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

Συγκριτική μελέτη αλγορίθμων ανίχνευσης συμπλέγματος QRS σε ηλεκτροκαρδιογραφήματα και υλοποίηση σε επίπεδο δικτύου ασύρματων αισθητήρων

Διπλωματική Εργασία

ΜΠΑΡΔΑΜΑΣΚΟΥ ΓΙΩΤΑ

ΧΕΙΛΑΔΑΚΗ ΣΤΕΛΛΑ

Επιβλέπων: Φίλιππος Κωνσταντίνου

Καθηγητής Ε.Μ.Π

ΑΘΗΝΑ, ΜΑΡΤΙΟΣ 2009

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

Συγκριτική μελέτη αλγορίθμων ανίχνευσης συμπλέγματος QRS σε ηλεκτροκαρδιογραφήματα και υλοποίηση σε επίπεδο δικτύου ασύρματων αισθητήρων

Διπλωματική Εργασία

ΜΠΑΡΔΑΜΑΣΚΟΥ ΓΙΩΤΑ
ΧΕΙΛΑΔΑΚΗ ΣΤΕΛΛΑ

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή
την ……Μαρτίου/2009

…….

Φ.Κωνσταντίνου	Κ. Νικήτα	Α. Παναγόπουλος
Καθηγητής ΕΜΠ	Καθηγήτρια ΕΜΠ	Καθηγητής ΕΜΠ

ΑΘΗΝΑ, ΜΑΡΤΙΟΣ 2009

……

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

……..

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

Περίληψη

Σκοπός αυτής της εργασίας είναι η μελέτη των αλγορίθμων ανίχνευσης συμπλέγματος QRS σε ηλεκτροκαρδιογραφήματα (ΗΚΓ) και υλοποίηση τους σε επίπεδο δικτύων ασύρματων αισθητήρων. Η τεχνολογία των δικτύων ασύρματων αισθητήρων 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 ΗΚΓ μέσω ενός ηλεκτροκαρδιογράφου, στη συνέχεια το σήμα αυτό υπόκειται σε επεξεργασία σε επίπεδο ασύρματων αισθητήρων (on board processing) και μεταδίδεται σε έναν ηλεκτρονικό υπολογιστή, όπου απεικονίζεται.

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.1-x, σχεδιασμένο ειδικά για ενσωματωμένα συστήματα. Όλη η οργανωτική δομή του TinyOS, οι βιβλιοθήκες και οι εφαρμογές του είναι γραμμένες στη γλώσσα προγραμματισμού NesC.

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. Πιο συγκεκριμένα, περιγράφονται οι κώδικες που υλοποιήθηκαν σε MATLAB για τη επεξεργασία του σήματος μέσω των αλγορίθμων ανίχνευσης QRS. Η σύγκριση και η αξιολόγηση σε αυτούς τους κώδικες οδήγησαν σε τρεις τεχνικές ανίχνευσης συμπλέγματος QRS με διαφορετικά χαρακτηριστικά η καθεμιά και με δυνατότητα υλοποίησής τους στο λειτουργικό σύστημα TinyOS.1-x. Στη συνέχεια, η συγκριτική μελέτη αυτών των τεχνικών οδήγησε στην υλοποίηση στο λειτουργικό σύστημα TinyOS.1-x της πιο αποδοτικής τεχνικής από αυτές τις τρεις. Η υλοποίηση της επέτρεψε την ασύρματη επεξεργασία του σήματος και τη μετάδοσή του, ώστε στη συνέχεια να απεικονιστεί στον ηλεκτρονικό υπολογιστή.

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.1-x, NesC, Zigbee, IEEE 802.15.4, ηλεκτροκαρδιογράφημα (ΗΚΓ), σύμπλεγμα QRS, MATLAB, A/D μετατροπέας, δειγματοληψία, επεξεργασία σήματος, on board processing, φίλτρο κινούμενου μέσου όρου.

Abstract

The scope of this thesis is the analysis on QRS complex detection algorithms for ECG signals and their implementation applied on wireless sensor networks. In the following years the technology of wireless sensor networks is expected to be used in medical and many other applications. In the application that we designed, a cardiac signal is being received from an ECG circuit, and then it is being processed over the wireless sensor network (on board processing) and finally it is transmitted to a PC, where it is pictured.

The application uses the wireless platform Tmote Sky (Moteiv), which is properly programmed and is described analytically. It is also being compared to a variety of other platforms that integrate wireless sensor networks. The operation of the Tmote Sky is based on the TinyOS operating system, which is specifically designed for embedded systems. The structure of TinyOS, its libraries and its applications are written in NesC programming language.

