-Σήμα resp1-5_listen.txt.Εφαρμογή της μεθόδου 2.

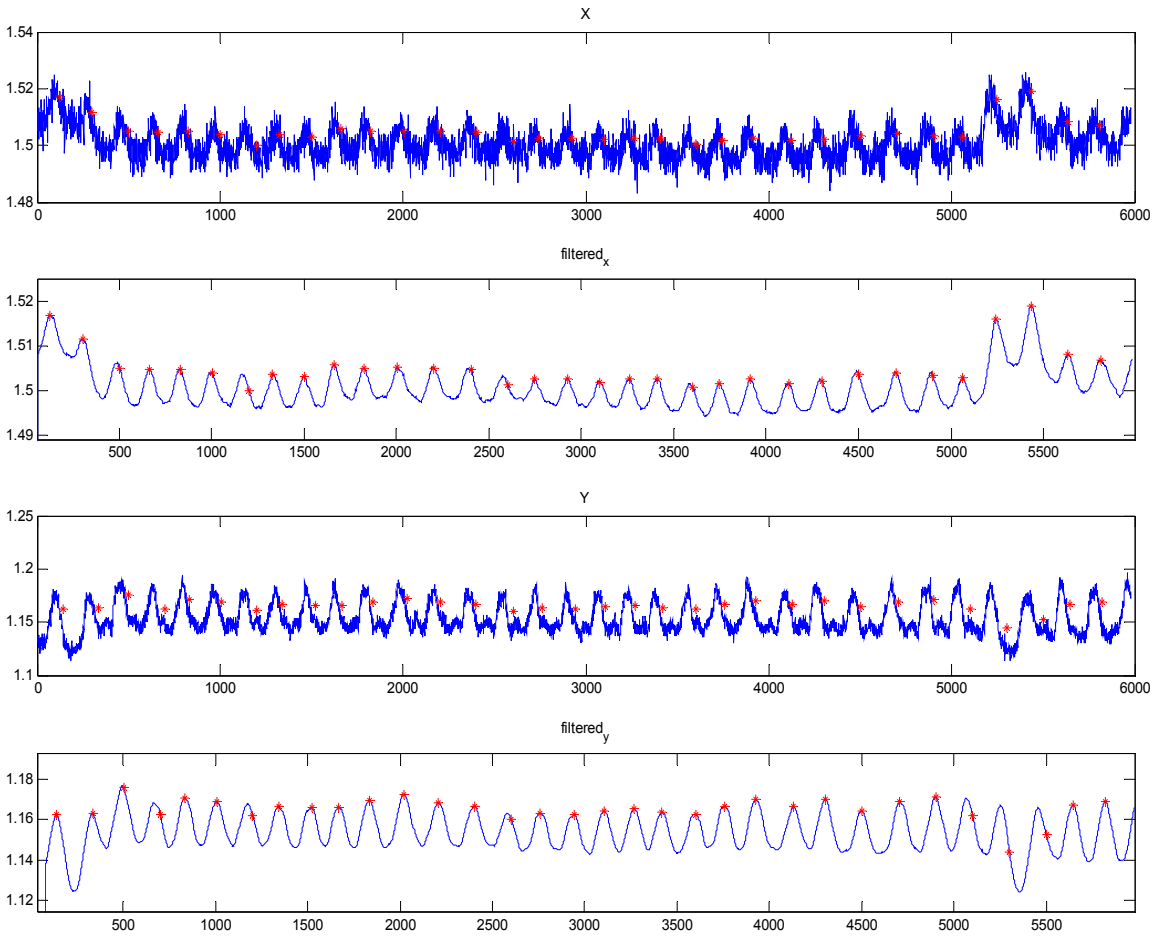
Μετρήσεις στο σημείο 1 με το εργαλείο MIG.

Στη μέτρηση που ακολουθεί το σήμα δειγματοληπτειται με συχνότητα 20 και έχει 36 αναπνοές. Στον άξονα Χ παρατηρούνται τόσο στην αρχή όσο και στο τέλος του σήματος μεταβολές στην τάση. Από την επεξεργασία προκύπτει η πρώτη μεθόδος κρίνεται αναξιόπιστη κυρίως λόγω του μεγαλύτερου θορύβου στον άξονα Χ

από ότι στον Y οπού η μέθοδος 1 παρουσιάζει καλύτερα αποτελέσματα. Όσον αφορά τη μέθοδο 2 αν και παρουσιάζει καλυτέρα αποτελέσματα και δεν επηρεάζεται από τις μεταβολές της τάσης στον άξονα X, μερικές αναπνοές, ειδικά στον Y, άξονα δεν τοποθετούνται στις κορυφές.

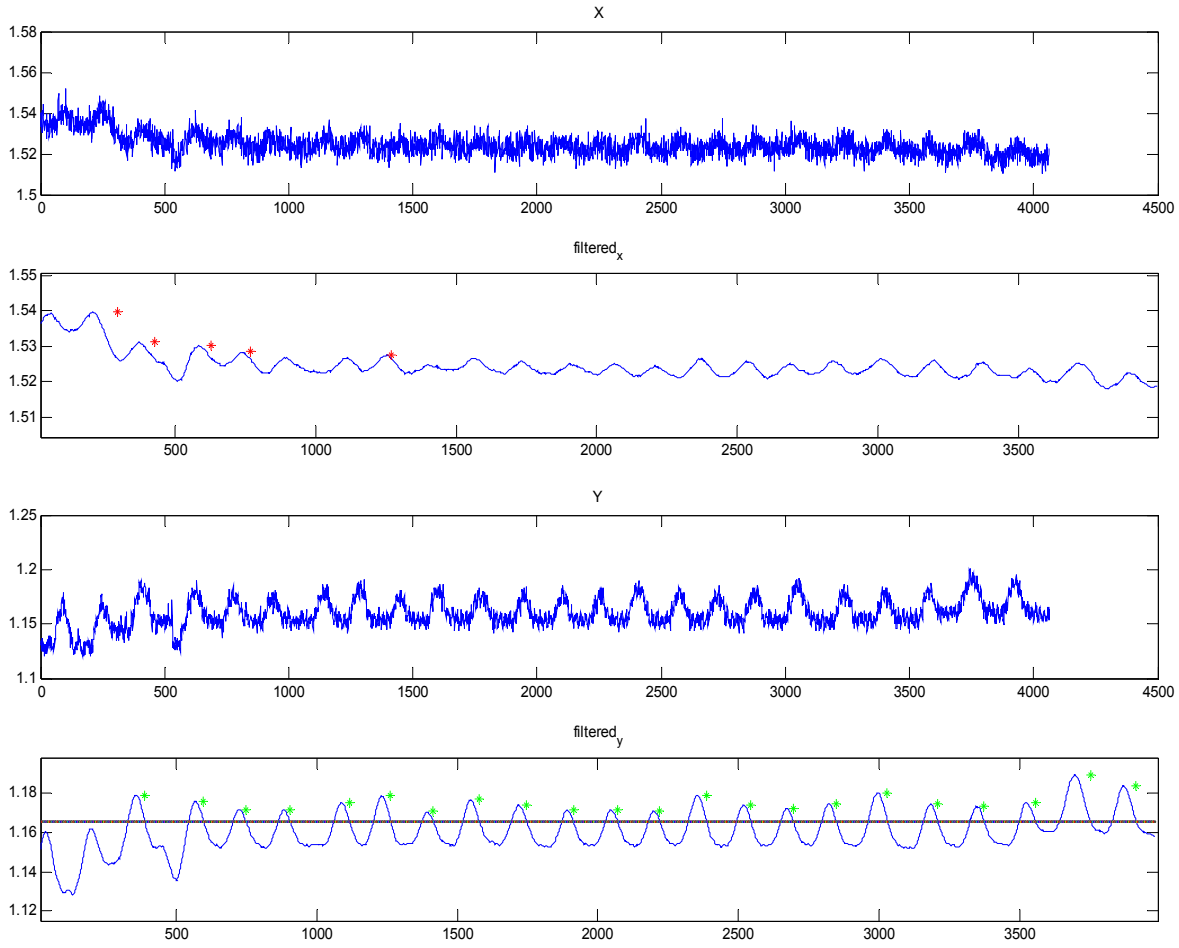


Εικόνα 82-Σήμα resp1_8_mig.txt.Εφαρμογή της μεθόδου 1.

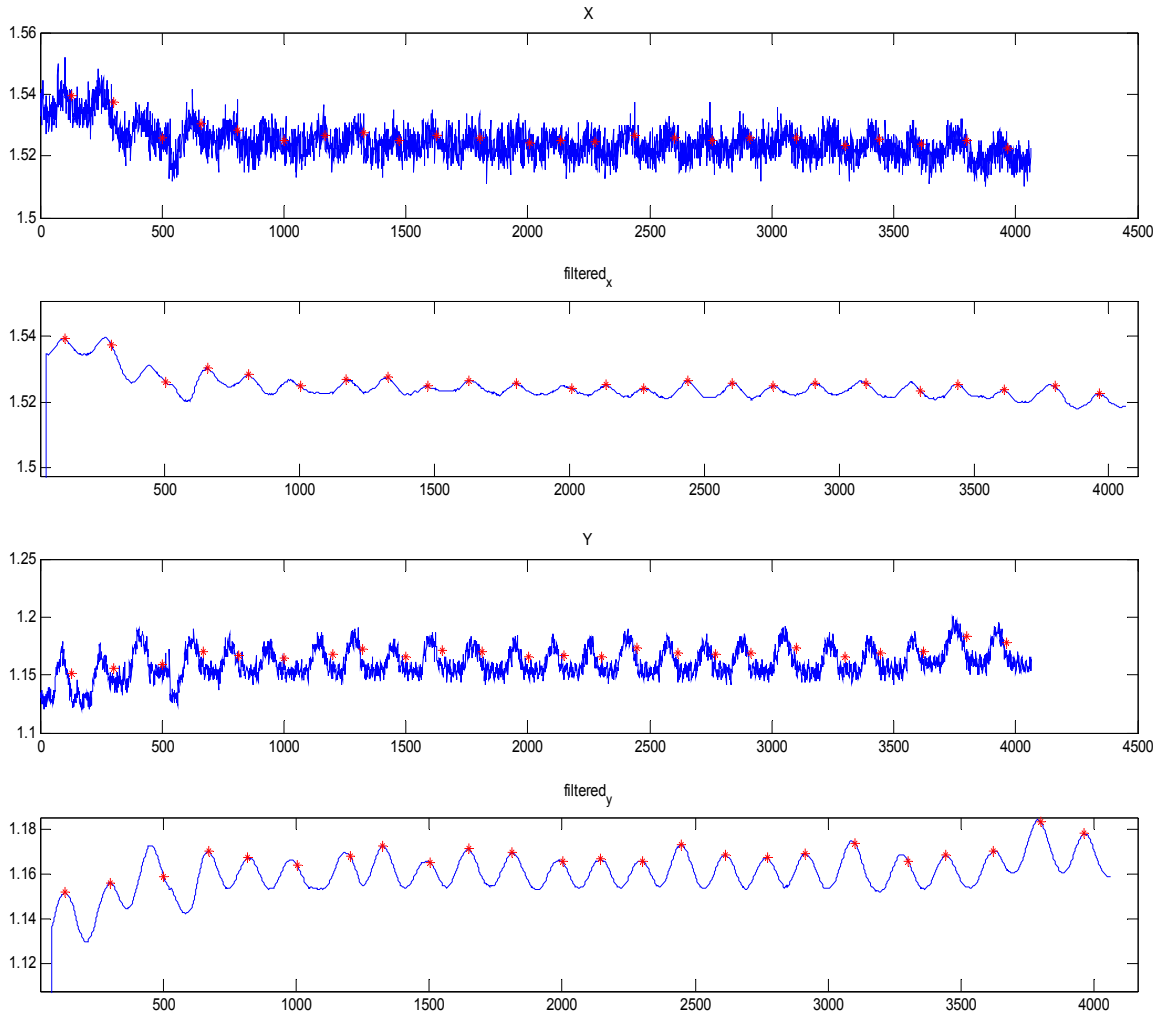


Εικόνα 83-Σήμα resp1_8_mig.txt.Εφαρμογή της μεθόδου 2.

Στη μέτρηση που ακολουθεί το σήμα resp1_9_mig.txt δειγματοληπτείται με συχνότητα 20 και έχει 24 αναπνοές. Από την επεξεργασία προκύπτει η πρώτη μεθόδος κρίνεται αναξιόπιστη κυρίως λόγω του μεγαλύτερου θορύβου στον άξονα X από ότι στον Y οπού η μέθοδος 1 παρουσιάζει καλύτερα αποτελέσματα.Στον άξονα X παρατηρούνται στην αρχή του σήματος μεταβολές στην τάση. Η μέθοδος 2 μετράει τόσο στον X όσο και στον Y άξονα 24 αναπνοές.



Εικόνα 84-Σήμα resp1_9_mig.Εφαρμογή της μεθόδου 1.

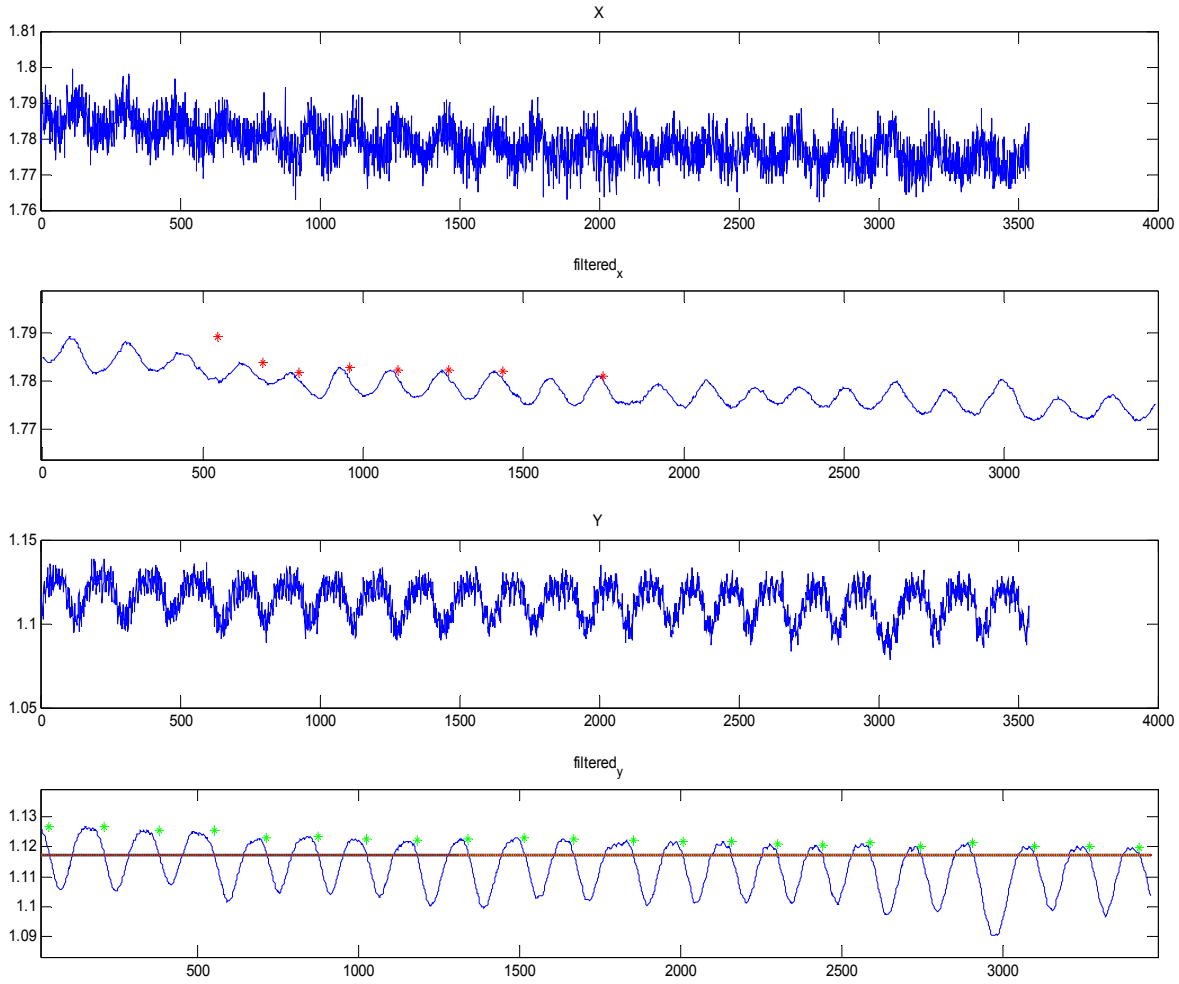


Εικόνα 85-Σήμα resp1_9_mig.txt. Εφαρμογή της μεθόδου 2.

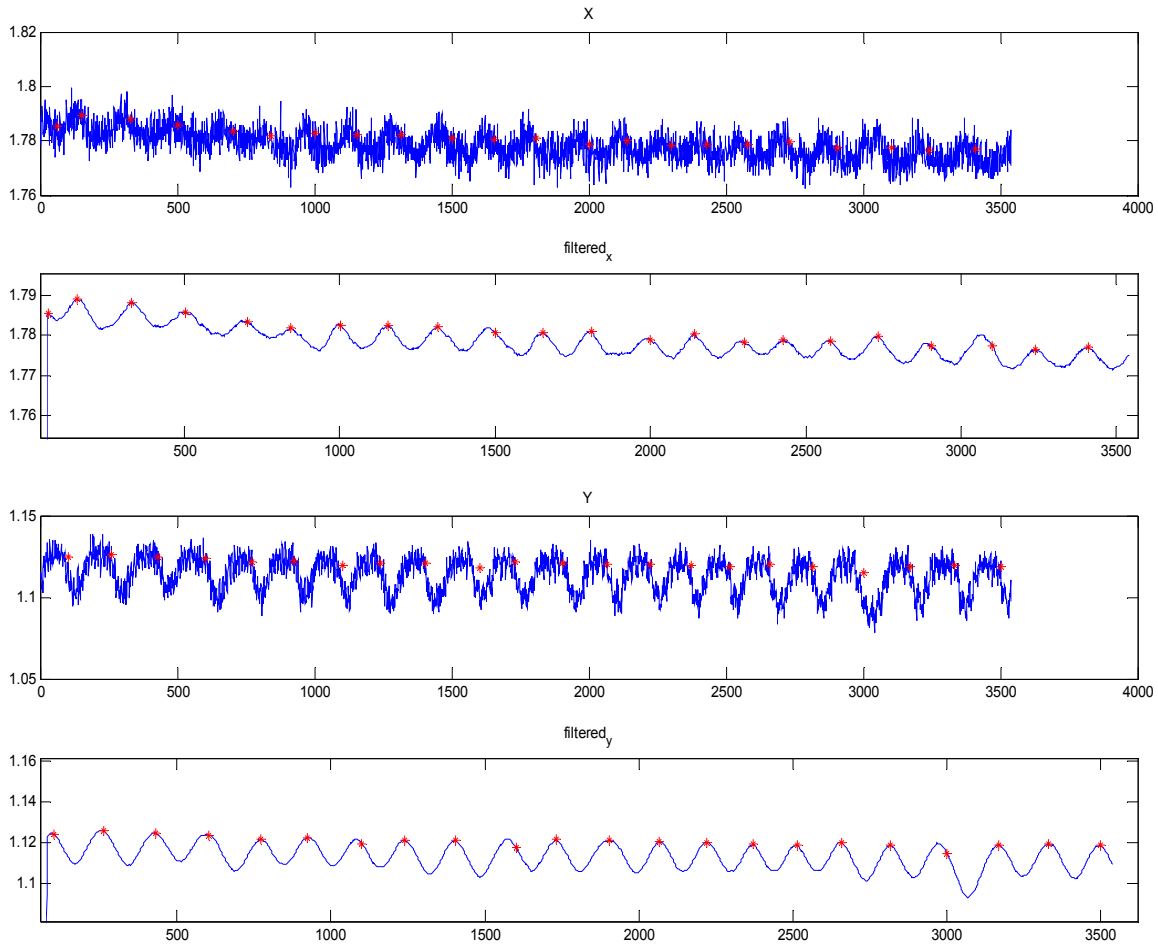
Μετρήσεις στο σημείο 2 με το εργαλείο Listen .

Αυτή η σειρά μετρήσεων έγινε στο σημείο 2 όπως φαίνεται και στο σχετικό διάγραμμα. Έγιναν μετρήσεις διαδοχικά με το εργαλείο Listen και Mig και με συχνότητα δειγματοληψίας 20 Hz.

Το σήμα resp2_3_listen.txt περιέχει 23 αναπνοές. Φαίνεται και πάλι ότι το σήμα του άξονα X περιέχει πολύ περισσότερο θόρυβο σε σχέση με τον σήμα του άξονα Y. Η μέθοδος 1 δεν ανταποκρίνεται καλά στο σήμα του X άξονα ενώ δίνει καλύτερα αποτελέσματα στο Y άξονα. Αντίθετα η μέθοδος 2 ανταποκρίνεται το ίδιο καλά και στους 2 άξονες.

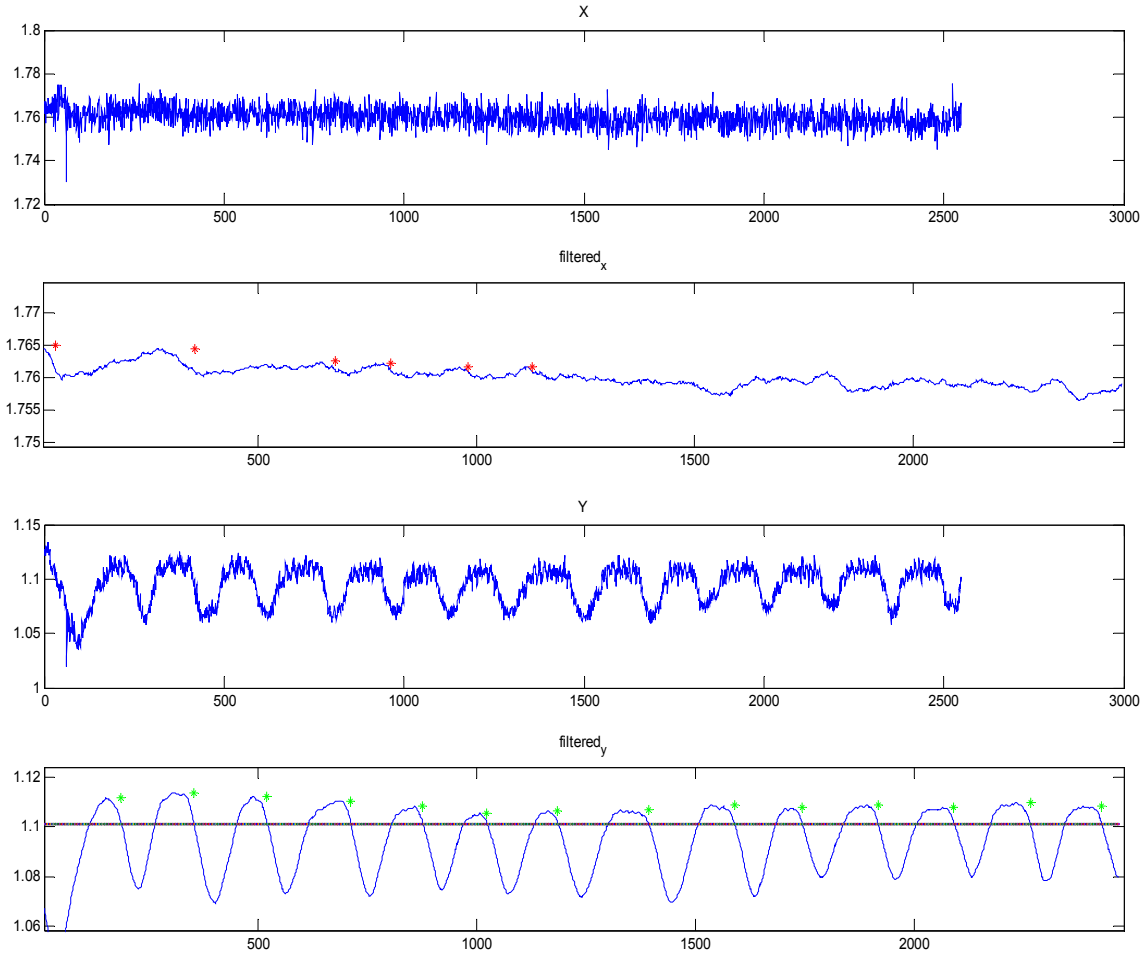


Εικόνα 86-Σήμα resp2_3_listen.txt. Εφαρμογή της πρώτης μεθόδου.

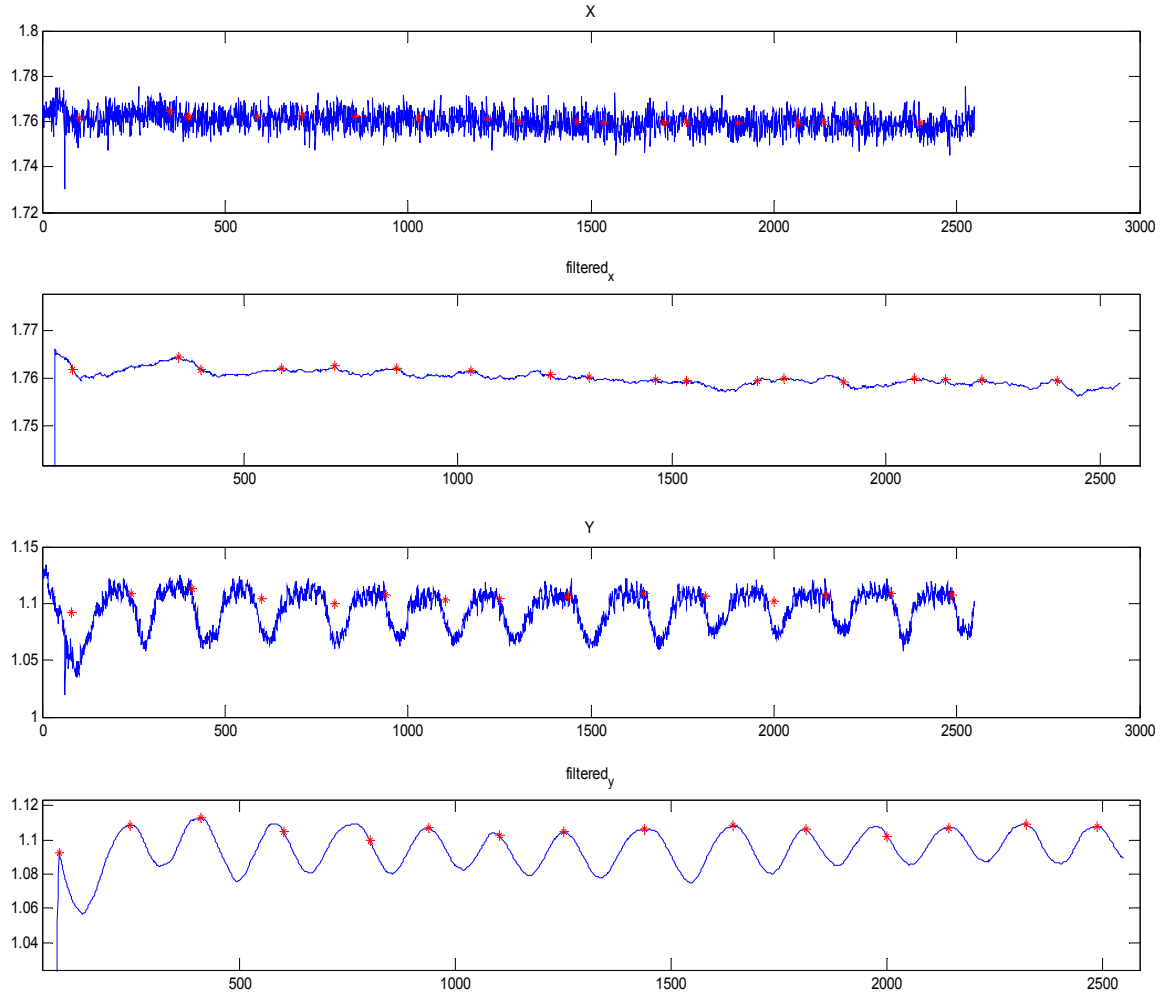


Εικόνα 87-Σήμα resp2_3_listen.txt. Εφαρμογή της μεθόδου 2.

Το σήμα resp2_4_listen.txt περιέχει 17 αναπνοές. το σήμα του άξονα X περιέχει πολύ περισσότερο θόρυβο σε σχέση με τον σήμα του άξονα Y. Συγκεκριμένα ο άξονας X δεν περιέχει καμιά χρήσιμη πληροφορία και για τον λόγο αυτό και οι δύο μέθοδοι δεν δίνουν καλά αποτέλεσματα. Αντίθετα το σήμα στον Y άξονα περιέχει ξεκάθαρη πληροφορία και αυτό προκύπτει και στην επεξεργασία από την εφαρμογή των μεθόδων ανίχνευσης.

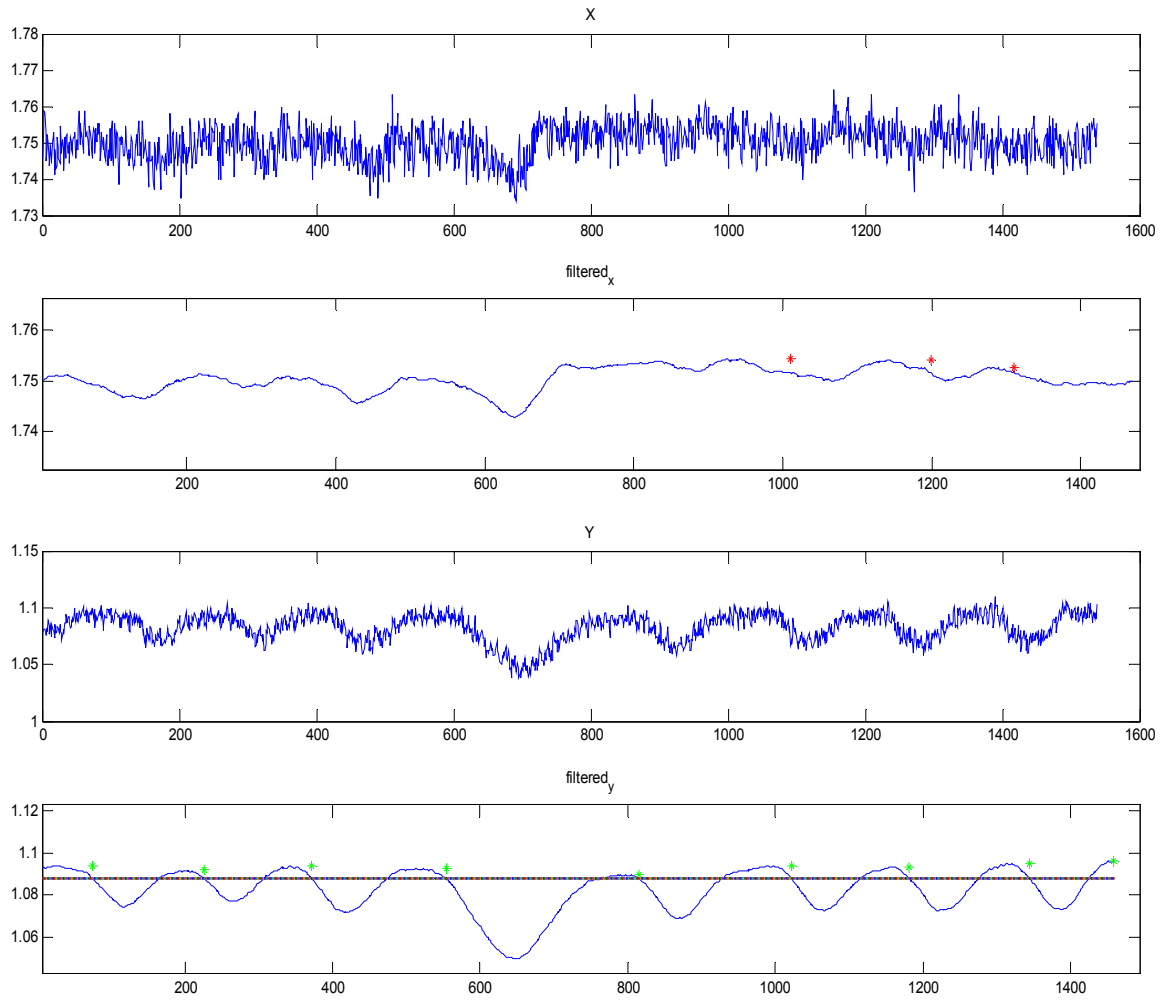


Εικόνα 88-Σήμα resp2_4_listen.txt. εφαρμογή της μεθόδου 1.

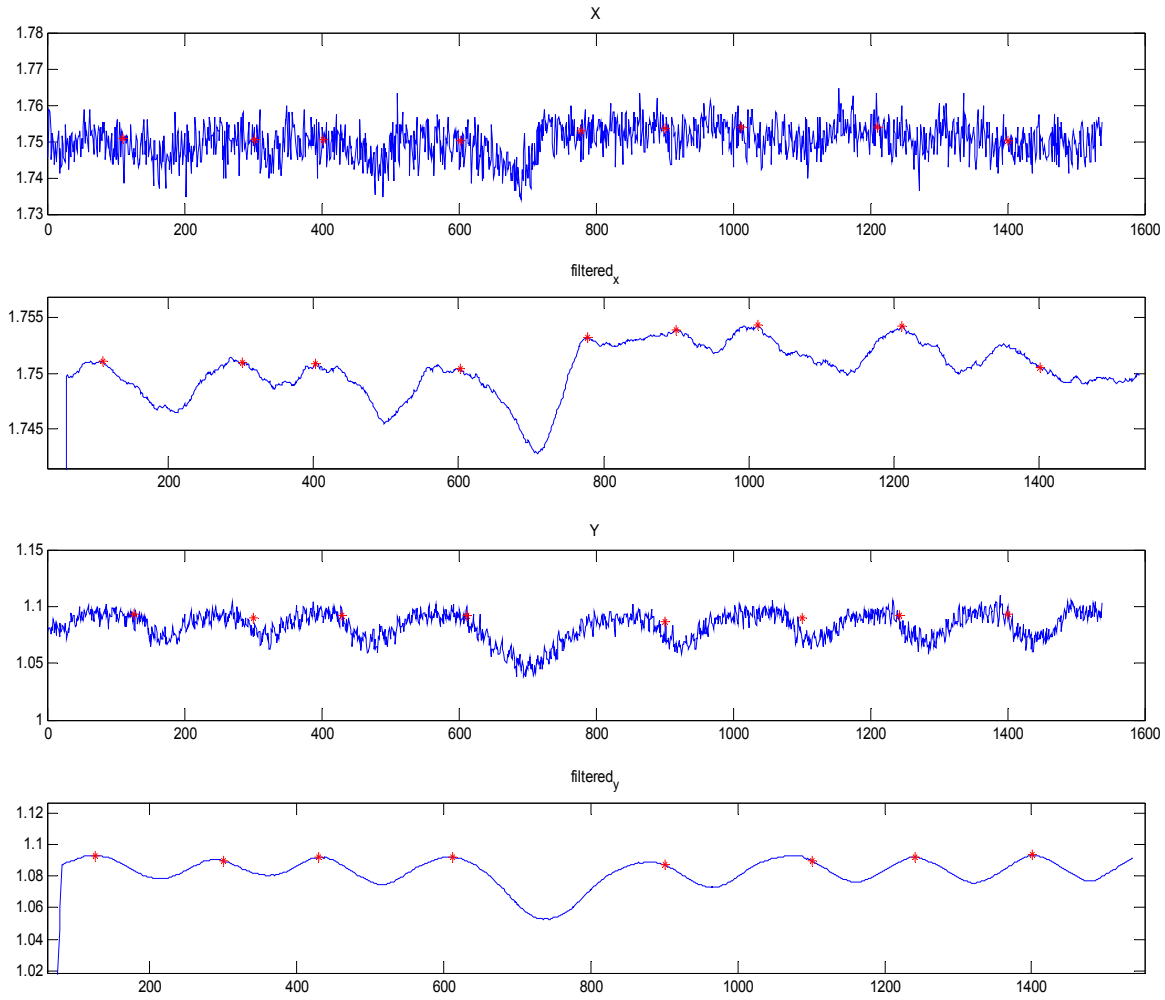


Εικόνα 89-Σήμα resp2_4_listen.txt. Εφαρμογή της μεθόδου 2.

Το σήμα που ακολουθεί resp2_5_listen.txt περιέχει 10 αναπνοές. Το σήμα του X άξονα όμως δεν είναι εύκολα επεξεργάσιμο.Αυτό προκύπτει κυρίως γιατί στο συγκεκριμένο σημείο μετρήσεων(2) οι μετατοπίσεις συνεπώς και οι επιταχύνσεις ήταν μικρές. Το σήμα του Y άξονα από την άλλη είναι καλό και ανταποκρίνεται καλά στη εφαρμογή των 2 μεθόδων.



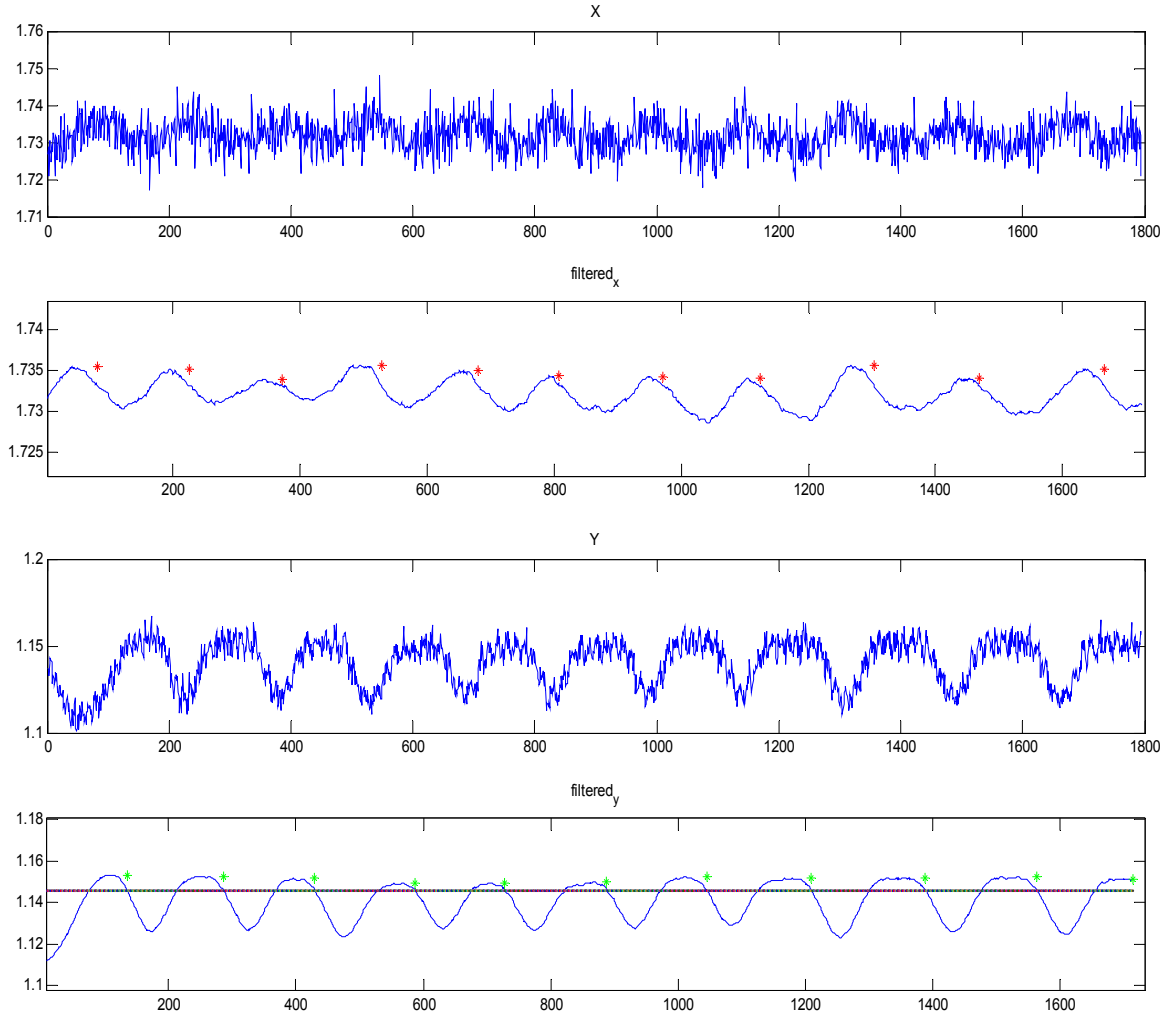
Εικόνα 90-Σήμα resp2_5_listen.Εφαρμογή της μεθόδου 1.