In this thesis, all parts pertaining to the application are described in detail, the way in which they are linked as well as the code in which they were written. We present the tools used for signal depiction and storage. Initially, the QRS complex detection algorithms were used for the implementation of the codes in MATLAB. These codes were needed for the signal processing. Based on the comparison and the evaluation of these codes, we managed to create three techniques for QRS complex detection. Each of them presented different characteristics and the possibility of its implementation applied on wireless sensor networks. Then, the comparative analysis on these techniques led to the implementation of the most efficient of them on the TinyOS operating system. In this way, the signal was on board processed and transmitted, so that it could be pictured on the PC.

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 applications, Tmote sky, Moteiv, TinyOS.1-x, NesC, Zigbee protocol, IEEE 802.15.4, electrocardiogram (ECG), QRS

detection, MATLAB, A/D converter (ADC), sampling, signal processing, on board processing, moving average filter.

Ευχαριστίες

Εκφράζουμε τις θερμές μας ευχαριστίες προς τον καθηγητή του Ε.Μ.Π κ. Φίλιππο Κωνσταντίνου, ο οποίος μας εμπιστεύτηκε την εκπόνηση της παρούσας διπλωματικής εργασίας και μας παρείχε όλη την απαραίτητη υποδομή για την επιτυχή διεκπεραίωση της. Ο υποδειγματικός τρόπος διδασκαλίας του, ο ζήλος και η αγάπη προς τους φοιτητές του καθώς και οι αμέτρητες γνώσεις που μας παρείχε κατά τα χρόνια 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 πολύτιμου χρόνου. Του ευχόμαστε τα καλύτερα για τη συνέχεια στη προσωπική και επαγγελματική του διαδρομή.

Θα θέλαμε να ευχαριστήσουμε 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 στις οικογένειές μας και στους φίλους και συμφοιτητές μας για την υποστήριξή τους.

Μπαρδαμάσκου Γιώτα

Χειλαδάκη Στέλλα

ΠΕΡΙΕΧΟΜΕΝΑ

Πίνακας Εικόνων

Πίνακας Πινάκων

Κεφάλαιο 1

ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ (WSN) ΚΑΙ ΕΦΑΡΜΟΓΕΣ

1.1 Εισαγωγή

Η ελαχιστοποίηση μεγέθους και κόστους που επέφερε η τεχνολογία ημιαγωγών την τελευταία δεκαετία, 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 ασύρματων 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 έχουν προταθεί και υλοποιηθεί, ενώ αναμένεται να προκαλέσουν σημαντικές αλλαγές στη καθημερινή μας ζωή.

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

Εικόνα 1Smart Dust. Πολλοί αισθητήρες 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 χρησιμοποιήθηκε σε πληθώρα εφαρμογών, όπως για παρακολούθηση των ατμοσφαιρικών και καιρικών συνθηκών. Αξιοσημείωτη εφαρμογή αποτελεί η βιοτεχνολογική προσέγγιση τις ιδέας με motes από χημικά συστατικά αντί για ηλεκτρονικά κυκλώματα.

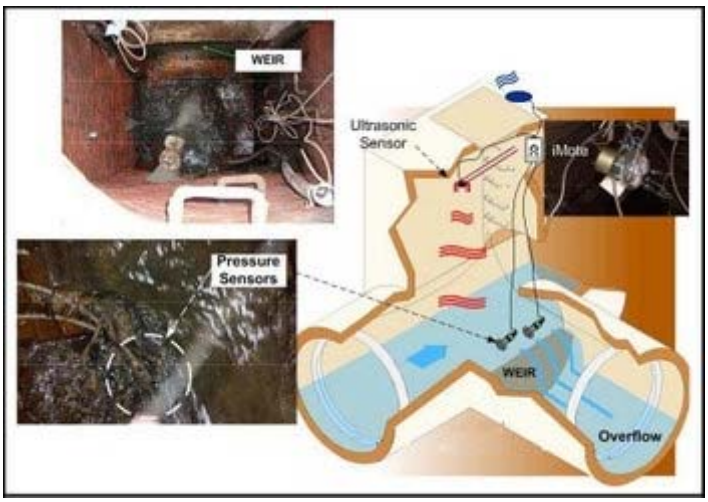
Ένα βασικό συστατικό των WSN είναι το μικρό, ανοικτού κώδικα, ενεργειακά αυτόνομο λειτουργικό σύστημα γνωστό ως Tiny Micro threading Operating System ή TinyOS που αναπτύχθηκε στα εργαστήρια του πανεπιστημίου του Berkeley. Το λειτουργικό αυτό προσφέρει το βασικό πλαίσιο και αναπτυξιακό περιβάλλον για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