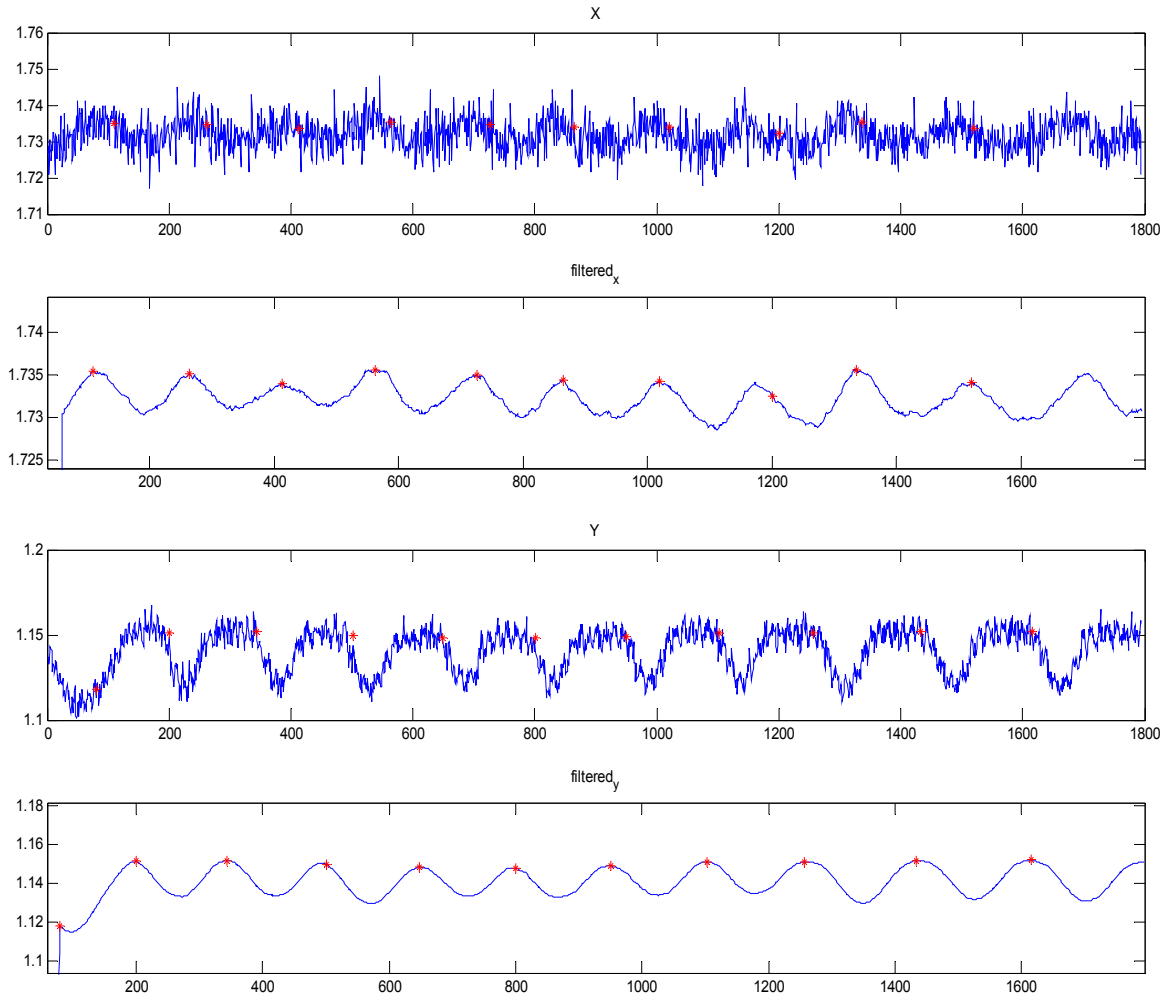
Εικόνα 91-Σήμα resp2_5_listen.txt. Εφαρμογή της μεθόδου 2.

Μετρήσεις στο σημείο 2 με το εργαλείο MIG .

Το σήμα resp2_6-mig.txt περιέχει 11 αναπνοές . Έχει ληφθεί με δειγματοληψία 20 Hz. Το σήμα είναι πολύ καλό και στους 2 άξονες έτσι και οι 2 μέθοδοι ανίχνευσης δίνουν πολύ καλά αποτελέσματα.

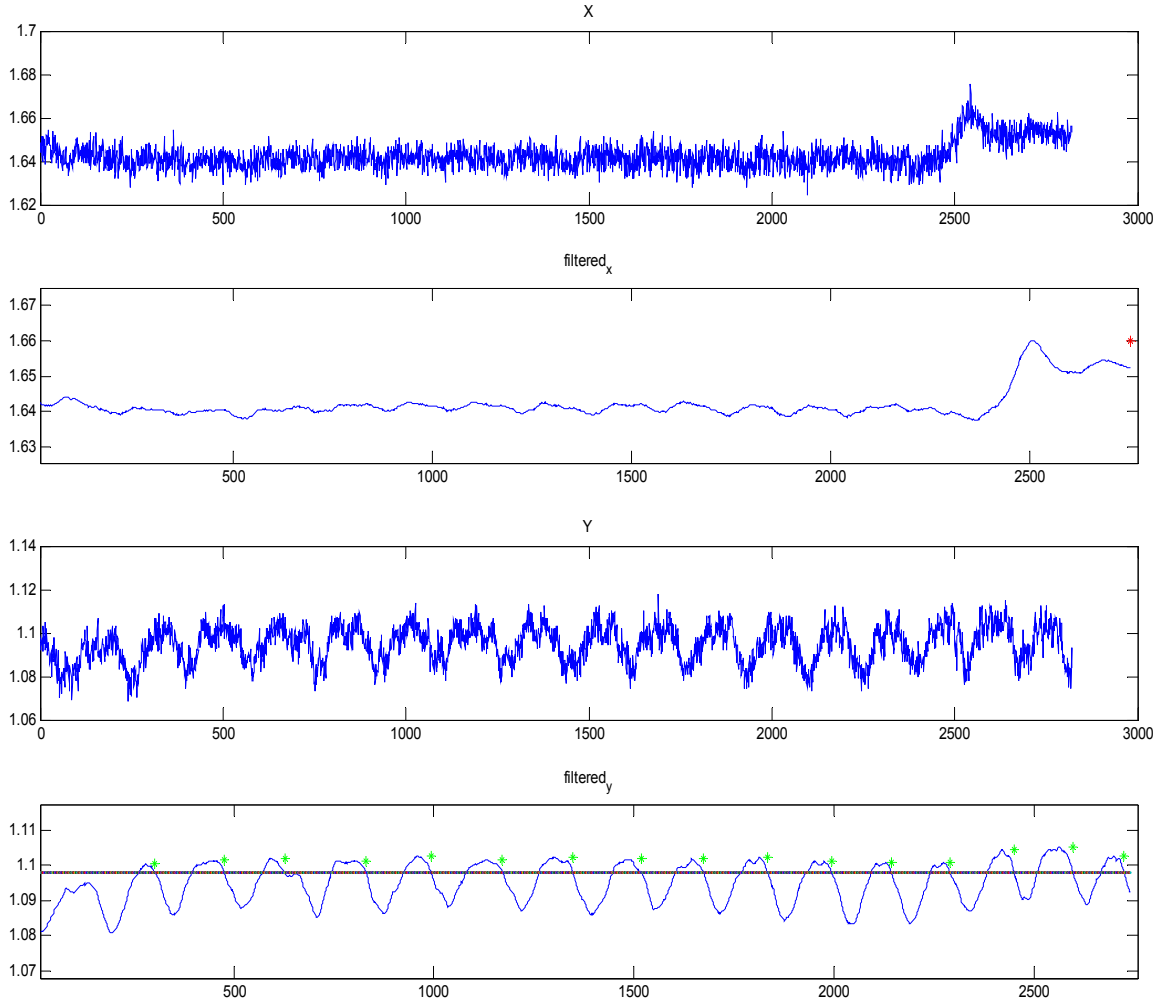


Εικόνα 92-Σήμα resp2_6_mig.txt.Εφαρμογή μεθόδου 1.

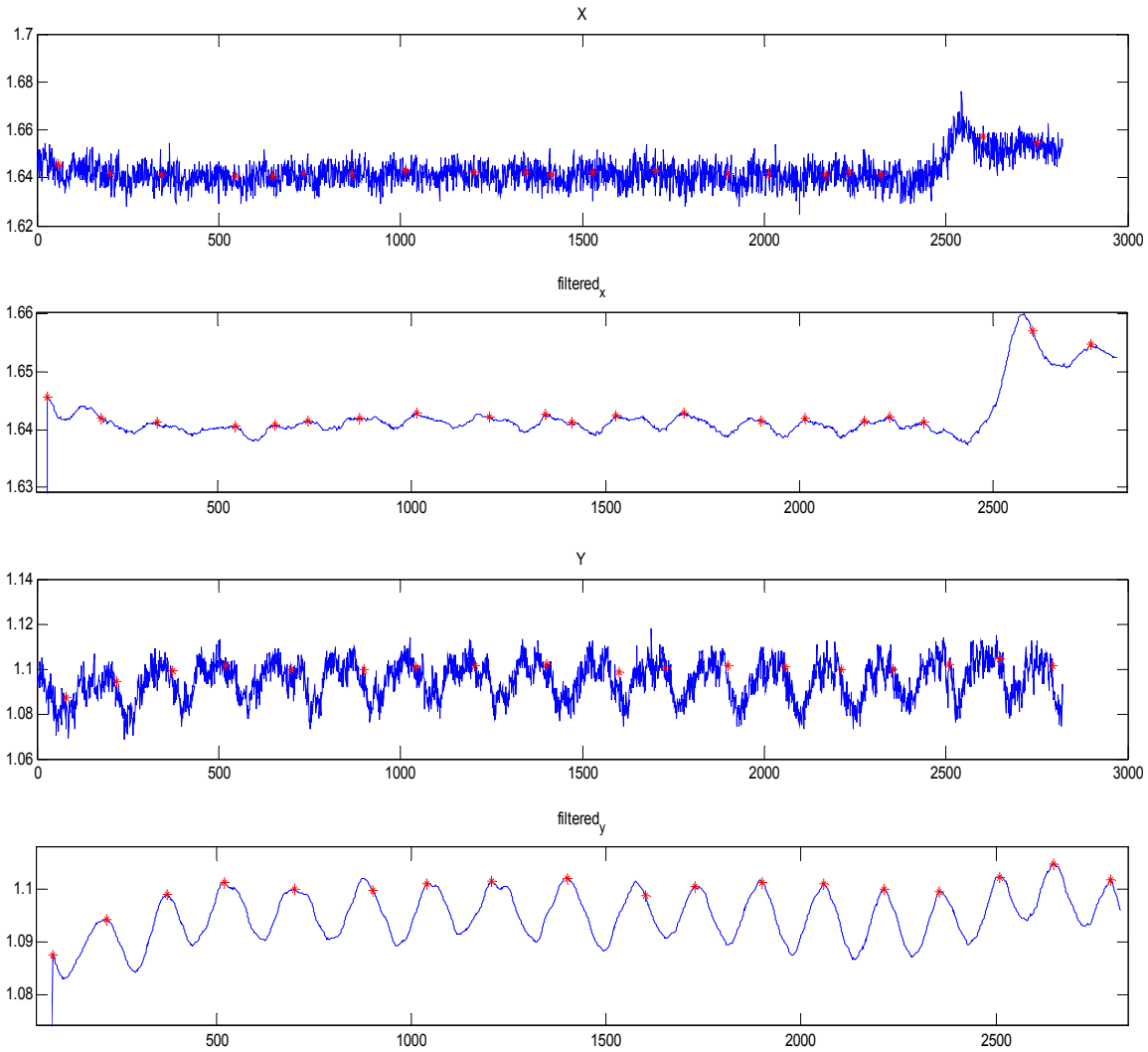


Εικόνα 93-Σήμα resp2_6_mig.txt.Εφαρμογή μεθόδου 2.

Το σήμα resp2_8_mig.txt περιέχει 18 αναπνοές. Το σήμα όμως στον άξονα X δεν μπορεί να δώσει πληροφορία σχετικά με τις αναπνοές με καμία από τις 2 μεθόδους ανίχνευσης γιατί περιέχει πολύ θόρυβο και λίγη πληροφορία. Το σήμα του άξονα Y περιέχει την ζητούμενη πληροφορία και αποδίδουν καλά και οι 2 μέθοδοι ανίχνευσης εκτός από την 1 αναπνοή με την 1 μέθοδο που βρίσκεται κάτω από το σχετικό κατώφλι. Μπορούμε φυσικά να αλλάξουμε το κατώφλι ώστε τα κριτήρια να γίνουν πιο ελαστικά και να αναγνωρίσει το κριτήριο και αυτή την αναπνοή.

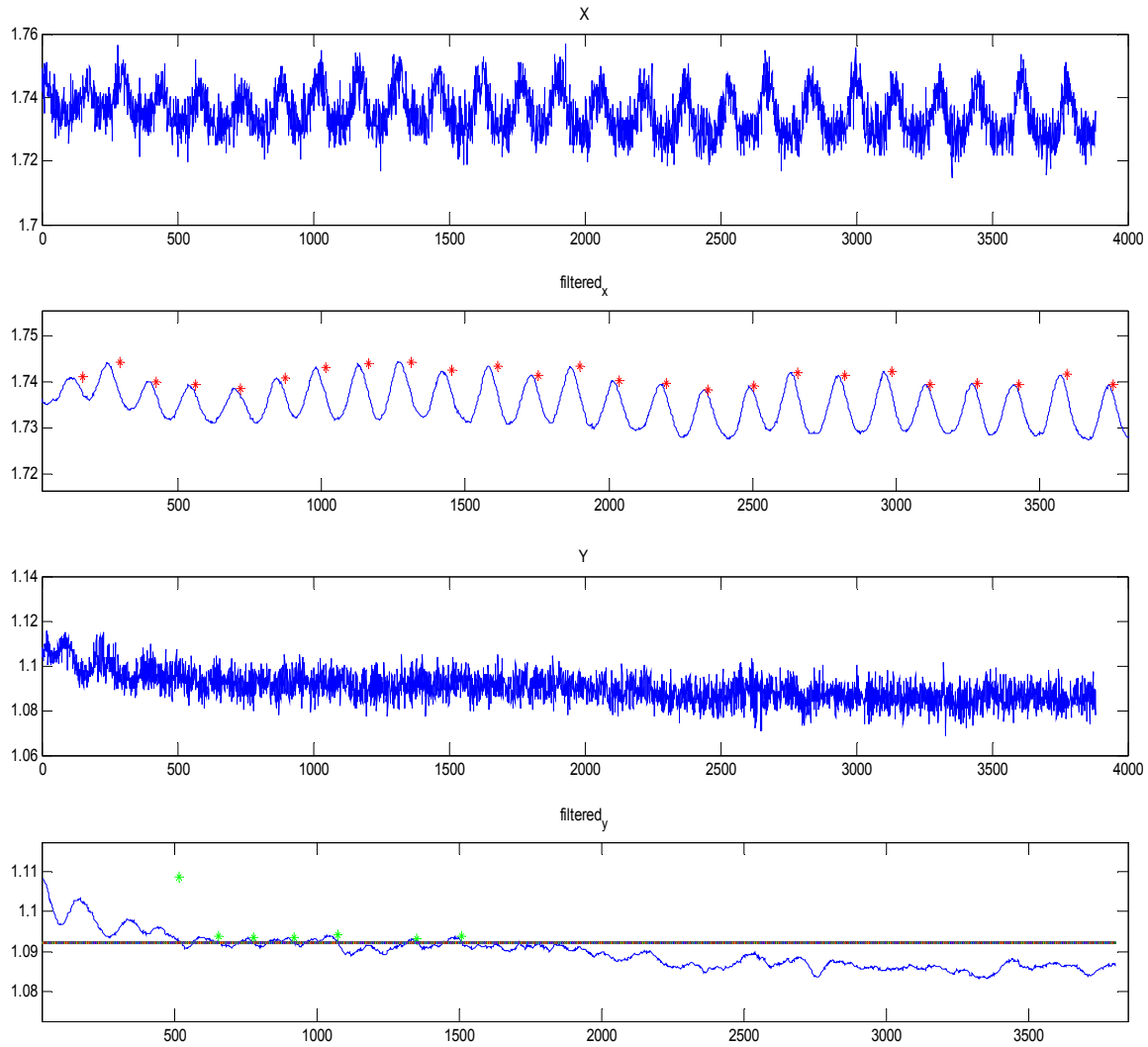


Εικόνα 94-Σήμα resp2_8_mig.txt .Εφαρμογή μεθόδου 1.

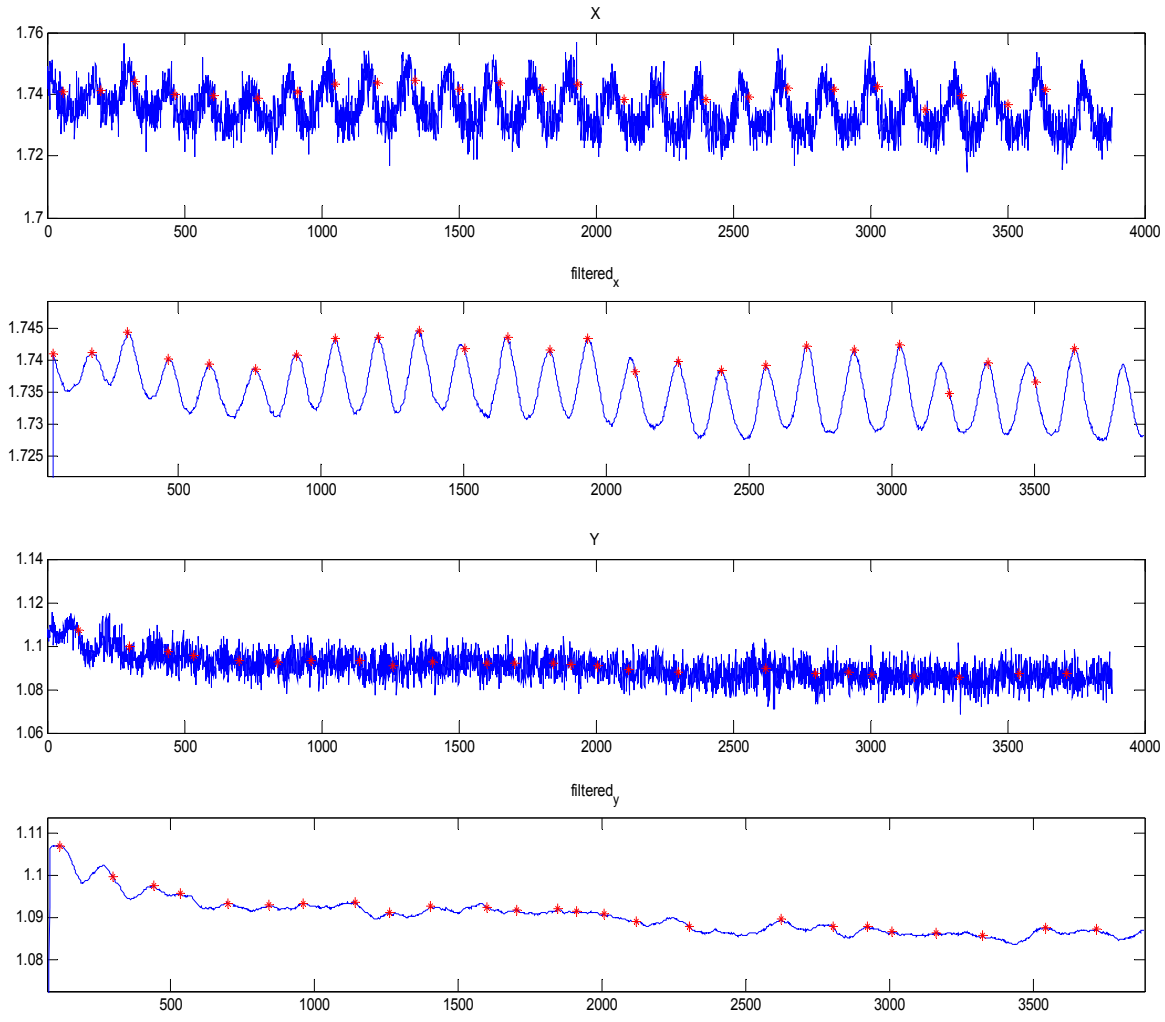


Εικόνα 95-Σήμα resp2_8_mig.txt. Εφαρμογή της μεθόδου 2.

Σε αντίθεση με το προηγούμενο σήμα το resp2_10_mig.txt που ακολουθεί δεν περιέχει ξεκάθαρη πληροφορία στον άξονα Υ και το σήμα αυτό μοιάζει με θόρυβο με αποτέλεσμα να μην λαμβάνονται αξιόπιστα αποτελέσματα με την εφαρμογή των 2 μεθόδων. Το σήμα του άξονα Χ είναι σαφώς πιο περιεκτικό και τα αποτελέσματα επ'αυτων είναι πολύ καλά και με τις 2 μεθόδους ανίχνευσης.



Εικόνα 96-Σήμα resp2_10_mig.txt. Εφαρμογή της μεθόδου 1.



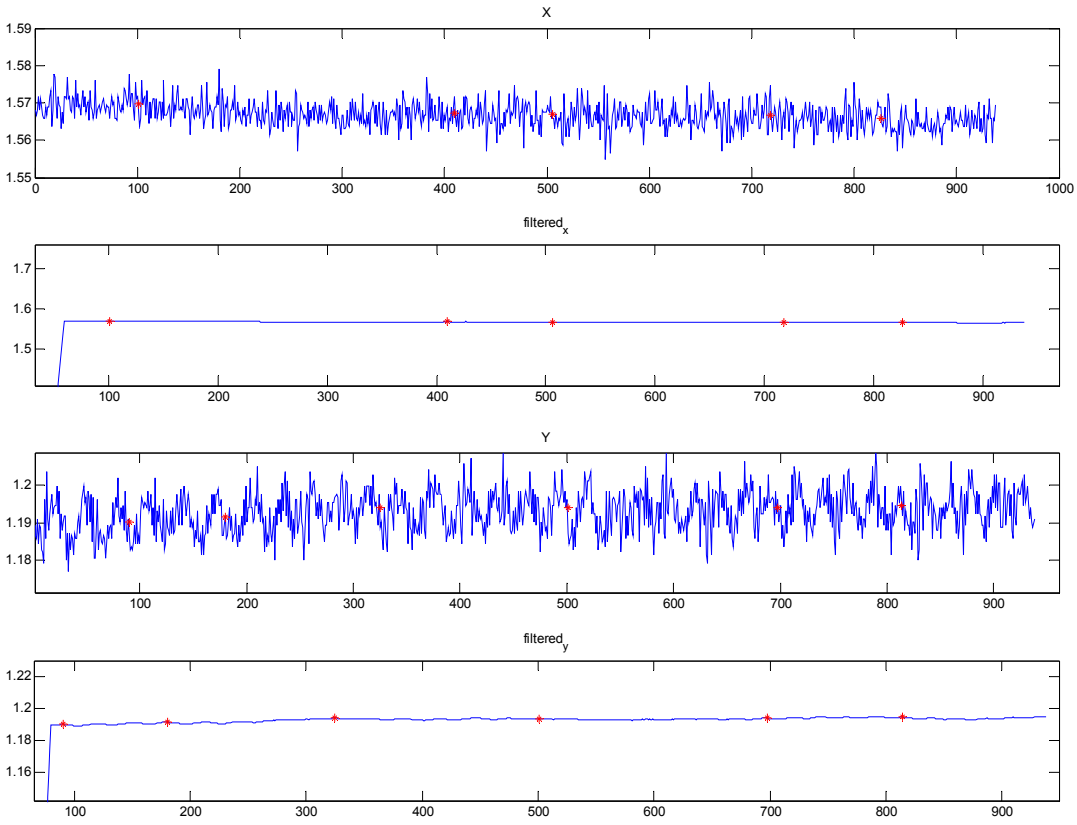
Εικόνα 97-Σήμα resp2_10-mig.txt. Εφαρμογή της μεθόδου 2.

Άπνοια

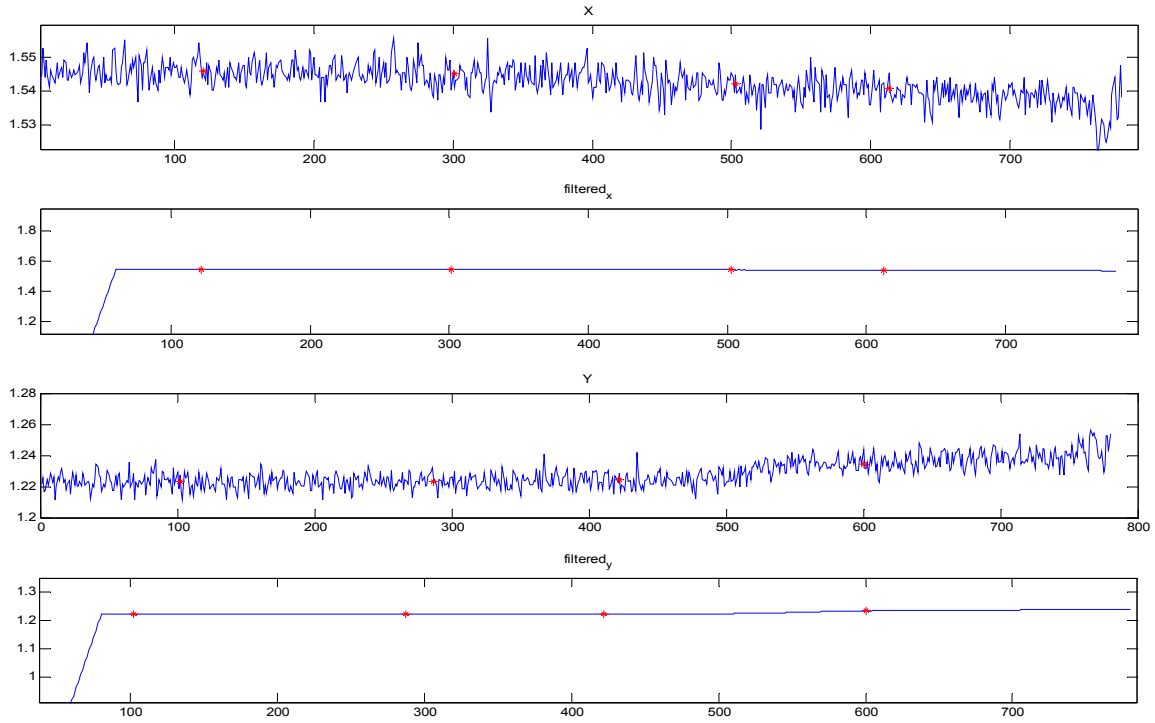
Άπνοια είναι το φαινόμενο της απουσίας εξαερισμού (αναπνοής) όταν υπερβαίνει τα 15 sec.Υπάρχουν οι εξής 3 κατηγορίες άπνοιας.

- Κεντρική άπνοια. Οφείλεται στην απουσία ή καταπίεση του σήματος πού διεγείρει τους μυείς του αναπνευστικού συστήματος
- Παρεμποδισμένη Άπνοια. Η θέληση για αναπνοή υπάρχει αλλά δεν είναι δυνατή γιατί το σύστημα φράσσεται από εμπόδια.
- Μικτή άπνοια. Ένας συνδυασμός των ανωτέρω 2 φαινόμενων.

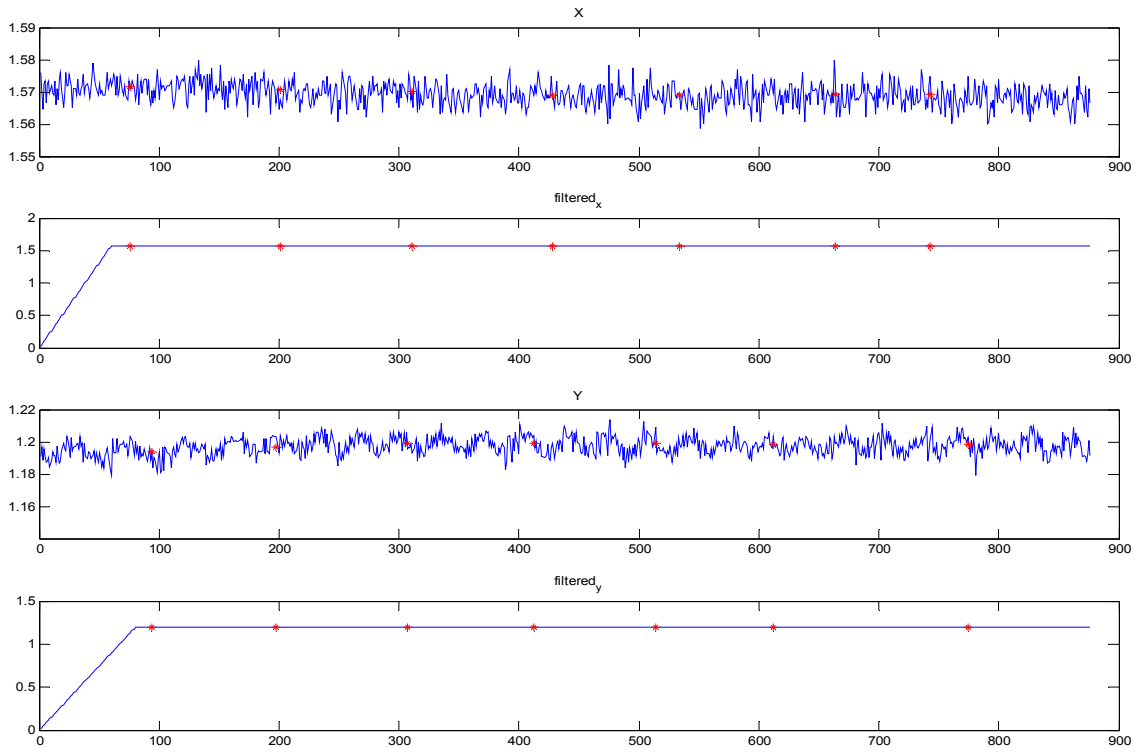
Κατά τη διεξαγωγή των μετρήσεων πήραμε μερικές ενδεικτικές μετρήσεις άπνοιας οι οποίες φαίνονται ακολούθως.



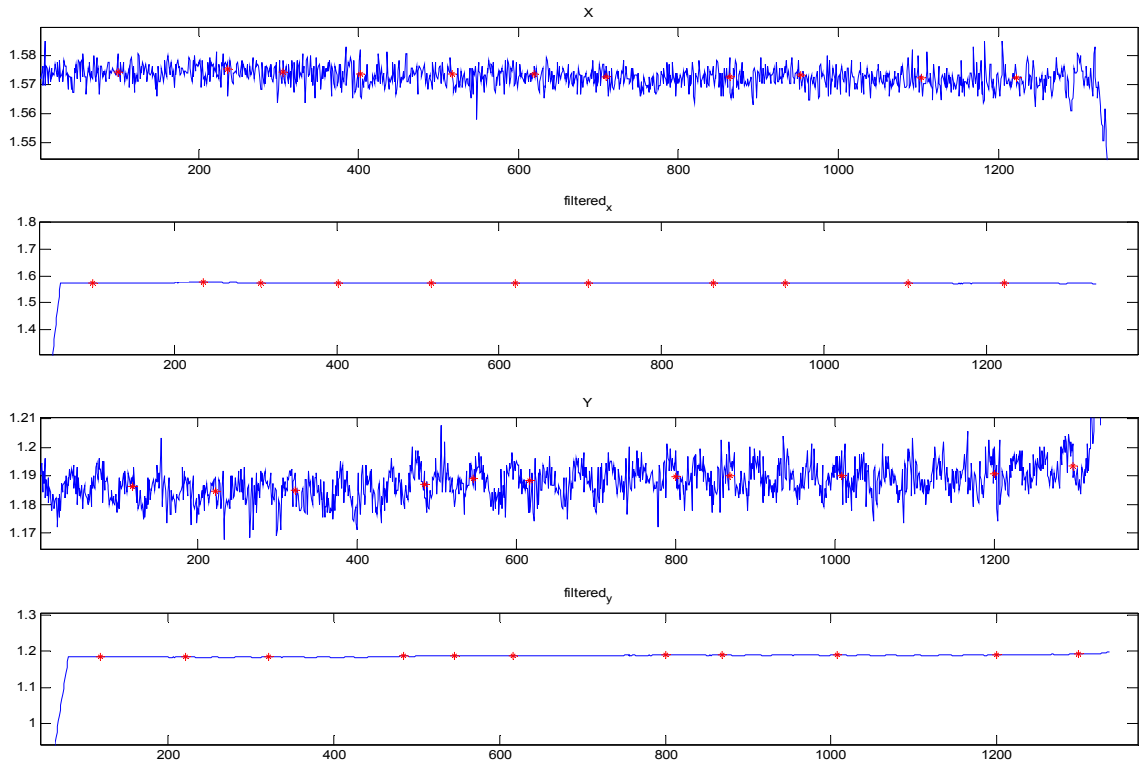
Εικόνα 98-Σήμα apnea2_listen.txt



Εικόνα 99-Σήμα apnea1_listen.txt



Εικόνα 100-Σήμα apnea3_mig.txt



Εικόνα 101-Σήμα apnea4_mig.txt

Παρατηρήσεις

Παρατηρούμε ότι αν και στα αρχικά σήματα φαίνονται κάποιες μεταβολές της τάσεις αυτές οφείλονται στις συσπάσεις των μυών που καταγράφει το επιταχυνσιόμετρο και όχι σε αναπνοές και διαστολή του θώρακα και του διαφράγματος.

Οι μετρήσεις αυτές μπορούν να χρησιμοποιηθούν για να καθαρίσουμε τα αρχικά σήματα από το θόρυβο. Μπορούμε δηλαδή να αφαιρέσουμε από ένα καλό σήμα που περιέχει αναπνοές το σήμα της άπνοιας που θεωρητικά περιέχει μόνο θόρυβο οπότε αυτό που απομένει είναι η καθαρή κυμματομορφή της αναπνοής .Σε μερικές δοκιμές που έγιναν στο Matlab τα αποτελέσματα που προέκυψαν δεν επιβεβαίωσαν αυτή τη θεωρία.

Αξιολόγηση των δύο μεθόδων Ανίχνευσης της Αναπνοής

Για την αξιολόγηση των 2 διαφορετικών μεθόδων ανίχνευσης αναπνοών που χρησιμοποιήσαμε θα εφαρμόσουμε μια στατιστική ανάλυση που βασίζετε στις εξής 3 κατηγορίες ανιχνεύσεων

- **TP** – True Positive. Αφορά τις αναπνοές που δεδομένα υπάρχουν στο σήμα και ανιχνεύονται από την εκάστοτε μέθοδο.
- **FP** – False Positive .Αφορά τις αναπνοές που δεδομένα δεν υπάρχουν στο σήμα αλλά ανιχνεύονται από την εκάστοτε μέθοδο.
- **FN** –False Negative.Αφορά τις αναπνοές που δεδομένα υπάρχουν στο σήμα αλλά δεν ανιχνεύονται από την εκάστοτε μέθοδο.

Για την αξιολόγηση του κάθε αλγορίθμου τα αποτελέσματα καταγράφοντα ως προς την ευαισθησία (Sensitivity) **Se** και την θετική προβλεπτικότητα(Positive Predictivity) **+P** του κάθε αλγορίθμου. Η χρήση της Positive Predictivity διασφαλίζει ότι ψηλές τιμές της ευαισθησίας που οφείλονται σε ψηλούς λόγους **FP** θα αναγνωρίζονται.

- **Se** (Ευαισθησία) Το κλάσμα των πραγματικών γεγονότων (αναπνοές) που ανιχνεύονται ορθά.

$$Se = \frac{TP}{TP + FN} \times 100$$

- **+P** (Positive Predictivity) Το κλάσμα των ανιχνεύσεων που είναι πραγματικά γεγονότα (αναπνοές).

$$+P = \frac{TP}{TP + FN} \times 100$$

Στον πίνακα που ακολουθεί τοποθετηθήκαν όλα τα σήματα από τις μετρήσεις που διεξάχθηκαν οι περισσότερες από τις οποίες φαίνονται και γραφικά στα προηγούμενα ,καθώς και συμπεριφορά των μεθόδων ανίχνευσης ως προς αυτές. Στην πρώτη στήλη είναι το όνομα του κάθε σήματος. Στη δεύτερη οι 2 άξονες στους

οποίους χωρίζεται το σήμα. Στην τρίτη στήλη οι δεδομένες αναπνοές που καταγράφηκαν κατά τη διαδικασία της μέτρησης και στις στήλες 4 και 5 οι αναπνοές που ανιχνεύει η δεύτερη και πρώτη μέθοδος αντίστοιχα. Στη στήλη 6 αναγράφεται το παράθυρο φιλτραρίσματος που χρησιμοποιήσαμε για τον καθαρισμό του κάθε σήματος πριν την επεξεργασία. Στις υπόλοιπες στήλες βρίσκουμε τα True Positives ,False Positives και False Negatives για την κάθε μέθοδο με την σειρά. Στην τελευταία γραμμή του πίνακα βρίσκονται τα αντίστοιχα αθροίσματα που θα χρησιμοποιηθούν στη μέθοδο.

Για τα αρχεία μετρήσεων resp1_listen.txt, resp2_listen.txt παρουσιάστηκε πρόβλημα προσπέλασης των δεδομένων τους σε επίπεδο επεξεργασίας και δεν αναλύθηκαν αλλά παρουσιάζονται για λόγους συνεχόμενης αρίθμησης.