Σήμερα, οι εφαρμογές των WSN χωρίζονται σε 3 κατηγορίες ως 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 2WSN για την έγκαιρη ανίχνευση προβλημάτων σε αγωγούς

Μία από τις πολυεθνικές εταιρίες που συνειδητοποίησε τις τεράστιες προοπτικές των WSN τεχνολογιών και έχει επενδύσει στην ανάπτυξη τους σε μεγάλη κλίμακα είναι η British Petroleum (BP). Σε μια πειραματική εφαρμογή μετρήθηκαν κατά τη διάρκεια των γεωτρήσεων τους οι μη φυσιολογικές δονήσεις που προειδοποιούν τους μηχανικούς για πιθανή επερχόμενη βλάβη του εξοπλισμού. Η 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 μετράται η πληρότητα και έπειτα εκπέμπεται μέσω δορυφόρου χαμηλής τροχιάς σε ένα σταθμό βάσης με αποτέλεσμα την άμεση ενημέρωση των πελατών για

τα αποθέματά τους, πριν αυτά εξαντληθούν. Η επιτυχία τέτοιου είδους κάλυψης με ενσύρματα μέσα δεν θα ήταν απλά μη συμφέρουσα οικονομικά και δύσκολη να υλοποιηθεί, αλλά ανέφικτη.

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

Εικόνα 3 Σύστημα εντοπισμού που αναπτύχθηκε στο Berkeley

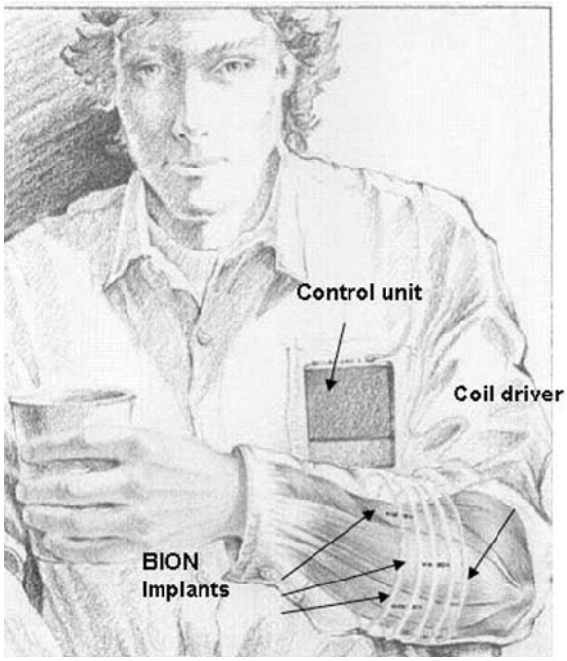
Το σύστημα εντοπισμού που χρησιμοποιήθηκε από το Berkeley στην εξέλιξη του smart dust.

Ενώ η τεχνολογία ασύρματων δικτύων εξελίσσεται και βρίσκει εφαρμογές σε όλο και ευρύτερο πεδίο, η πρόκληση βρίσκεται 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, γεγονός 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).

1.2 Ασύρματα δίκτυα αισθητήρων στην περιοχή του ανθρωπίνου σώματος (BSN)

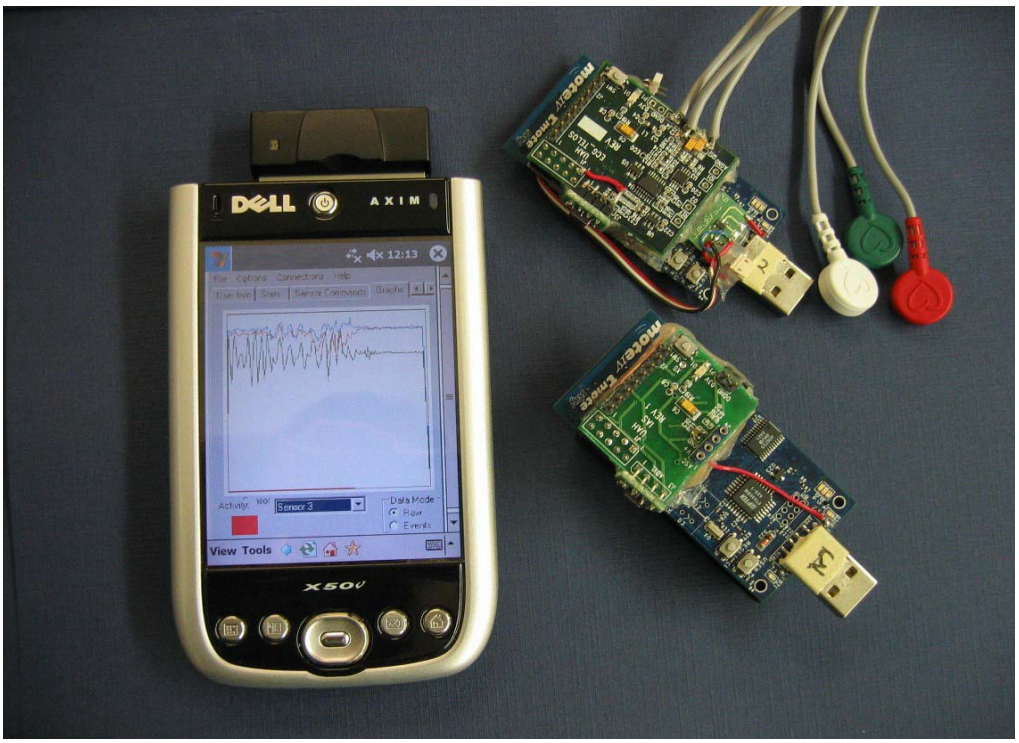
Καινοτομίες σε όλους τους τομείς της ζωής έχουν προσφερθεί στην κοινωνία από το σύγχρονο τεχνολογικό επίπεδο. Ιδιαίτερα, τέτοιες καινοτομίες στα πλαίσια της εφαρμογής και ενσωμάτωσης νέων τεχνολογιών παρατηρούνται και στον τομέα της υγείας και της ευρύτερης ιατρικής. Οι υπηρεσίες της τηλεϊατρικής προσανατολίζονται 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 διαρκής εξ αποστάσεως έλεγχος της κατάστασης της υγείας του ασθενούς μέσω της συλλογής, επεξεργασίας, αξιολόγησης, αξιοποίησης και αποθήκευσης κατάλληλης πληροφορίας.



Εικόνα 4 Σύστημα 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 την πλήρη εξάπλωση και εξάρτησή μας από τις ευρέως διαθέσιμες υποδομές ασυρμάτων και κινητών επικοινωνιών κατέστησαν 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 αποτελούν τη νέα τεχνολογία πληροφοριών που δύναται να υποστηρίξει την έγκαιρη ανίχνευση μη φυσιολογικών καταστάσεων και να προλάβει τις συνέπειες

τους. Πολλοί ασθενείς μπορούν να ωφεληθούν από τη συνεχή παρακολούθηση ως μέρος μιας διαγνωστικής διαδικασίας, τη βέλτιστη συντήρηση από μια χρόνια νόσο ή κατά τη διάρκεια εποπτευόμενης αποκατάστασης από μία 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) η έλλειψη ολοκληρωμένων συστημάτων των μεμονωμένων αισθητήρων,

γ) οι παρεμβολές στα κοινά ασύρματα κανάλια επικοινωνίας και

δ) η ανύπαρκτη υποστήριξη για την συλλογή μεγάλου όγκου δεδομένων.

Παραδοσιακά συστήματα ιατρικού ελέγχου, 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 διαθέτουν πολλά καλώδια μεταξύ των ηλεκτροδίων και του συστήματος ελέγχου, τα οποία περιορίζουν τη δραστηριότητα του ασθενή και το επίπεδο άνεσης και συνεπώς επηρεάζουν 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.