		Μετρούμε Αναπνοέ	**Resp Counter Method2**	**Resp Counter Method1**	**moving_ average wi (X,Y)**	**TP_2**	**FP_2**	**FN_2**	**TP_1**	**FP_1**	**FN_1**
resp1_ txt	X										
	Y										
resp2_ txt	X										
	Y										
resp3_ txt	X	10	5	3	50	5		5	3		7
	Y		6	4	50	6		4	4		6
resp4_ txt	X	11	10	11	50	10		1	11		0
	Y		11	18	50	11		0	11	7	
resp5_ txt	X	5	5	2	80	5		0	2		3
	Y		3	6	80	3		2	5	1	
resp6_ txt	X	7	6	1	80	6		1	1		6
	Y		7	7	80	7		0	7		0
resp11_ n.txt	X	15	14	5	60	14		1	5		10
	Y		14	19	80	14		1	15	4	
resp6_ t	X	10	9	7	60	9		1	7		3
	Y		10	15	80	10		0	10	5	
resp7_ t	X	7	6	7	60	6		1	7		0
	Y		7	14	80	7		0	7	7	0
resp8_ t	X	4	3	2	60	3		1	2		2
	Y		3	3	80	3		1	3		1

resp9_t	X	26	25	24	60	25		1	24		2
	Y		26	49	80	26		0	26	23	
resp10_xt	X	20	22	10	60	20	2		10		10
	Y		22	27	80	20	2		20	7	
resp1_n.txt	X	12	12	11	60	12		0	11		1
	Y		12	22	80	12		0	12	10	
resp1_n.txt	X	9	9	1	60	9		0	1		8
	Y		9	9	80	9		0	9		
resp1_n.txt	X	17	16	12	60	16		1	12		7
	Y		16	29	80	16		1	17	12	
resp1_n.txt	X	8	8	5	60	8		0	5		3
	Y		8	12	80	8		0	8	4	
resp1_n.txt	X	16	16	13	60	16		0	13		3
	Y		16	29	80	16		0	16	13	
resp2_n.txt	X	7	5	3	60	5		2	3		4
	Y		4	6	80	4		3	6		1
resp2_n.txt	X	9	8	6	60	8		1	6		3
	Y		8	15	80	8		1	9	6	
resp2_n.txt	X	23	22	8	60	22		1	8		15
	Y		22	30	80	22		1	23	7	
resp2_n.txt	X	17	18	6	60	17	1		6		11
	Y		15	20	80	15		2	17	3	
resp2_n.txt	X	10	9	3	60	9		1	3		7
	Y		8	12	80	8		2	10	2	
resp1_.txt	X	4	4	2	60	4		0	2		2
	Y		4	5	80	4		0	4	1	
resp1_.txt	X	3	3	3	60	3		0	3		
	Y		3	5	80	3		0	3	2	
resp1_.txt	X	33	33	18	60	33		0	18		15
	Y		33	52	80	33		0	33	19	
resp1_.txt	X	24	24	5	60	24		0	5		19
	Y		24	27	80	24		0	24	3	
resp1_g.txt	X	4	3	2	60	3		1	2		2

	Y		4	5	80	4		0	4	1	
resp2_(.txt	X	11	10	11	60	10		1	11		
	Y		11	22	80	11		0	11	11	
resp2_ .txt	X	10	10	9	60	10		0	9		1
	Y		10	18	80	10		0	10	8	
resp2_ .txt	X	18	20	1	60	18	2		1		17
	Y		18	17	80	18		0	17		1
resp2_ .txt	X	2	1	1	60	1		1	1		1
	Y		2	3	80	2		0	2	1	
resp2_ g.txt	X	26	25	25	60	25		1	25		1
	Y		25	32	80	25		1	26	6	
Αθροισ						**715**	**7**	**41**	**586**	**163**	**172**

Έτσι για την δεύτερη μέθοδο προκύπτει

$$Se = \frac{TP}{TP + FN} \times 100 = \frac{715}{715 + 41} \times 100 = 95\%$$

$$+P = \frac{TP}{TP + FN} \times 100 = \frac{715}{715 + 7} \times 100 = 99\%$$

Ενώ για τη πρώτη

$$Se = \frac{TP}{TP + FN} \times 100 = \frac{586}{586 + 172} \times 100 = 77\%$$

$$+P = \frac{TP}{TP + FN} \times 100 = \frac{586}{586 + 163} \times 100 = 78\%$$

Είναι ξεκάθαρο από τα αποτελέσματα ότι της δεύτερης μεθόδου είναι πολύ πιο αξιόπιστα από αυτά της πρώτης. Η διαφορά των 18 ποσοστιαίων μονάδων στην ευαισθησία αλλά και η μεγάλη διαφορά στη θετική προβλεπτικότητα το φανερώνουν ξεκάθαρα.

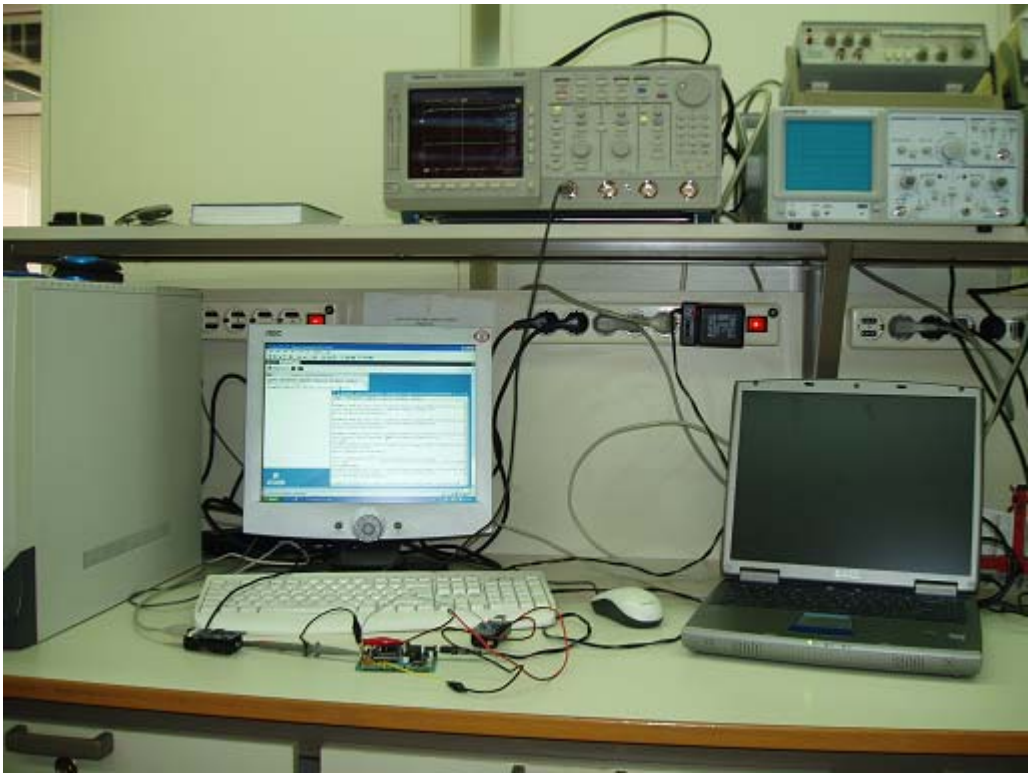
Παρατηρήσεις.

Μπορεί να παρατηρηθεί από τα σήματα που παρατέθηκαν πιο πριν και από τα αποτελέσματα που φαίνονται στον πίνακα ότι πολλά σήματα παρουσίασαν σε πολλές περιπτώσεις περίεργη συμπεριφορά. Πολλά ενώ έχουν ξεκάθαρη πληροφορία στον ένα άξονα στον άλλο δεν έχουν καμία πληροφορία. Αυτό οφείλεται κατά βάση στη θέση του σώματος που έγινε η μέτρηση. Σε άλλα παρατηρείται μεταβολή της μέσης τιμής είτε θετική είτε αρνητική που κατά βάση επηρέαζε την πρώτη μέθοδο ανίχνευσης. Η μεταβολή αυτή στις περισσότερες των περιπτώσεων οφείλετε σε μετατόπιση του evaluation board και του επιταχυνσιομέτρου κατά τη διάρκεια της μέτρησης. Σε αρκετά σήματα παρατηρούνται απότομες αυξήσεις ή και βυθίσεις της τάσης που προέρχονται από χασμουρητά , βήχα ή και φτερνίσματα κατά τη μέτρηση. Ένα πρόβλημα ήταν και ο συγχρονισμός της εκκίνησης του προβληματικού Serial Forwarder με την μέτρηση των αναπνοών στο δείγμα από τον ‘ασθενή’.

Όσον αφορά τις μεθόδους ανίχνευσης των αναπνοών στην πρώτη που λαμβάνει ως παράμετρο την μέση τιμή του σήματος επί κάποιο συντελεστή αποδείχτηκε ότι για σήματα που η μέση τιμή τους έχει μεγάλες αποκλίσεις δεν δίνει καλά αποτελέσματα. Η γενικευμένη χρήση της μεθόδου σε όλα τα σηματα κρίνετε μη ικανοποιητική λόγω των διαφορών στις στατιστικές παραμέτρους του κάθε σήματος ξεχωριστά. Η βελτίωση στην απόδοση της μεθόδου θα μπορούσε να επιτευχθεί εφόσον τα σήματα αναλύονταν ως προς τα στατιστικά του χαρακτηριστικά και προσαρμόζοντας τη μέθοδο με βάση αυτά.

```
mean1 = mean(decvalue_x);
n = length(decvalue_x);
stdev = sqrt(sum((decvalue_x-mean1).^2/n));
y_up = mean1 + 0.5*stdev;
```

Στη δεύτερη μέθοδο που κατά γενική ομολογία παρουσίασε πολύ καλύτερα αποτελέσματα. Ενα από τα σημεία που σε πολλές περιπτώσεις έδωσε λανθασμένα αποτελέσματα και η απόκλιση των ανιχνευμένων αναπνοών από τοην πραγματικό αριθμό ήταν μία αναπνοή ,αφορά τα τελευταία δείγματα. Κατά τον χωρισμό του σήματος σε παράθυρα των 100 δειγμάτων τα τελευταία δείγματα αγνοούνται. Σε πολλές περιπτώσεις μέσα στο τελευταίο αυτό παράθυρο μπορεί να περιέχονται 70-80 δείγματα και να αποτελούν μια πλήρη αναπνοή η οποία αγνοείται.



Εικόνα 102-Στη φωτογραφία αυτή βλέπουμε το evaluation board συνδεδεμένο στο mote και στον παλμογράφο.Ο υπολογιστής καταγράφει τα πακέτα που στέλνει το basestation.

Γενικά Συμπεράσματα

Στα πλαίσια αυτής της εργασίας έγινε μια μελέτη για την μη επεμβατική καταγραφή του βιοσήματος αναπνοής με τη χρήση ενός διαξονικού επιταχυνσιόμετρου και η ασύρματη μετάδοση του μέσα από ένα δίκτυο αισθητήρων προς τον σταθμό βάσης. Η εφαρμογή δεν περιορίστηκε απλά στη μετάδοση του σήματος αλλά και στην απεικόνιση και περαιτέρω επεξεργασία του στον υπολογιστή, ο οποίος και ήταν ο τελικός αποδέκτης. Η υλοποίηση της εφαρμογής έγινε με χρήση και κατάλληλο προγραμματισμό της ασύρματης πλατφόρμας Tmote Sky (Moteiv). Το σύστημα που υλοποιήθηκε σαν εφαρμογή για τη μετάδοση βιοσημάτων μπορεί φυσικά να επεκταθεί και σε περισσότερες συσκευές καταγραφής, πλην της αναπνοής όπως ηλεκτροκαρδιογράφους, θερμόμετρα, πιεσόμετρα, ανιχνευτές κίνησης κ.α.

Τα βασικά πλεονεκτήματα των μονάδων ασύρματων αισθητήρων είναι ότι συνδυάζουν δυνατότητες αποθήκευσης, επεξεργασίας και επικοινωνίας έτσι ώστε να επιτυγχάνουν την εκτέλεση πολλαπλών διεργασιών και να μπορούν να χρησιμοποιηθούν σε ένα πλήθος εφαρμογών, ύστερα από προγραμματισμό τους, ικανοποιώντας τις απαιτήσεις των σχεδιαστών τέτοιων δικτύων.

Μάλιστα, όλα αυτά τα χαρακτηριστικά προσφέρονται με τη μικρότερη δυνατή κατανάλωση ενέργειας και το μικρότερο δυνατό κόστος. Κάποια από τα χαρακτηριστικά που τίθενται ως βασικοί στόχοι υλοποίησης τέτοιων δικτύων είναι ο χρόνος ζωής, δεδομένου ότι πολλά από τα δίκτυα αυτά τροφοδοτούνται από μπαταρίες, το μέγεθος κάλυψης και η επεκτασιμότητα, το κόστος παραγωγής και η ευκολία ανάπτυξης, η αντοχή σε σφάλματα, οι δυνατότητες συγχρονισμού και ο χρόνος απόκρισης σε συμβάντα, καθώς και η ασφάλεια που παρέχουν στον τελικό χρήστη. Οι εφαρμογές τέτοιων συστημάτων δεν περιορίζονται φυσικά στην καταγραφή βιοσημάτων αλλά έχουν ένα τεράστιο εύρος.

Μέσα από την περιγραφή των κυριότερων πλατφορμών που υλοποιούν τα σημερινά δίκτυα αισθητήρων, επικεντρωθήκαμε στην πλατφόρμα Tmote Sky (Moteiv), η οποία είναι και η μονάδα που χρησιμοποιήσαμε για την εφαρμογή. Η μονάδα tmote χρησιμοποιεί το πρότυπο **IEEE 802.15.4** για την επικοινωνία, εκπέμποντας στη συχνότητα των 2.4GHZ με μέγιστη ταχύτητα μετάδοσης δεδομένων τα 250 Kbps. Τα σημαντικά τεχνικά χαρακτηριστικά του, η χαμηλή κατανάλωση ισχύος, ο μικρός χρόνος αφύπνισης, οι μεγάλες δυνατότητες

επέκτασης και η ευχρηστία του οδήγησε στην επιλογή της για την υλοποίηση της εφαρμογή.

Η λειτουργία του Tmote Sky βασίζεται στο λειτουργικό σύστημα **TinyOS**, σχεδιασμένο ειδικά για ενσωματωμένα συστήματα. Όλη η οργανωτική δομή του **TinyOS**, οι βιβλιοθήκες και οι εφαρμογές είναι γραμμένες, όπως είπαμε, στη γλώσσα προγραμματισμού **NesC**. Στην εργασία μας περιγράψαμε αναλυτικά τις σημαντικότερες ιδιότητες τόσο του **TinyOS-2.X** όσο και της **NesC**, καθώς στη συνέχεια χρειάστηκε να προγραμματίσουμε την πλατφόρμα Tmote Sky με αυτή τη γλώσσα ώστε να υλοποιήσουμε την εφαρμογή.

Η εφαρμογή απαιτούσε την δειγματοληψία των αναλογικών εξόδων της πλατφόρμας αξιολόγησης ενός επιταχυνσιόμετρου 2 αξόνων Memsic. Όπως αναλύθηκε πιο πριν το επιταχυνσιόμετρο συνδέθηκε με τις θύρες ADC1 και ADC2 του mote καθώς και τη γείωση λαμβάνοντας μετρήσεις με κατάλληλη συχνότητα δειγματοληψίας που εμείς ορίσαμε (20 και 50 Hz) και με την κατάλληλη μορφοποίηση των δεδομένων τα αποστέλλει προς ένα σταθμό βάσης και στην συνέχεια στον υπολογιστή μέσω των κατάλληλων εργαλείων για να εφαρμοστεί το στάδιο της επεξεργασίας.

Εκτός από την αναλυτική περιγραφή του κώδικα της εφαρμογής για το τμήμα αποστολής του σήματος, στη συνέχεια επικεντρωθήκαμε στο τμήμα λήψης και στη σωστή απεικόνιση και επεξεργασία του. Όπως αναφέραμε, το σήμα αποστελλόταν ασύρματα σε ένα σταθμό βάσης, δηλαδή στη δεύτερη μονάδα tmote, συνδεδεμένη με τον υπολογιστή μέσω της θύρας USB. Περιγράφηκε η εφαρμογή με την οποία ήταν προγραμματισμένη η συγκεκριμένη μονάδα, η οποία προωθούσε στη σειριακή θύρα του υπολογιστή τα πακέτα που λάμβανε ασύρματα και στη συνέχεια αναφερθήκαμε στα εργαλεία που χρησιμοποιήσαμε για την απεικόνιση του σήματος (Listen, Oscilloscope, SerialForwarder) και του τρόπου που αυτά χειρίζονται τα λαμβανόμενα πακέτα.

Όσον αφορά το σήμα καταγραφής δηλαδή το σήμα αναπνοής όπως καταγράφεται σαν τάση από το επιταχυνσιόμετρο από τις μετατοπίσεις του στήθους και της κοιλιακής χώρας. Εξηγήθηκε η λειτουργία του επιταχυνσιόμετρου και ο τρόπος λήψης την σημάτων ξεχωριστά για κάθε άξονα. Διαπιστώθηκε η επίδραση του θορύβου στο σήμα και αναλύθηκαν μερικές πιθανές πηγές θορύβου καθώς και ο τρόπος που επηρεάζουν το σήμα. Το σήμα που λαμβάναμε τελικά στον υπολογιστή, έφτανε σε μορφή πακέτων που έπρεπε να επεξεργαστούν. Επιπλέον, περιείχε

κάποιες συνιστώσες θορύβου οι οποίες έπρεπε να ελαχιστοποιηθούν μέσω κατάλληλης επεξεργασίας του, ώστε να βελτιωθεί η διαγνωστική του αξία. Για το λόγω αυτό υλοποιήθηκε, σε περιβάλλον Matlab, κώδικας ο οποίος διαχειρίζεται κατάλληλα τα εισερχόμενα πακέτα δεδομένων, ξεχωρίζει τις τιμές των δειγμάτων του ADC για κάθε άξονα ξεχωριστά που αποτελούν το σήμα μας και στη συνέχεια τα επεξεργάζεται.

Για το φιλτράρισμα επιλέχθηκε ο αλγόριθμος κινούμενου μέσου όρου, καθώς αποδείχθηκε ότι έδινε ικανοποιητικά αποτελέσματα. Στη συνέχεια υλοποιηθήκαν και εφαρμόστηκαν σε περιβάλλον Matlab δύο αλγόριθμοι για την ανίχνευση και καταγραφή των αναπνοών που περιέχει το κάθε σήμα. Ο πρώτος αλγόριθμος ανιχνεύει τις κορυφές(peaks) σήματος σε σχέση με την μέση του τιμή επί κάποιο συντελεστή ενώ ο δεύτερος αλγόριθμος αφού χωρίζει τα σήματα σε παράθυρα ανιχνεύει σε κάθε παράθυρο την μέγιστη τιμή και με το κριτήριο της παραγώγου αποφασίζει αν πρόκειται για αναπνοή. Ακολούθως με την κατάλληλη στατιστική διεργασία αξιολογήσαμε την αποτελεσματικότητα του κάθε αλγορίθμου ξεχωριστά όπου διαπιστόθηκε ότι ο δεύτερος είχε σαφώς καλύτερα αποτέλεσμα από τον πρώτο.

Επιπλέον έγινε και διασύνδεση του παλμογράφου και του υπολογιστή μέσω GPIB διεπαφής, με τη συγγραφή κατάλληλου κώδικα στο Matlab, ώστε να μπορούμε να αποθηκεύουμε τις μετρήσεις από τον παλμογράφο, όποτε αυτό κρινόταν αναγκαίο.

Προεκτάσεις

Κατά την τριβή με το αντικείμενο στα πλαίσια της εργασίας βρέθηκαν πολλά σημεία τα οποία θα μπορούσαν να βελτιώσουν σημαντικά την εφαρμογή και αρκετές ιδέες-προτάσεις οι οποίες μπορούν να υλοποιηθούν προς προέκταση και πειραματισμό.

Σημαντική βελτίωση μπορεί να σημειωθεί όσον αφορά το επιταχυνσιόμετρο. Μπορεί να χρησιμοποιηθεί στην παρούσα εφαρμογή ένα επιταχυνσιόμετρο 3 αξόνων με μεγαλύτερη ίσως ευαισθησία από το παρόν. Ο κώδικας της εφαρμογής υποστηρίζει το τριαξονικό επιταχυνσιόμετρο χωρίς καμιά τροποποίηση.

Ακόμα σημαντική μείωση του όγκου της πλατφόρμας αξιολόγησης θα το καταστήσει πολύ πιο εύχρηστο και αξιόπιστο στις μετρήσεις του. Με την τοποθέτηση της πλατφόρμας αξιολόγησης σε μια ειδική ζώνη που θα μπορεί να φορά ο ασθενής ώστε να μην έχουμε μετακινήσεις της πλατφόρμας που να επηρεάζουν το σήμα. Έτσι ο ασθενής θα μπορεί να φορά αν χρειάζεται το επιταχυνσιόμετρο στο σώμα του 24 ώρες τη μέρα. Φυσικά έχουν εξελιχθεί motes που μπορούν να ενσωματώσουν την πλατφόρμα αξιολόγησης στο mote χωρίς καλώδια τα οποία αποτελούν περιοριστικό παράγοντα κατά τη διάρκεια της πειραματικής διαδικασίας .Έτσι ο ασθενής θα μπορεί να έχει εφαρμοσμένο πάνω του μόνο το mote το οποίο θα αποτελείται και από το επιταχυνσιόμετρο.