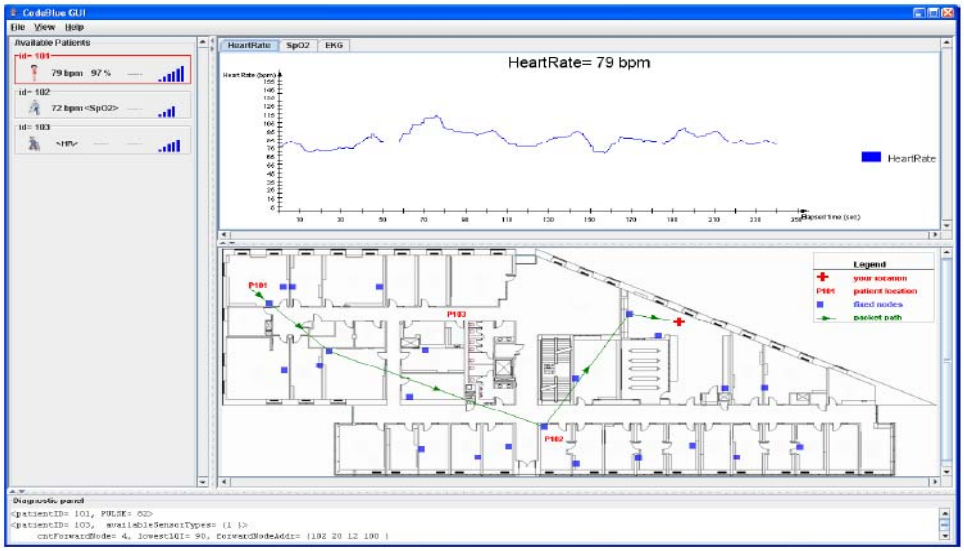
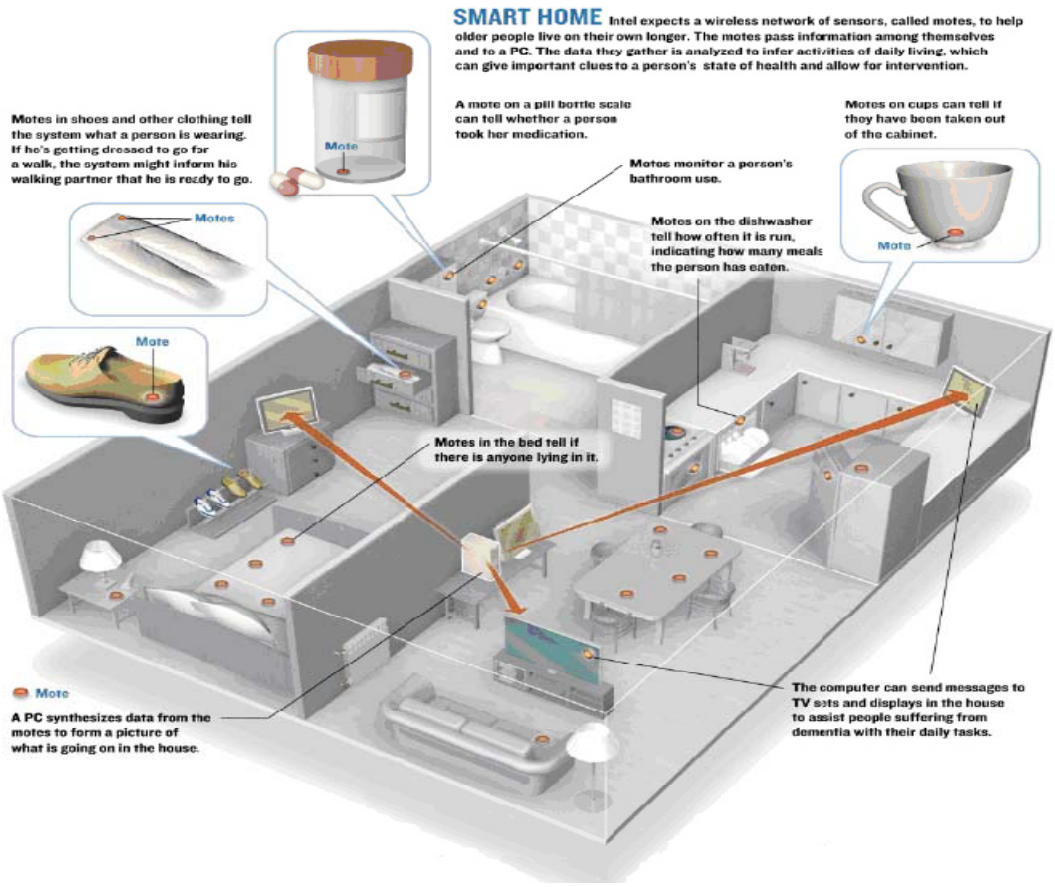


Figure 9: **The CodeBlue user interface.** *This is an actual screenshot of the CodeBlue GUI running in our building with three patient sensors reporting data to a laptop*