Σαν προέκταση της όλης διαδικασίας εκτός από το βιοσήμα αναπνοής μπορούμε να καταγράφουμε ταυτόχρονα και άλλα ζωτικά βιοσήματα όπως το ηλεκτροκαρδιογράφημα, τη θερμοκρασία του ασθενούς, την αρτηριακή του πίεση, τον κορεσμό του οξυγόνου στο αίμα. Ο συνδυασμός και η ταυτόχρονη παρακολούθηση όλων αυτών των σημάτων με μη επεμβατικό τρόπο μπορεί να δώσει μια πλήρη εικόνα της κατάστασης του ασθενούς ανά πάσα ώρα.

Η εφαρμογή θα μπορούσε να επεκταθεί με τροποποίηση του κώδικα ώστε η πλατφόρμα Tmote Sky να συνδυάζει ταυτόχρονα τη χρήση περισσότερων από έναν αισθητήρων, να συνδυάζει τα δεδομένα που λαμβάνει και να εκτελεί κάποιου είδους επεξεργασία πριν τα μεταδώσει ασύρματα. Αυτή η επεξεργασία θα μπορούσε να περιλαμβάνει και την εκτέλεση αλγορίθμων για την εξαγωγή σημαντικών παραμέτρων ή ακόμα και την αναγνώριση επικίνδυνων καταστάσεων, με βάση συγκεκριμένα κριτήρια που θα έχουν τεθεί. Έτσι, για παράδειγμα, ο κάθε κόμβος θα μπορούσε εκτός από το να στέλνει συνεχώς δεδομένα σε κάποιο σταθμό βάσης, να τα επεξεργάζεται πρώτα και να μεταδίδει ασύρματα μόνο συγκεκριμένες τιμές που προκύπτουν από την επεξεργασία αυτή ή να ειδοποιεί για καταστάσεις κινδύνου όταν αυτό απαιτείται.

Η πρακτική αυτή θα μπορούσε φυσικά να μειώσει την κατανάλωση ενέργειας, εφόσον το πιο ενεργοβόρο τμήμα ενός τέτοιου συστήματος είναι το τμήμα μετάδοσης-λήψης δεδομένων. Επιπλέον θα μπορούσε να μειώσει πιθανή συμφόρηση του δικτύου λόγω αυξημένης κίνησης δεδομένων. Βέβαια η ενσωμάτωση τοπικής επεξεργασίας σήματος και η παράλληλη μετάδοσή τους αυξάνει και την χρήση του επεξεργαστή, ενώ εισάγει και επιπλέον παράγοντες ως προς την συμπεριφορά και τις απαιτήσεις, θέματα που οφείλει να εξετάσει ο

σχεδιαστής ενός συστήματος προτού υλοποιήσει την εφαρμογή του, αντισταθμίζοντας τα οφέλη και τα μειονεκτήματα που μπορεί να προκύψουν από μια τέτοια κίνησή για το συγκεκριμένο τύπο της εφαρμογής που καλείται να υλοποιήσει.

Επίσης, ο κώδικας της εφαρμογής θα μπορούσε να τροποποιηθεί ώστε να υποστηρίζει multihop επικοινωνία μεταξύ των κόμβων. Η multihop επικοινωνία είναι ένα από τα σημαντικότερα χαρακτηριστικά των ασύρματων δικτύων αισθητήρων. Με τη χρήση multi-hop τεχνικών είναι εφικτή η επέκταση της κάλυψης αρκετά πιο μακριά από την ακτίνα που επιτρέπει ο χρησιμοποιούμενος πομπός. Μπορεί επίσης να βελτιώσει την ενεργειακή συμπεριφορά του δικτύου καθώς στηρίζεται στην επικοινωνία μεταξύ των κόμβων για τη μετάδοση των δεδομένων στον τελικό προορισμό. Δίνει τη δυνατότητα, μέσα από κατάλληλο σχεδιασμό και υλοποίηση αλγορίθμου, δημιουργίας αυτοδικτυούμενων συστημάτων αισθητήρων με δυνατότητα αποκατάστασης χαμένων κόμβων και αυτόματης ενσωμάτωσης νέων μελών στο δίκτυο.

Απαραίτητη είναι προς αυτή την κατεύθυνση και η μελέτη και υλοποίηση αλγορίθμων που θα επιτρέπουν αξιόπιστη και γρήγορη δρομολόγηση δεδομένων προς τον τελικό κόμβο-προορισμό. Η υλοποίηση αυτών των αλγορίθμων μπορεί να δώσει δυναμικό χαρακτήρα στο δίκτυο ώστε να ανταπεξέρχεται εύκολα στην μεταβολή της θέσης των κόμβων ή στην περίπτωση απώλειας κάποιων από αυτών. Με τον τρόπο αυτό, ένας γιατρός σε κάποιο χώρο νοσηλείας θα μπορούσε να κινείται μέσα στο κτίριο και να έχει ανά πάσα στιγμή διαθέσιμες σε μια φορητή συσκευή τις πληροφορίες που αφορούν κάποιο συγκεκριμένο ασθενή. Επίσης ένας ασθενής, θα μπορούσε να κινείται μέσα στο χώρο του νοσοκομείου ή ακόμα και του σπιτιού του και η συνδεδεμένη πάνω σε αυτόν μονάδα αισθητήρα να μεταδίδει συνεχώς τα δεδομένα που καταγράφει σε κάποιο σταθμό βάσης, μέσα από τη δρομολόγησή τους στους κατάλληλους κόμβους κάθε φορά. Όλα τα παραπάνω μπορούν να αποτελέσουν αντικείμενο μελέτης και πειραματισμού προς επέκταση της εφαρμογής.

Τέλος πρέπει να εξεταστεί και το φαινόμενο της κατανάλωσης ενέργειας, καθώς μια αύξηση της συχνότητας δειγματοληψίας οδηγεί και σε αύξηση του duty cycle της μονάδας. Βέβαια, σε περίπτωση τροφοδότησης από μια σταθερή πηγή αυτό ίσως δεν μας απασχολεί αρκετά, αλλά σε περιπτώσεις που είναι αναγκαία η φορητότητα και η μεγάλη διάρκεια αυτονομίας τότε πρέπει να ληφθεί υπόψη. Κάποιες ακόμα προτάσεις που μπορούν να υλοποιηθούν είναι η συγγραφή κώδικα

που θα επιτρέπει την επικοινωνία ενός χρήστη του ασύρματου δικτύου αισθητήρων με συγκεκριμένους κόμβους του δικτύου, για τη συλλογή πληροφοριών αλλά και τη διαχείριση της λειτουργίας τους. Η προσομοίωση του δικτύου μέσω του εργαλείου που παρέχει το **TinyOS** για αυτό το σκοπό, το TOSSIM, καθώς και η οπτικοποίηση του μέσω του TinyViz αποτελούν επίσης θέματα που μπορούν να εξεταστούν.

Μια βελτίωση που έγινε προσπάθεια να επιτευχθεί χωρίς επιτυχία είναι η χρήση της εφαρμογής Oscilloscope για την οπτικοποίηση των λαμβανόμενων δεδομένων. Με τον τρόπο αυτό θα ήταν δυνατή η παρατήρηση του σήματος σε πραγματικό χρόνο (Real time) και ο καθορισμός της συχνότητας δειγματοληψίας χωρίς να επεμβαίνουμε κάθε φορά στα αρχεία της εφαρμογής. Η προσθήκη αυτή του Oscilloscope μπορεί να γίνει εφικτή με την κατάλληλη προσαρμογή του κώδικα Java της εφαρμογής Oscilloscope.

Όσον αφορά την επεξεργασία του σήματος, προτείνεται η διασύνδεση του Matlab με το **TinyOS** και του δικτύου αισθητήρων για επεξεργασία σε πραγματικο χρόνο και απεικόνιση δεδομένων κατευθείαν μέσα από το περιβάλλον του Matlab.

Παράλληλα, όσον αφορά τους αλγόριθμους ανίχνευσης της αναπνοής, αν και παρουσίασαν σημαντική αξιοπιστία ειδικά ο δεύτερος όπως αποδείχθηκε στο μεγαλύτερο τμήμα των μετρήσεων που πραγματοποιήσαμε, μπορεί να βελτιωθούν περαιτέρω με την ενσωμάτωση ιδιοτήτων αυτόματης προσαρμογής, σε πραγματικό χρόνο, στο σήμα εισόδου για την εξαγωγή ακόμα πιο αξιόπιστων αποτελεσμάτων. Έχουν ήδη επισημανθεί και αναφερθεί λόγοι και αδυναμίες του κάθε αλγόριθμου ξεχωριστά και πως επηρεάζουν τα αποτελέσματα. Ήδη στη διεθνή βιβλιογραφία έχει προταθεί ένα πλήθος τέτοιων αλγορίθμων που εμφανίζουν αυξημένα ποσοστά επιτυχούς ανιχνευσης.

Επιπλέον, η συσχέτιση του σήματος για τον κάθε άξονα, ειδικά όταν υπάρχει μια πιο ολοκληρωμένη εικόνα του σήματος και στους 3 άξονες, είναι μια προτεινόμενη βελτίωση.

Τέλος, η δημιουργία λογισμικού που θα επιτρέπει τη χωροταξική γραφική απεικόνιση του δικτύου (localization) και τον χωρικό εντοπισμό του κάθε κόμβου, ξεχωριστά είναι μια περαιτέρω προσθήκη στις δυνατότητες της εφαρμογής. Μάλιστα, το θέμα αυτό έχει γίνει αντικείμενο έρευνας επιστημονικών ομάδων, ενώ μπορεί να βρει και σημαντική εφαρμογή στον τομέα της Ιατρικής περίθαλψης.

Η χρήση συστημάτων ασύρματης τηλεμετρίας αποτελεί μια πραγματικότητα εδώ και καιρό. Πολλές συσκευές έχουν αναπτυχθεί και διατίθενται στο εμπόριο,

συμπεριλαμβανομένων καρδιογράφων και άλλων συσκευών καταγραφής ζωτικών σημάτων. Όλες όμως αυτές οι συσκευές έχουν απλά σχεδιαστεί ώστε να ‘καταργήσουν’ το καλώδιο μεταξύ του ασθενή και της συσκευής απεικόνισης. Δεν προορίζονται όμως για συμμετοχή σε δίκτυο, ώστε να μπορούν να ανταλλάσουν δεδομένα, να επικοινωνούν, να μεταδίδουν ταυτόχρονα σε πολλαπλούς παραλήπτες και να έχουν τη δυνατότητα χρήσης τους σε ένα πλήθος εφαρμογών, διατηρώντας παράλληλα σημαντικά μειωμένο μέγεθος και ικανότητα αυτονομίας. Εδώ είναι που τα ασύρματα δίκτυα αισθητήρων καινοτομούν.

Η τάση που επικρατεί αναδεικνύει το σημαντικό ρόλο που μπορούν να διαδραματίσουν τα δίκτυα αισθητήρων στις ιατρικές εφαρμογές αλλά και στη ζωή μας γενικότερα, στο άμεσο μέλλον. Μάλιστα, υπάρχουν κάποιες βασικές απαιτήσεις τις οποίες τα ασύρματα δίκτυα αισθητήρων μπορούν και καλούνται να ικανοποιήσουν ιδιαίτερα για τις Ιατρικές εφαρμογές. Αυτές περιλαμβάνουν την απαίτηση χρήσης μικρών, φορητών αισθητήρων, τη δυνατότητα αξιόπιστης και αποδοτικής ασύρματης επικοινωνίας, τη δυνατότητα ταυτόχρονης λήψης δεδομένων σε πολλαπλούς αποδέκτες, την ικανότητα οργάνωσης σε περιπτώσεις αλλαγής και μετακίνησης κόμβων (λόγω μετακίνησης ιατρικού προσωπικού αλλά και ασθενών) καθώς και την απαίτηση για ασφάλεια στη διακίνηση των ‘ευαίσθητων’ δεδομένων, με τη χρήση αλγορίθμων κρυπτογραφίας.

Το σίγουρο είναι ότι η προοπτική της χρήσης ασύρματων δικτύων αισθητήρων, θα μπορούσε να επιδράσει θετικά σε ένα μεγάλο πλήθος ιατρικών, και όχι μόνο, εφαρμογών και να συμβάλει στη βελτίωση της ποιότητας της ζωής του ανθρώπου.

Παραρτήματα

Παράρτημα 1

Το πρόγραμμα ReadMoistureSensorsC

```
/* Copyright (c) 2007, Ecosensory Austin Texas All rights reserved.
 *  This code funded by TX State San Marcos University. BSD license full
text at:
 * http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-
contrib/ecosensory/license.txt
 * Takes readings for two banks of sensorboard a2d12ch. Bank switching is
 * handled with the  HplMsp430GeneralIO in  ReadMoistureSensorsP.nc
 * since the IO functions are used as is, no mods.
 * There is no IO code in a2d12ch modules, only multichannel-a2d and
resource.
 * Two temp data buffers are returned with the multichannel.getData
 * interface as one shorter buffer with the samplesperbank buffer entries
 * all averaged together.

 * by John Griessen <john@ecosensory.com>
 * Rev 1.0 14 Dec 2007
 */
#include <Timer.h>
#include "ReadMoistureSensors.h"
#include "Msp430Adc12.h"

configuration ReadMoistureSensorsC {
}
implementation {
  components MainC, LedsC;
  components ActiveMessageC;
  components new AMSenderC(AM_MOISTURESENSORSMSG);
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;
  components new TimerMilliC() as Timer3;
  components ReadMoistureSensorsP;
  ReadMoistureSensorsP.Boot -> MainC;
  ReadMoistureSensorsP.Leds -> LedsC;
  ReadMoistureSensorsP.timer0 -> Timer0;
  ReadMoistureSensorsP.timer1 -> Timer1;
  ReadMoistureSensorsP.timer2 -> Timer2;
  ReadMoistureSensorsP.timer3 -> Timer3;
  ReadMoistureSensorsP.AMSend -> AMSenderC;
  ReadMoistureSensorsP.AMRadioOn -> ActiveMessageC;

  components a2d12chP, a2d12chC;
  ReadMoistureSensorsP.a2d12ch -> a2d12chP.a2d12ch;
//Msp430Adc12MultiChannel
  ReadMoistureSensorsP.AdcConfigure -> a2d12chP.AdcConfigure;
//Msp430Adc12MultiChannel
  ReadMoistureSensorsP.Resource -> a2d12chP.Resource; //ResourceRVG (first
thing to do)

  components HplMsp430GeneralIOC;
  ReadMoistureSensorsP.a2dmuxdisable -> HplMsp430GeneralIOC.Port21;
```

```
  ReadMoistureSensorsP.a2dbankselect -> HplMsp430GeneralIOC.Port23;
  ReadMoistureSensorsP.a2dsenvdd1drv -> HplMsp430GeneralIOC.ADC7;
  ReadMoistureSensorsP.a2dsenvdd2drv -> HplMsp430GeneralIOC.ADC6;
  ReadMoistureSensorsP.a2dterm2drv -> HplMsp430GeneralIOC.Port17;

}
```

Το πρόγραμμα ReadMoistureSensorsP

```
/* Copyright (c) 2007, Ecosensory Austin Texas All rights reserved.
 *  This code funded by TX State San Marcos University. BSD license full
text at:
 * http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-
contrib/ecosensory/license.txt
 * by John Griessen <john@ecosensory.com>
 * Rev 1.0 14 Dec 2007
 */
#include <Timer.h>
#include "ReadMoistureSensors.h"
#include "Msp430Adc12.h"
#include "a2d12ch.h"

module ReadMoistureSensorsP {
  // uses interface Packet;
  uses interface AMSend;
  uses interface SplitControl as AMRadioOn;
  uses interface Timer<TMilli>  as timer0;
  uses interface Timer<TMilli>  as timer1;
  uses interface Timer<TMilli>  as timer2;
  uses interface Timer<TMilli>  as timer3;
  uses interface Boot;
  uses interface Leds;
  uses interface Resource;
  uses interface Msp430Adc12MultiChannel as a2d12ch;
  uses interface HplMsp430GeneralIO as   a2dbankselect;
  uses interface HplMsp430GeneralIO as   a2dmuxdisable;
  uses interface HplMsp430GeneralIO as   a2dsenvdd1drv;
  uses interface HplMsp430GeneralIO as   a2dsenvdd2drv;
  uses interface HplMsp430GeneralIO as   a2dterm2drv;
  uses interface AdcConfigure<const msp430adc12_channel_config_t*>;
}
implementation {
  bool busy = FALSE;  //Used by AMSend
  bool bank1 = FALSE;  //Used by  a2d12ch.dataReady
  message_t pkt;   //Used by AMSend
//  message_t rmspkt; //old way
  MoistureSensorsMsg* rmspkt; //jg08jan08  needs testing.
  uint8_t pktlen; //Used by AMSend
//  uint16_t timerend = 0;   //del a.
  uint16_t timestamp = 0;
  uint16_t TOS_NODEID;

//uint8_t samplesperbank = 1;
uint8_t jiffies = JIFFIES;
uint16_t buffer[BUFFER_SIZE];   //see a2d12ch.h
uint16_t  bufferlen = BUFFER_SIZE;
```

```
//  ref volt from generator = 2.50 Volts
//CHANNEL1 {INPUT_CHANNEL_A0, REFERENCE_VREFplus_AVss}  see a2d12ch.h
//CHANNEL2 {INPUT_CHANNEL_A1, REFERENCE_VREFplus_AVss}
//CHANNEL3 {INPUT_CHANNEL_A2, REFERENCE_VREFplus_AVss}
//CHANNEL4 {INPUT_CHANNEL_A3, REFERENCE_VREFplus_AVss}
//CHANNEL5 {INPUT_CHANNEL_A4, REFERENCE_VREFplus_AVss}
//CHANNEL6 {INPUT_CHANNEL_A5, REFERENCE_VREFplus_AVss}
// adc12memctl_t  struct defined in Msp430Adc12.h
adc12memctl_t memctl[5] =
{{CHANNEL2},{CHANNEL3},{CHANNEL4},{CHANNEL5},{CHANNEL6}};
//adc12memctl_t memctl[2] = {{INPUT_CHANNEL_A1,
REFERENCE_VREFplus_AVss},{INPUT_CHANNEL_A2, REFERENCE_VREFplus_AVss}};
uint8_t numMemctl = 5; //numMemctl counts channels after the first one.


//  AdcConfigure.getConfiguration() is called from  here
// inside a2d12chP when a getData event happens.
//  We defined data
// above as const config and use it below as &config.
//We use the call a2d12chP.AdcConfigure to get config data in a2d12chP.
//  memctl and buffer lengt do not need to be defined in the a2d12chP
module.
//  They are about how much data, not the config.
  task void MoistureSensorsMsgSend();
  task void MoistureSensorsMsgBank1();
  task void MoistureSensorsMsgBank2();

  event void Boot.booted() {
    call AMRadioOn.start();  //AMRadio renames SplitControl, Start comes
from it.
                  //this is simplistic.  NOT low power since radio on
indefinitely.
    call a2dbankselect.selectIOFunc();
    call a2dbankselect.makeOutput();
    call a2dbankselect.clr();  //first state is LO to choose SENSIG1
muxed.
    call a2dmuxdisable.selectIOFunc();
    call a2dmuxdisable.makeOutput();
    call a2dmuxdisable.set();  //first state is HI to choose mux disabled.
    call a2dsenvdd1drv.selectIOFunc();
    call a2dsenvdd1drv.makeOutput();
    call a2dsenvdd1drv.clr();  //first state is  LO to choose senvdd1 off.
    call a2dsenvdd2drv.selectIOFunc();
    call a2dsenvdd2drv.makeOutput();
    call a2dsenvdd2drv.clr();  //first state is LO to choose senvdd2 off.
    call a2dterm2drv.selectIOFunc();
    call a2dterm2drv.makeOutput();
    call a2dterm2drv.set();  //first state is HI to choose term2 cutoff.

  }
  event void AMRadioOn.startDone(error_t starterr) {
    if (starterr == SUCCESS) {
       call timer0.startPeriodic(TIMER_PERIOD_MILLI);
    }
    else {
      call AMRadioOn.start();
    }
  }
  event void AMRadioOn.stopDone(error_t starterr) {
  }     // stopDone required by SplitControl do nothing.
```

```
  event void timer0.fired() {   //From Timer<TMilli>
//    timerend = call  timer0.getNow();  // time senvdd1 was turned on.
del a.
//    call timer3.startOneShotAt(timerend, ALWAYS_SHUTOFF_MILLI);    del
a.
    call timer1.startOneShot(ECH2O_WARMUP_MILLI);
    call timer3.startOneShot(ALWAYS_SHUTOFF_MILLI);
    atomic  { bank1 = TRUE;  // next data ready goes in bank1 data slots.
    }
    call a2dmuxdisable.clr();  //timer1 -->  LO to choose mux enabled.
    call a2dbankselect.clr();  //timer0 -->   LO to choose SENSIG1 muxed.
    call a2dsenvdd1drv.set();  //timer0 -->  set HI a2dsenvdd1drv.
    atomic {
      timestamp++;   //a simplistic timestamp value to put in data slot
for now.
      }
    //   to let the ECH2O sensors get setup time done.
  }

  event void timer1.fired() {   //timer1 dt = ECH2O_WARMUP_MILLI
//    timerend = call  timer1.getNow();  // time senvdd2 was turned on.
del a.
    call timer2.startOneShot(ECH2O_WARMUP_MILLI);
    call a2dsenvdd2drv.set();  //timer1 -->  set HI a2dsenvdd2drv.
    atomic  {
      if (!busy) {       //AMSend not in progress
        call Resource.request();  // Resource --> adc12multichannel
    // 08jan08JG  get the outgoing packet payload pointer, (and default
len).
    //09dec07jg    //cast result of .getPayload to pointer rmspkt.
  //  rmspkt = (MoistureSensorsMsg*) (call AMSend.getPayload(&pkt,
pktlen));
 rmspkt = (MoistureSensorsMsg*) (call AMSend.getPayload(&pkt));
    rmspkt->nodeid = TOS_NODEID;  //  outgoing packet struct empty of a2d
data.
      }
    }
  }

  event void timer2.fired() {   //timer2 dt = ECH2O_WARMUP_MILLI
    call a2dmuxdisable.clr();  //timer2 -->  LO to choose mux enabled.
    call a2dbankselect.set();  //timer1 -->   HI to choose SENSIG2 muxed.
    atomic  {
    bank1 = FALSE;  // next data ready goes in bank2 data slots.
      if (!busy) {       //AMSend not in progress
        call Resource.request();  // Resource --> adc12multichannel
    // use the same outgoing packet payload pointer, (and default len).
      }
    }
  }

  event void timer3.fired() {   //action when AMSend was busy
    call a2dmuxdisable.set();  //HI to choose mux disabled.
    call a2dsenvdd1drv.clr();  //LO to choose senvdd1 off.
    call a2dsenvdd2drv.clr();  //LO to choose senvdd2 off.
  }

  async event void a2d12ch.dataReady(uint16_t *readybuf, uint16_t
readybuflen)
  {
    atomic {
```

```
    bufferlen = readybuflen;  //probably can just not use *readybuf
    }
    //sort adc read values into nx_struct MoistureSensorsMsg.
    if (bank1 == TRUE) {       // async handling of banks.
      rmspkt->timestamp = timestamp;  // 2nd bank will have delta from
stamp.
      post MoistureSensorsMsgBank1();
    }
    else {
      post MoistureSensorsMsgBank2();
// do something to send message now
    post MoistureSensorsMsgSend();
    }
  }

  event void Resource.granted()  //resource meaning is "just the a2d
channel"
  {
    const msp430adc12_channel_config_t* a2d12chconfig = call
AdcConfigure.getConfiguration();  //to get setup from a2d12ch.
        // start the adc read, ( use a2d12chconfig from above).
    atomic {
    if ( call a2d12ch.configure(a2d12chconfig, memctl, numMemctl, buffer,
bufferlen, jiffies) == SUCCESS){
      call Leds.led0On();  //debug aid
      call a2d12ch.getData();  // returns buffer[], bufferlen integer
    }
   }
  }

  event void AMSend.sendDone(message_t* msg, error_t sendresult) {
    call Leds.led0Off();  //debug aid
    call Leds.led1Off();  //debug aid
    call Leds.led2Off();  //debug aid
    if (&pkt == msg) {  // req'd when others could be sending "which msg?"
        // if (sendresult ==  SUCCESS)  lighter test than (&pkt == msg)
      atomic {
        busy = FALSE;
      }
    }
  }

  task void MoistureSensorsMsgBank1()
  {
    //assign buffer[] elements to rmspkt  nx_struct elements.
    atomic
    {
      rmspkt->adc00 = buffer[0];
      rmspkt->adc01 = buffer[1];
      rmspkt->adc02 = buffer[2];
     // rmspkt->adc03 = buffer[3];
    //  rmspkt->adc04 = buffer[4];
   //   rmspkt->adc05 = buffer[5];
    }
    //rmspkt struct is now filled with new bank1 data.
      //  shut off bank1 related.
      call Resource.release();
      call a2dsenvdd1drv.clr();  //dataReady -->  a2dsenvdd1drv = LO.
  }

  task void MoistureSensorsMsgBank2()
```

```
  {
    atomic
    {
      rmspkt->adc10 = buffer[0];
      rmspkt->adc11 = buffer[1];
      rmspkt->adc12 = buffer[2];
    //  rmspkt->adc13 = buffer[3];
    //  rmspkt->adc14 = buffer[4];
   //   rmspkt->adc15 = buffer[5];
    }
      //  shut off bank2 related.
      call a2dmuxdisable.set();  //data stored -->  HI disables mux.
      call a2dsenvdd2drv.clr();  //data stored -->  a2dsenvdd2drv = LO.
      call Resource.release();
  }

  task void MoistureSensorsMsgSend()
  {

    call Leds.led1On();  //debug aid
    if (call AMSend.send(AM_BROADCAST_ADDR, &pkt,
sizeof(MoistureSensorsMsg)) == SUCCESS) {
      call Leds.led2On();  //debug aid
      busy = TRUE;  //the data is queued to go out.
    }
  }
}
```

ReadMoistureSensors.h

```
/* Copyright (c) 2007, Ecosensory Austin Texas All rights reserved.
 *  This code funded by TX State San Marcos University. BSD license full
text at:
 * http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-
contrib/ecosensory/license.txt
 * by John Griessen <john@ecosensory.com>
 * Rev 1.0 14 Dec 2007
 */
#ifndef READMOISTURESENSORS_H
#define READMOISTURESENSORS_H

enum {
  ALWAYS_SHUTOFF_MILLI = 30,    // a2d reading delta t after sensors
powered.
  ECH2O_WARMUP_MILLI = 10,    // a2d reading delta t after sensors
powered.
  TIMER_PERIOD_MILLI = 50,   // a2d reading period.
//  AM_MOISTURESENSORSMSG is an active message type.  Depends on Makefile
//  BUILD_EXTRA_DEPS=MoistureSensorsMsg.class
//  MoistureSensorsMsg.class:  MoistureSensorsMsg.java
//  and MoistureSensorsMsg.java:
//  see tos/types/AM.h for other basic definitions related.
  AM_MOISTURESENSORSMSG = 3,
  TOS_NODEID = 22
};
typedef nx_struct MoistureSensorsMsg {
```

```
  nx_uint16_t nodeid;
  nx_uint16_t adc00; // bank 0, channel 0
  nx_uint16_t adc01; // bank 0, channel 1
  nx_uint16_t adc02; // bank 0, channel 2
  //nx_uint16_t adc03; // bank 0, channel 3
  //nx_uint16_t adc04; // bank 0, channel 4
  //nx_uint16_t adc05; // bank 0, channel 5
  nx_uint16_t adc10; // bank 1, channel 0
  nx_uint16_t adc11; // bank 1, channel 1
  nx_uint16_t adc12; // bank 1, channel 2
  //nx_uint16_t adc13; // bank 1, channel 3
  //nx_uint16_t adc14; // bank 1, channel 4
  //nx_uint16_t adc15; // bank 1, channel 5
  nx_uint16_t timestamp;
} MoistureSensorsMsg;
#endif
```

MoistureSensorsMsg.java

```
/**
 * This class is automatically generated by mig. DO NOT EDIT THIS FILE.
 * This class implements a Java interface to the 'MoistureSensorsMsg'
 * message type.
 */

public class MoistureSensorsMsg extends net.tinyos.message.Message {

    /** The default size of this message type in bytes. */
    public static final int DEFAULT_MESSAGE_SIZE = 16;

    /** The Active Message type associated with this message. */
    public static final int AM_TYPE = 3;


    /** Create a new MoistureSensorsMsg of size 28. */
    public MoistureSensorsMsg() {
        super(DEFAULT_MESSAGE_SIZE);
        amTypeSet(AM_TYPE);
    }

    /** Create a new MoistureSensorsMsg of the given data_length. */
    public MoistureSensorsMsg(int data_length) {
        super(data_length);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg with the given data_length
     * and base offset.
     */
    public MoistureSensorsMsg(int data_length, int base_offset) {
        super(data_length, base_offset);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg using the given byte array
     * as backing store.
```

```
     */
    public MoistureSensorsMsg(byte[] data) {
        super(data);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg using the given byte array
     * as backing store, with the given base offset.
     */
    public MoistureSensorsMsg(byte[] data, int base_offset) {
        super(data, base_offset);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg using the given byte array
     * as backing store, with the given base offset and data length.
     */
    public MoistureSensorsMsg(byte[] data, int base_offset, int
data_length) {
        super(data, base_offset, data_length);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg embedded in the given message
     * at the given base offset.
     */
    public MoistureSensorsMsg(net.tinyos.message.Message msg, int
base_offset) {
        super(msg, base_offset, DEFAULT_MESSAGE_SIZE);
        amTypeSet(AM_TYPE);
    }

    /**
     * Create a new MoistureSensorsMsg embedded in the given message
     * at the given base offset and length.
     */
    public MoistureSensorsMsg(net.tinyos.message.Message msg, int
base_offset, int data_length) {
        super(msg, base_offset, data_length);
        amTypeSet(AM_TYPE);
    }

    /**
    /* Return a String representation of this message. Includes the
     * message type name and the non-indexed field values.
     */
    public String toString() {
      String s = "Message <MoistureSensorsMsg> \n";
      try {
        s += "  [nodeid=0x"+Long.toHexString(get_nodeid())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [adc00=0x"+Long.toHexString(get_adc00())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [adc01=0x"+Long.toHexString(get_adc01())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
```

```
        s += "  [adc02=0x"+Long.toHexString(get_adc02())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      //try {
        //s += "  [adc03=0x"+Long.toHexString(get_adc03())+"]\n";
      //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
      //try {
        //s += "  [adc04=0x"+Long.toHexString(get_adc04())+"]\n";
      //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
      //try {
        //s += "  [adc05=0x"+Long.toHexString(get_adc05())+"]\n";
     //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
      try {
        s += "  [adc10=0x"+Long.toHexString(get_adc10())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [adc11=0x"+Long.toHexString(get_adc11())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      try {
        s += "  [adc12=0x"+Long.toHexString(get_adc12())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      //try {
        //s += "  [adc13=0x"+Long.toHexString(get_adc13())+"]\n";
      //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
     // try {
        //s += "  [adc14=0x"+Long.toHexString(get_adc14())+"]\n";
      //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
      //try {
        //s += "  [adc15=0x"+Long.toHexString(get_adc15())+"]\n";
      //} catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */
}
      try {
        s += "  [timestamp=0x"+Long.toHexString(get_timestamp())+"]\n";
      } catch (ArrayIndexOutOfBoundsException aioobe) { /* Skip field */ }
      return s;
    }

    // Message-type-specific access methods appear below.

    ////////////////////////////////////////////////////////
    // Accessor methods for field: nodeid
    //   Field type: int, unsigned
    //   Offset (bits): 0
    //   Size (bits): 16
    ////////////////////////////////////////////////////////

    /**
     * Return whether the field 'nodeid' is signed (false).
     */
    public static boolean isSigned_nodeid() {
        return false;
    }

    /**
     * Return whether the field 'nodeid' is an array (false).
     */
    public static boolean isArray_nodeid() {
```

```
    return false;
}

/**
 * Return the offset (in bytes) of the field 'nodeid'
 */
public static int offset_nodeid() {
    return (0 / 8);
}

/**
 * Return the offset (in bits) of the field 'nodeid'
 */
public static int offsetBits_nodeid() {
    return 0;
}

/**
 * Return the value (as a int) of the field 'nodeid'
 */
public int get_nodeid() {
    return (int)getUIntBEElement(offsetBits_nodeid(), 16);
}

/**
 * Set the value of the field 'nodeid'
 */
public void set_nodeid(int value) {
    setUIntBEElement(offsetBits_nodeid(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'nodeid'
 */
public static int size_nodeid() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'nodeid'
 */
public static int sizeBits_nodeid() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: adc00
//   Field type: int, unsigned
//   Offset (bits): 16
//   Size (bits): 16
////////////////////////////////////////////////////////

/**
 * Return whether the field 'adc00' is signed (false).
 */
public static boolean isSigned_adc00() {
    return false;
}

/**
 * Return whether the field 'adc00' is an array (false).
```

```
 */
public static boolean isArray_adc00() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc00'
 */
public static int offset_adc00() {
    return (16 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc00'
 */
public static int offsetBits_adc00() {
    return 16;
}

/**
 * Return the value (as a int) of the field 'adc00'
 */
public int get_adc00() {
    return (int)getUIntBEElement(offsetBits_adc00(), 16);
}

/**
 * Set the value of the field 'adc00'
 */
public void set_adc00(int value) {
    setUIntBEElement(offsetBits_adc00(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc00'
 */
public static int size_adc00() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc00'
 */
public static int sizeBits_adc00() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: adc01
//   Field type: int, unsigned
//   Offset (bits): 32
//   Size (bits): 16
////////////////////////////////////////////////////////

/**
 * Return whether the field 'adc01' is signed (false).
 */
public static boolean isSigned_adc01() {
    return false;
}
```

```
/**
 * Return whether the field 'adc01' is an array (false).
 */
public static boolean isArray_adc01() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc01'
 */
public static int offset_adc01() {
    return (32 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc01'
 */
public static int offsetBits_adc01() {
    return 32;
}

/**
 * Return the value (as a int) of the field 'adc01'
 */
public int get_adc01() {
    return (int)getUIntBEElement(offsetBits_adc01(), 16);
}

/**
 * Set the value of the field 'adc01'
 */
public void set_adc01(int value) {
    setUIntBEElement(offsetBits_adc01(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc01'
 */
public static int size_adc01() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc01'
 */
public static int sizeBits_adc01() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: adc02
//   Field type: int, unsigned
//   Offset (bits): 48
//   Size (bits): 16
////////////////////////////////////////////////////////

/**
 * Return whether the field 'adc02' is signed (false).
 */
public static boolean isSigned_adc02() {
    return false;
```

```
}

/**
 * Return whether the field 'adc02' is an array (false).
 */
public static boolean isArray_adc02() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc02'
 */
public static int offset_adc02() {
    return (48 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc02'
 */
public static int offsetBits_adc02() {
    return 48;
}

/**
 * Return the value (as a int) of the field 'adc02'
 */
public int get_adc02() {
    return (int)getUIntBEElement(offsetBits_adc02(), 16);
}

/**
 * Set the value of the field 'adc02'
 */
public void set_adc02(int value) {
    setUIntBEElement(offsetBits_adc02(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc02'
 */
public static int size_adc02() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc02'
 */
public static int sizeBits_adc02() {
    return 16;
}



////////////////////////////////////////////////////////
// Accessor methods for field: adc10
//   Field type: int, unsigned
//   Offset (bits): 64
//   Size (bits): 16
////////////////////////////////////////////////////////

/**
```

```
 * Return whether the field 'adc10' is signed (false).
 */
public static boolean isSigned_adc10() {
    return false;
}

/**
 * Return whether the field 'adc10' is an array (false).
 */
public static boolean isArray_adc10() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc10'
 */
public static int offset_adc10() {
    return (64 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc10'
 */
public static int offsetBits_adc10() {
    return 64;
}

/**
 * Return the value (as a int) of the field 'adc10'
 */
public int get_adc10() {
    return (int)getUIntBEElement(offsetBits_adc10(), 16);
}

/**
 * Set the value of the field 'adc10'
 */
public void set_adc10(int value) {
    setUIntBEElement(offsetBits_adc10(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc10'
 */
public static int size_adc10() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc10'
 */
public static int sizeBits_adc10() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: adc11
//   Field type: int, unsigned
//   Offset (bits): 80
//   Size (bits): 16
////////////////////////////////////////////////////////
```

```
/**
 * Return whether the field 'adc11' is signed (false).
 */
public static boolean isSigned_adc11() {
    return false;
}

/**
 * Return whether the field 'adc11' is an array (false).
 */
public static boolean isArray_adc11() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc11'
 */
public static int offset_adc11() {
    return (80 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc11'
 */
public static int offsetBits_adc11() {
    return 80;
}

/**
 * Return the value (as a int) of the field 'adc11'
 */
public int get_adc11() {
    return (int)getUIntBEElement(offsetBits_adc11(), 16);
}

/**
 * Set the value of the field 'adc11'
 */
public void set_adc11(int value) {
    setUIntBEElement(offsetBits_adc11(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc11'
 */
public static int size_adc11() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc11'
 */
public static int sizeBits_adc11() {
    return 16;
}

////////////////////////////////////////////////////////
// Accessor methods for field: adc12
//   Field type: int, unsigned
//   Offset (bits): 96
```

```
//   Size (bits): 16
//////////////////////////////////////////////////////

/**
 * Return whether the field 'adc12' is signed (false).
 */
public static boolean isSigned_adc12() {
    return false;
}

/**
 * Return whether the field 'adc12' is an array (false).
 */
public static boolean isArray_adc12() {
    return false;
}

/**
 * Return the offset (in bytes) of the field 'adc12'
 */
public static int offset_adc12() {
    return (96 / 8);
}

/**
 * Return the offset (in bits) of the field 'adc12'
 */
public static int offsetBits_adc12() {
    return 96;
}

/**
 * Return the value (as a int) of the field 'adc12'
 */
public int get_adc12() {
    return (int)getUIntBEElement(offsetBits_adc12(), 16);
}

/**
 * Set the value of the field 'adc12'
 */
public void set_adc12(int value) {
    setUIntBEElement(offsetBits_adc12(), 16, value);
}

/**
 * Return the size, in bytes, of the field 'adc12'
 */
public static int size_adc12() {
    return (16 / 8);
}

/**
 * Return the size, in bits, of the field 'adc12'
 */
public static int sizeBits_adc12() {
    return 16;
}


//////////////////////////////////////////////////////
```

```
    // Accessor methods for field: timestamp
    //   Field type: int, unsigned
    //   Offset (bits): 112
    //   Size (bits): 16
    //////////////////////////////////////////////////////

    /**
     * Return whether the field 'timestamp' is signed (false).
     */
    public static boolean isSigned_timestamp() {
        return false;
    }

    /**
     * Return whether the field 'timestamp' is an array (false).
     */
    public static boolean isArray_timestamp() {
        return false;
    }

    /**
     * Return the offset (in bytes) of the field 'timestamp'
     */
    public static int offset_timestamp() {
        return (112 / 8);
    }

    /**
     * Return the offset (in bits) of the field 'timestamp'
     */
    public static int offsetBits_timestamp() {
        return 112;
    }

    /**
     * Return the value (as a int) of the field 'timestamp'
     */
    public int get_timestamp() {
        return (int)getUIntBEElement(offsetBits_timestamp(), 16);
    }

    /**
     * Set the value of the field 'timestamp'
     */
    public void set_timestamp(int value) {
        setUIntBEElement(offsetBits_timestamp(), 16, value);
    }

    /**
     * Return the size, in bytes, of the field 'timestamp'
     */
    public static int size_timestamp() {
        return (16 / 8);
    }

    /**
     * Return the size, in bits, of the field 'timestamp'
     */
    public static int sizeBits_timestamp() {
        return 16;
    }
}
```