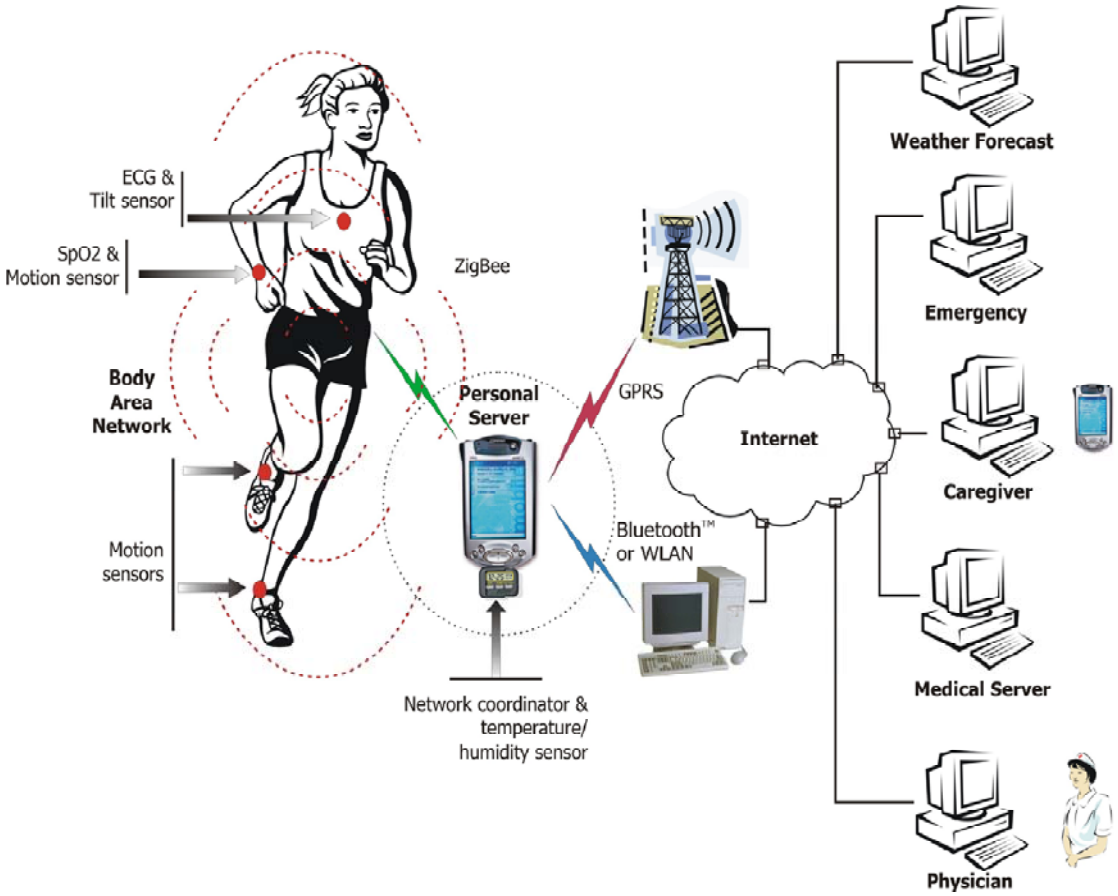
Εικόνα 6 Το interface της εφαρμογής Code Blue

Πρόσφατες πρόοδοι τεχνολογίας στην ασύρματη δικτύωση, τη μικροϋπολογιστική επεξεργασία, την ολοκλήρωση φυσικών αισθητήρων, ενσωματωμένων μικροελεγκτών και διεπαφών σε ένα ενιαίο κύκλωμα υπόσχονται μια νέα γενιά 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 επιρρεπείς 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 Οικιακό δίκτυο ασύρματων αισθητήρων

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 αποδειχθεί σημαντικά για την λειτουργία της μηχανικά υποβοηθούμενης αποκατάστασης των ασθενών.



Εικόνα 8 Δίκτυο αισθητήρων στο ανθρώπινο σώμα

Τα φορητά συστήματα τεχνολογίας και βιοανάδρασης εμφανίζονται να είναι έγκυρη λύση, δεδομένου ότι μειώνουν τον χρόνο προετοιμασίας του ασθενούς πριν από κάθε σύνοδο και απαιτούν λιγότερο χρόνο συμμετοχής των παθολόγων και των θεραπόντων. Η φορητή ασύρματη τεχνολογία επιτρέπει στους αισθητήρες να είναι τοποθετημένοι στον ασθενή για μεγάλες περιόδους, επομένως εξαλείφεται 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 πρόθυμος 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 των θεράποντων ιατρών θα μπορούσε να εξοικονομηθεί. 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 αποτελούν το εγκεφαλικό, η φυσική αποκατάσταση στο χώρο του ασθενούς μετά από χειρουργικές επεμβάσεις, η ανάρρωση από μυοκαρδιακό έμφραγμα, η 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 αντιμετωπιστεί αυτό το ζήτημα.

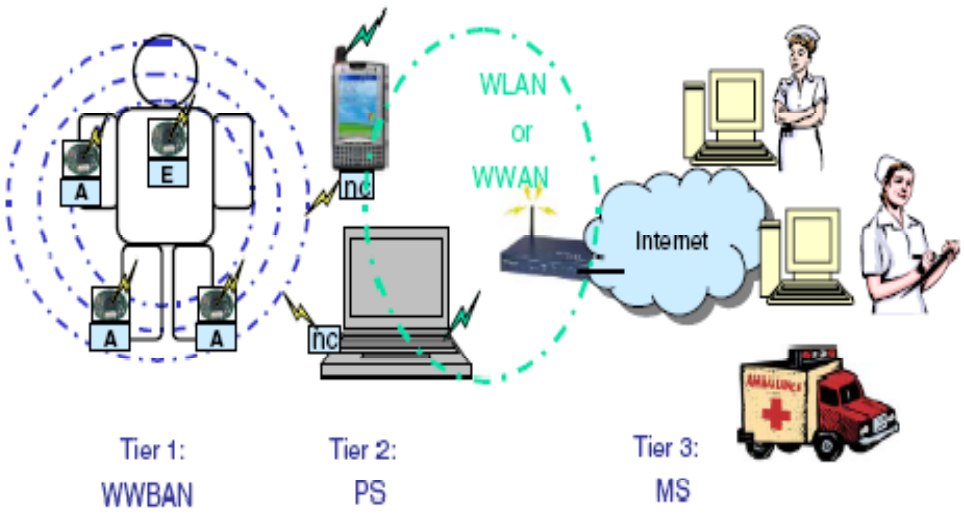


Fig. 1. WWBAN integrated into a telemedical system for health monitoring.

Εικόνα 9 Οι εφαρμογές του WSN στον τομέα της υγείας

Μια τέτοια εφαρμογή είναι το ActiS, βασισμένο σε μία τυποποιημένη ασύρματη πλατφόρμα αισθητήρων με μια εξειδικευμένη βάση για μονοκαναλικό βιοενισχυτή και δύο επιταχύνσιόμετρα. Ως αισθητήρας καρδιάς, το ActiS μπορεί να χρησιμοποιηθεί για να ελέγξει τη καρδιακή λειτουργία και τη θέση του ανώτερου κορμού. Ο ίδιος

αισθητήρας μπορεί να χρησιμοποιηθεί στην παρακολούθηση της θέσης και λειτουργίας των άνω και κάτω άκρων. Ένα φορητό σύστημα με το ActiS επιτρέπει την πρόσβαση στον μεταβολικό ρυθμό και δίνει τη συσσωρευμένη καταναλισκομένη ενέργεια σαν παράμετρο διαχείρισης πολλών ιατρικών καταστάσεων.

Μια αρχική έκδοση του ActiS έχει βασιστεί σε ευφυείς 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), η ZigBee λίστα πρωτοκόλλου καθώς επίσης και το νέο ZigBee συμβατό σε πλατφόρμα αισθητήρων Telos οδήγησαν 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 να επηρεάσει σημαντικά την επόμενη γενεά συστημάτων υγείας. Αυτή η τάση μπορεί, επίσης, να παρατηρηθεί στα πρόσφατα ανεπτυγμένα φυσιολογικά συστήματα οργάνων ελέγχου από το Harvard and Welch-Allen.

1.3 Αρχιτεκτονική συστημάτων

Συνεχείς τεχνολογικές πρόοδοι στα ολοκληρωμένα κυκλώματα, την ασύρματη επικοινωνία και τους αισθητήρες κατέστησαν εφικτή 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 με ένα ιατρικό κεντρικό υπολογιστή βάσεων δεδομένων. Ανάλογα με την περίπτωση, χρησιμοποιείται βασική γραμμή (ιατρική βάση δεδομένων), αισθητήρες (WBAN) και περιβαλλοντικές πληροφορίες (πρόβλεψη έκτακτης ανάγκης ή καιρού), με αλγορίθμους 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.

1.4 Επίπεδο αισθητήρων

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

- ένα αισθητήρα ΗΚΓ (ηλεκτροκαρδιογραφημάτων) για τον έλεγχο της καρδιακής δραστηριότητας,
- ένα αισθητήρα ΗΜΓ (ηλεκτρομυογραφία) για τον έλεγχο της δραστηριότητας των μυών,
- ένα αισθητήρα ΗΕΓ (ηλεκτροεγκεφαλογραφήματος) για τον έλεγχο ηλεκτρικής δραστηριότητας εγκεφάλου,
- έναν αισθητήρα πίεσης αίματος,
- ένα αισθητήρα κλίσης, για τον έλεγχο της θέσης κορμών,
- ένα αισθητήρα αναπνοής, για τον έλεγχο της αναπνοής,
- αισθητήρες μετακίνησης που χρησιμοποιούνται για να υπολογίσουν τη δραστηριότητα του χρήστη,
- έναν "αισθητήρα έξυπνων καλτσών" ή μια εξοπλισμένη με αισθητήρες σόλα παπουτσιών, για να σκιαγραφήσει τις φάσεις μεμονωμένων βημάτων.