Παράρτημα 2

Το πρόργαμμα ReadMIG3Channels.m

```
%% Read measurement data from MIG format files

clear all;
close all;

%3 Channels respiration measurement Test M file
node_id = [];
ADC00_ar = [];
ADC01_ar = [];
ADC02_ar = [];
ADC10_ar = [];
ADC11_ar = [];
ADC12_ar = [];
timestamp_ar = [];

%open file at the path
 (3channels)\3ch respiration measurements

fid = fopen('C:\Documents and Settings\Loizos\My Documents\my
docs\Diplomatiki\Metriseis_resp24_03\Apnea\apnea4_mig.txt');
line = fgetl(fid);

while (~feof(fid))

    equal_sym = find(line=='=');
    if (isempty(equal_sym))
        line = fgetl(fid);
    else
        switch (line(4:equal_sym-1))
            case 'adc00'
                ADC00_ar = [ADC00_ar (hex2dec(line(12:14))/4096)*3];
            case {'adc01'}
                ADC01_ar = [ADC01_ar (hex2dec(line(12:14))/4096)*3];
            case {'adc02'}
                ADC02_ar = [ADC02_ar (hex2dec(line(12:14))/4096)*3];
            case {'adc10'}
                ADC10_ar = [ADC10_ar (hex2dec(line(12:14))/4096)*3];
            case {'adc11'}
                ADC01_ar = [ADC01_ar (hex2dec(line(12:14))/4096)*3];
            case {'adc12'}
                ADC02_ar = [ADC02_ar (hex2dec(line(12:14))/4096)*3];
            case {'timestamp'}
                x_index = find(line== 'x');
                end_index = find(line == ']');
```

```
            timestamp_ar = [timestamp_ar hex2dec(line(x_index+1:end_index-1))];
        end
        line = fgetl(fid);

    end
end

%% Create timestamp
timestamp_ar = timestamp_ar - timestamp_ar(1);
time_stamp = [timestamp_ar (timestamp_ar(1:end)+timestamp_ar(end))];

%% Channels ADC01 and ADC11 sample the X axis and ADC02 and ADC12 the Y
axis

X = ADC01_ar ;
Y = ADC02_ar ;

% Plot the results
subplot(2,1,1);
plot(time_stamp,X);
title('X plot');
subplot(2,1,2);
plot(time_stamp,Y);
title('Y plot');
```

Το πρόγραμμα ReadListen3Channel.m

```
clear all;
close all;

%3 Channels respiration measurement Test M file
node_id = [];
ADC00_ar = [];
ADC01_ar = [];
ADC02_ar = [];
ADC10_ar = [];
ADC11_ar = [];
ADC12_ar = [];
timestamp_ar = [];

%open file at the path
 (3channels)\3ch respiration measurements
```

```
fid = fopen('C:\Documents and Settings\Loizos\My Documents\my
docs\Diplomatiki\Metriseis_resp24_03\Apnea\apnea1_listen.txt');

%% Access every line of the file until EOF

while (~feof(fid))
   line = fgetl(fid);

   nodeid = [line(25:26) line(28:29)];
   node_id = [node_id hex2dec(nodeid)];

   ADC00 = [line(31:32) line(34:35)];
   ADC00_ar = [ADC00_ar (hex2dec(ADC00)/4096)*3];

   ADC01 = [line(37:38) line(40:41)];
   ADC01_ar = [ADC01_ar (hex2dec(ADC01)/4096)*3];

   ADC02 = [line(43:44) line(46:47)];
   ADC02_ar = [ADC02_ar (hex2dec(ADC02)/4096)*3];

   ADC10 = [line(49:50) line(52:53)];
   ADC10_ar = [ADC10_ar (hex2dec(ADC10)/4096)*3];

   ADC11 = [line(55:56) line(58:59)];
   %ADC11_ar = [ADC11_ar (hex2dec(ADC11)/4096)*3];
   ADC01_ar = [ADC01_ar (hex2dec(ADC11)/4096)*3];

   ADC12 = [line(61:62) line(64:65)];
   %ADC12_ar = [ADC12_ar (hex2dec(ADC12)/4096)*3];
   ADC02_ar = [ADC02_ar (hex2dec(ADC12)/4096)*3];

   timestamp = [line(67:68) line(70:71)];
   timestamp_ar = [timestamp_ar hex2dec(timestamp)];

end

% create timestamp array with the first message stamped with 0 and the
% last with 366 since we have 2 samples per message for each channel

%% Create a unique timestamp array for this measurement file

%Substract the value of the first timestamp in order to start from 0
%offset = timestamp_ar(1);
timestamp_ar = timestamp_ar - timestamp_ar(1);
time_stamp = [timestamp_ar (timestamp_ar(1:end)+timestamp_ar(end))];

%% Channels ADC01 and ADC11 sample the X axis and ADC02 and ADC12 the Y
axis
```

```
X = ADC01_ar ;
Y = ADC02_ar ;

% Plot the results
subplot(2,1,1);
plot(time_stamp,X);
title('X plot');
subplot(2,1,2);
plot(time_stamp,Y);
title('Y plot');
```

Το πρόγραμμα Method1_respcounter.m

```
%function resp_counter(decvalue_x,decvalue_y)
%Simple RESP detection algorithm
close all;

%Initialization values
resp_count = 0;
index_max = 0;
peak_values = [];
peak_time_values = [];
period_peak = [];
filtered_x = moving_average(60,X);
decvalue_x = filtered_x(70:end);
filtered_y = moving_average(50,Y);
decvalue_y = filtered_y(80:end);

%Mean value standard deviation, criterion y_up
mean1 = mean(decvalue_x);
n = length(decvalue_x);
stdev = sqrt(sum((decvalue_x-mean1).^2/n));
y_up = mean1 + 0.5*stdev;

%Possible RESP complexes stored at the possibleQRS matrix.We assume we have
%a RESP complex if voltage value greater than mean value + 0.5*standard
%deviation (criterion)

possibleRESP = find(decvalue_x > y_up);

%Το possible_RESP περιεχει ολους τα indices των τιμών του πίνακα decvalue που
ικανοποιουν
%το κριτηριο.Για τις πραγματικές τιμές τάσης πρέπει να αναφέρομαι ως
```

```
%decvalue(possibleRESP(i)) ενώ για τις αντίστοιχες τιμές χρόνου ως
%xdata(possibleRESP(i))
%Ειναι ομως το RESP πολλα δειγματα που ειναι πανω απο το
%κριτηριο και ειναι συνεχομενης αριθμησης στο καθε RESP οποτε πρεπει να ψαξω
%για συνεχομενη αριθμηση που σημαινει ενας παλμός αρα ενα RESP complex.

index = 1;

while(index <= length(possibleRESP))

    %Find real RESP complexes and avoid counting spikes or peaks.If we have
    %a real RESP complex then possibleRESP(i) values will be continuous.
    %Οριοθετώ το RESP complex με δεικτη την πρωτη και την τελευταια τιμη του
    %δειγματος στον πινακα possibleRESP (temp,index-1).
    temp = index;
    i=index;
    while (i+1 <=length(possibleRESP) && abs(possibleRESP(i)-
possibleRESP(i+1))==1)
        %temp = index;
        i = i+1;
    end
    index = i+1;

    if (index - temp >20)
        resp_count = resp_count + 1;
        temp_max = 0;

        %για τον δεικτη του πινακα possibleRESP που παραπέμπει στο
        %αντίστοιχο δεδομένο του πινακα decvalue πρεπει να βρω σε ποιο
        %σημειο παρουσιαζει μεγιστο οποτε μεσω του xdata να πάρω την τιμή
        %του χρόνου

        for j=temp:index-1
            temp1 = decvalue_x(possibleRESP(j));
            if (temp1 > temp_max)
                %temp_max = temp1;
                index_max = j;
            end
        end

        peak_values = [peak_values max(decvalue_x(possibleRESP(temp:index-1)))];
        peak_time_values = [peak_time_values possibleRESP(index_max)];
    end
end
resp_count;
% peak_time_values
% peak_values

j =1;
while (j < length(peak_time_values))
```

```
    period_peak = [period_peak (peak_time_values(j+1)-peak_time_values(j))];
    j = j+1;
end

% mean_period = mean(period_peak)
 disp('Number of respirations on the x-axis = ');disp(resp_count);
%period_peak

% figure(1);
% plot(decvalue_x);
% hold on;
% plot(peak_time_values,peak_values,'r*');
% hold on;
% plot(1:length(decvalue_x),y_up)


%Same procedure fot the y-axis

 %Mean value standard deviation, criterion y_up
mean1 = mean(decvalue_y);
n = length(decvalue_y);
stdev = sqrt(sum((decvalue_y-mean1).^2/n));
y_up = mean1 + 0.5*stdev;


possibleRESP = find(decvalue_y > y_up);


index = 1;

while(index <= length(possibleRESP))


    temp = index;
    i=index;
    while (i+1 <=length(possibleRESP) && abs(possibleRESP(i)-
possibleRESP(i+1))==1)
        %temp = index;
        i = i+1;
    end
    index = i+1;


    if (index - temp >20)
        resp_count = resp_count + 1;
        temp_max = 0;



        for j=temp:index-1
```

```
            temp1 = decvalue_y(possibleRESP(j));
            if (temp1 > temp_max)
                %temp_max = temp1;
                index_max = j;
            end
        end

        peak_values = [peak_values max(decvalue_y(possibleRESP(temp:index-1)))];
        peak_time_values = [peak_time_values possibleRESP(index_max)];
    end
end
resp_count;
% peak_time_values
% peak_values

j =1;
while (j < length(peak_time_values))
    period_peak = [period_peak (peak_time_values(j+1)-peak_time_values(j))];
    j = j+1;
end

% mean_period = mean(period_peak)
disp('Number of respirations on the y-axis = ');;disp(resp_count);
disp('--------------------------------------------------------------------------------');
%period_peak

figure(1);
subplot(4,1,1);plot(X);title('X');
subplot(4,1,2);plot(decvalue_x);title('filtered_x');
hold on;
plot(peak_time_values,peak_values,'r*');
hold on;
plot(1:length(decvalue_x),y_up);
subplot(4,1,3);plot(Y);title('Y');
subplot(4,1,4);plot(decvalue_y);title('filtered_y');
hold on;
plot(peak_time_values,peak_values,'g*');
hold on;
plot(1:length(decvalue_y),y_up);
```

Το πρόγραμμα Method2_respcounter.m

```
%function RespCounter(X,Y)

%Simple resp detection algorithm which breaks the signal into windows and
%finds the maximun value in each window.then by using the 1 differenciation
%counts the picks of the signal hence the respirations .

index_array_x  = [];
index_array_y = [];

filtered_x = moving_average(60,X);
filtered_y = moving_average(80,Y);

close all

start = 1;
end_index =100;
max_value_x = [];
Index_max_x = [];
 for i = 1:fix(length(filtered_x)/100)
    [max1 Index] = max(filtered_x(start:end_index));
    max_value_x = [max_value_x max1];
    Index_max_x = [Index_max_x ((i-1)*100+Index)];
    start = end_index+1;
    end_index = end_index + 100;
 end

 start = 1;
end_index = 100;
max_value_y = [];
Index_max_y = [];
 for i = 1:fix(length(filtered_y)/100)
    [max1 Index] = max(filtered_y(start:end_index));
    max_value_y = [max_value_y max1];
    Index_max_y = [Index_max_y ((i-1)*100+Index)];
    start = end_index+1;
    end_index = end_index + 100;
 end




diff_fltx = diff(filtered_x);
diff_flty = diff(filtered_y);


counter=0;
j=1;
%for j=1:length(Index_max_x)
```

```
while(j<=length(Index_max_x))
   if ((j+1)>length(Index_max_x))
      break;
   else
      if (Index_max_x(j+1)-Index_max_x(j)) <=60;
         counter=counter+1;
         j=j+1;
         index_array_x = [index_array_x Index_max_x(j)];
      else
         if  diff_fltx(Index_max_x(j))==0;
            counter=counter+1;
            index_array_x = [index_array_x Index_max_x(j)];
   % disp(j);
   %disp(counter);
         else
            if diff_fltx(Index_max_x(j))<0 & diff_fltx(Index_max_x(j)-1)>0 %&
diff_fltx(Index_max_x(j)-2)>0 %& diff_fltx(Index_max_x(j)+4)<0;
               counter=counter+1;
               index_array_x = [index_array_x Index_max_x(j)];
   % disp(j);
            else
               if diff_fltx(Index_max_x(j)-1)>0 & diff_fltx(Index_max_x(j)+1)<=0 &
diff_fltx(Index_max_x(j)+2)<0;
                  counter=counter+1;
                  index_array_x = [index_array_x Index_max_x(j)];
  %  disp(j);
  %disp(counter);
               end
               end
            end
         end
      end
  j=j+1;
end
if (j == length(Index_max_x))
  if diff_fltx(Index_max_x(j))<0 & diff_fltx(Index_max_x(j)-1)>0
    counter=counter+1;
    index_array_x = [index_array_x Index_max_x(j)];
  else
    if diff_fltx(Index_max_x(j)-1)>0 & diff_fltx(Index_max_x(j)+1)<=0 &
diff_fltx(Index_max_x(j)+2)<0;
      counter=counter+1;
      index_array_x = [index_array_x Index_max_x(j)];
    end
  end
end
disp('Number of respirations at X axis =');disp(counter);

counter=0;
j=1;
```

```
%for j=1:length(Index_max_y)
while(j<=length(Index_max_y))
   if ((j+1)>length(Index_max_y))
      break;
   else
      if (Index_max_y(j+1)-Index_max_y(j)) <=60;
         counter=counter+1;
         j=j+1;
         index_array_y = [index_array_y Index_max_y(j)];
      else
         if diff_flty(Index_max_y(j))== 0;
            counter=counter+1;
            index_array_y = [index_array_y Index_max_y(j)];
    % disp(j);
   %disp(counter);
         else
            if diff_flty(Index_max_y(j))<0 & diff_flty(Index_max_y(j)-1)>0 %&
diff_flty(Index_max_y(j)-2)<0% & diff_flty(Index_max_y(j)-2)>0;
               counter=counter+1;
               index_array_y = [index_array_y Index_max_y(j)];
    %disp(j);
            else
               if diff_flty(Index_max_y(j)-1)>0 & diff_flty(Index_max_y(j)+1)<=0 &
diff_flty(Index_max_y(j)+2)<0;
                  counter=counter+1;
                  index_array_y = [index_array_y Index_max_y(j)];

  %  disp(j);
  %disp(counter);
               end
            end
         end
      end
   end
   j =j +1;
end
if (j == length(Index_max_y))
  if diff_flty(Index_max_y(j))<0 & diff_flty(Index_max_y(j)-1)>0
     counter=counter+1;
     index_array_y = [index_array_y Index_max_y(j)];
  else
     if diff_flty(Index_max_y(j)-1)>0 & diff_flty(Index_max_y(j)+1)<=0 &
diff_flty(Index_max_y(j)+2)<0;
        counter=counter+1;
        index_array_y = [index_array_y Index_max_y(j)];
     end
  end
end
disp('Number of respirations at Y axis =');disp(counter);
```

```
disp('-----------------------------------------------------------------------------------------------------------
-----');
subplot(4,1,1);plot(X);title('X');hold
on;plot(index_array_x,filtered_x(index_array_x),'r*');
subplot(4,1,2);plot(filtered_x);title('filtered_x');hold
on;plot(index_array_x,filtered_x(index_array_x),'r*');
subplot(4,1,3);plot(Y);title('Y');hold
on;plot(index_array_y,filtered_y(index_array_y),'r*');
subplot(4,1,4);plot(filtered_y);title('filtered_y');hold
on;plot(index_array_y,filtered_y(index_array_y),'r*');
```

Βιβλιογραφία

[1] ‘Body Sensor Networks’, Guang-Zhong Yang .Spinger 2006

[2] ‘Μελέτη ,Σχεδίαση και ανάπτυξη πρωτοκόλλου ελέγχου πρόσβασης στο φυσικό μέσο για ασύρματα δίκτυα ασύρματα αισθητήρων’, Διδακτορική διατριβή Λαμπρινός Ε Ηλίας ,Αθήνα Σεπτέμβριος 2006.

[3] ‘Μετάδοση και επεξεργασία ζωτικών σημείων μέσω Ασύρματων δικτύων Αισθητήρων’, Διπλωματική εργασία. Ιωάννης Γ.Νέλλας , Στάυρος Β.Πινατσης ,Αθήνα, Μαρτιος 2006

[4] ‘nesC 1.1 Language Reference Manual’ ,David Gay, Philip Levis, David Culler, Eric Brewer,May 2003

[5] ‘TinyOS Programming ,Philip Levis ,June 28, 2006

[6] ‘12 Lectures on Wireless Sensor Networks’,

[7] ‘System Architecture Directions for Networked Sensors _ ,Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Piste , Department of Electrical Engineering and Computer Sciences ,University of California, Berkeley,

[8] ‘Wireless sensor networks for personal health monitoring: Issues and an implementation’ Aleksandar Milenkovic´ *, Chris Otto, Emil Jovanov

[9] ‘SYSTEM ARCHITECTURE OF A WIRELESS BODY AREA SENSOR NETWORK FOR UBIQUITOUS HEALTH MONITORING’ CHRIS OTTO, ALEKSANDAR MILENKOVIĆ, COREY SANDERS, EMIL JOVANOV **University of Alabama in Huntsville**, Revised January 10, 2006

[10] ‘Sensor Networks Project Advanced Networking’ COSC 7377 Spring 2005 Ayaz Ali, Ruta Shah Department of Computer Science University of Houston

[11] ‘Sensor Networks for Medical Care’, Victor Shnayder, Borrong **,**Chen, Konrad Lorincz, **,**Thaddeus R. F. FulfordJones, and Matt Welsh
Division of Engineering and Applied Sciences Harvard University

[12] ‘A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation’, Emil Jovanov*1, Aleksandar Milenkovic1, Chris Otto1 and Piet C de Groen2, March 2005., Journal of NeuroEngineering and Rehabilitation

[13] Memsic ,’application note’, Accelerometer fundamentals

[14] Memsic RS232 evaluation Board

[15] Memsic, RS232 Evaluation Board Operation Manual

[16] Moteiv, Tmote Sky JTAG guide

[17] Tmote Sky Data Sheet, Tmote Sky Quick Start Guide, http://www.moteiv.com/community/Tmote_Sky_Downloads

[18] 'Signal Processing Methods for Non-Invasive Respiration Monitoring' Laura Mason

[19] TOSSIM: A Simulator for TinyOS Networks ,Philip Levis and Nelson Lee pal@cs.berkeley.edu ,September 17, 2003

[20] 'Instruments and Methods Statistical techniques to select detection thresholds for peak signals in ice-core data'2005 L. KARLOF, T.A. dIGARD, F. GODTLIEBSEN, M. KACZMARSKA, H. FISCHER

[21] Wireless Sensor Networks, The Morgan Kaufmann Series in Networking Series Editor, David Clark, M.I.T.

[22] Analysis of respiratory mechanomyographic signals by means of empirical mode decomposition.,A Torres,R Jane,E Lasiar,J A Fiz,J B Galdiz,J Gea,J Morera,2007

[23] www.biopac.com

[24] ZigBee Wikipedia, http://en.wikipedia.org/wiki/ZigBee

[25] www.zigbee.com

[26] TinyOs Community Forum, http://www.tinyos.net/scoop

[27] Moteiv, www.moteiv.com

[28] *Programming Tektronix OpenChoice Oscilloscopes with Matlab*', Tektronix OpenChoice Software Developers' Kit – Articles (PHP0249)

[29] TinyOs Community Forum, http://www.tinyos.net/scoop

[30] '*A quantitative comparison of different methods to detect cardiorespiratory coordination during night-time sleep*', Dirk Cysarz, Henrik Bettermann, Silke Lange, Daniel Geue and Peter van Leeuwen, BioMedical Engineering OnLine 2004, 3:44

[31] *'Moving Average Filters'*, The Scientist and Engineer's Guide to Digital Signal Processing, Chapter 15

[32] *'Signal Processing Toolbox, For use with Matlab', User's Guide, Version 7.14b*

[33] http://www.memsic.com/products/selector_app.htm

[34] http://www.tinyos.net/tinyos-2.x/doc/nesdoc/

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

[36] http://nesc.sourceforge.net

[37] http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/ecosensory/