Αυτοί οι αισθητήρες παράγουν τα χαρακτηριστικά αναλογικά σήματα, τα οποία 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 ικανοποιούν τις ακόλουθες απαιτήσεις: ελάχιστο βάρος, μικροσκοπική μορφή, χαμηλή ισχύς λειτουργίας ώστε να επιτραπεί παρατεταμένη λειτουργία, συνεχής ένταξη σε ένα WBAN, πρωτόκολλα διεπαφών, συγκεκριμένη βαθμονόμηση στο επίπεδο του ασθενούς, συντονισμός και προσαρμογή. Αυτές οι απαιτήσεις αποτελούν πρόκληση, αλλά είναι κρίσιμες εάν θέλουμε να προχωρήσουμε πέρα από απαρχαιωμένα συστήματα στην υγειονομική περίθαλψη, όπου ένας προμηθευτής δημιουργεί όλα τα προϊόντα. Μόνο υβριδικά με αυτό συστήματα, εφαρμοσμένα σε τμήματα υλικού και λογισμικού, που κατασκευάζονται από διαφορετικό προμηθευτή 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 διαβιβάζουν τη θέση και τα δεδομένα τους περιοδικά, επομένως μειώνουν σημαντικά την κατανάλωση ισχύος και 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 διαθέσιμες.

1.5 Εφαρμογές WSN

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

Στην κατηγορία των **περιβαλλοντικών εφαρμογών** έχουμε:

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

Εικόνα 10 Ασύρματος αισθητήρας τύπου crossbow για την εφαρμογή fire mentor

- 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 φαινομένων.

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.

Στην κατηγορία των **εφαρμογών ασφαλείας** έχουμε:

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

Στην κατηγορία των **στρατιωτικών εφαρμογών** έχουμε:

- Έλεγχος των κινήσεων του αντιπάλου.

Εικόνα 11 Το στρατιωτικό σύστημα υποβοήθησης στρατιωτών SaS

- Διαφύλαξη της ασφάλειας μίας περιοχής.

Εικόνα 12 Στιγμιότυπο από το σύστημα υποβοήθησης στρατιωτών SaS

- Απομακρυσμένο έλεγχο υλικού.

Στην κατηγορία των **εφαρμογών αντιμετώπισης φυσικών καταστροφών** έχουμε:

- Πρόληψη και διάγνωση σεισμικής δραστηριότητας.
- Εντοπισμός παγιδευμένων ατόμων.
- Ενσωμάτωση σε κατάλληλα σημεία, κατά την κατασκευή υποδομών, αισθητήρων που καταγράφουν την καταπόνηση του υλικού και τη μετακίνηση του.

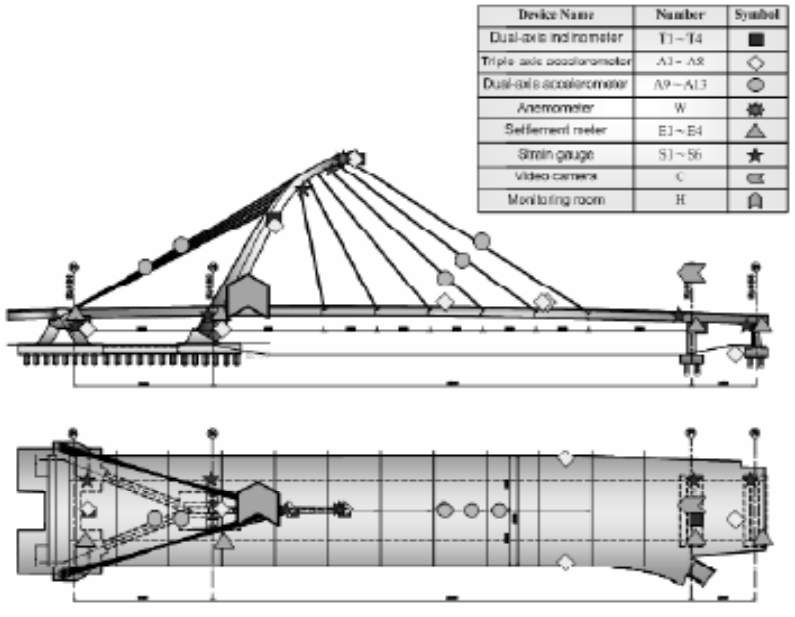


Figure 3. The layout diagram of sensors

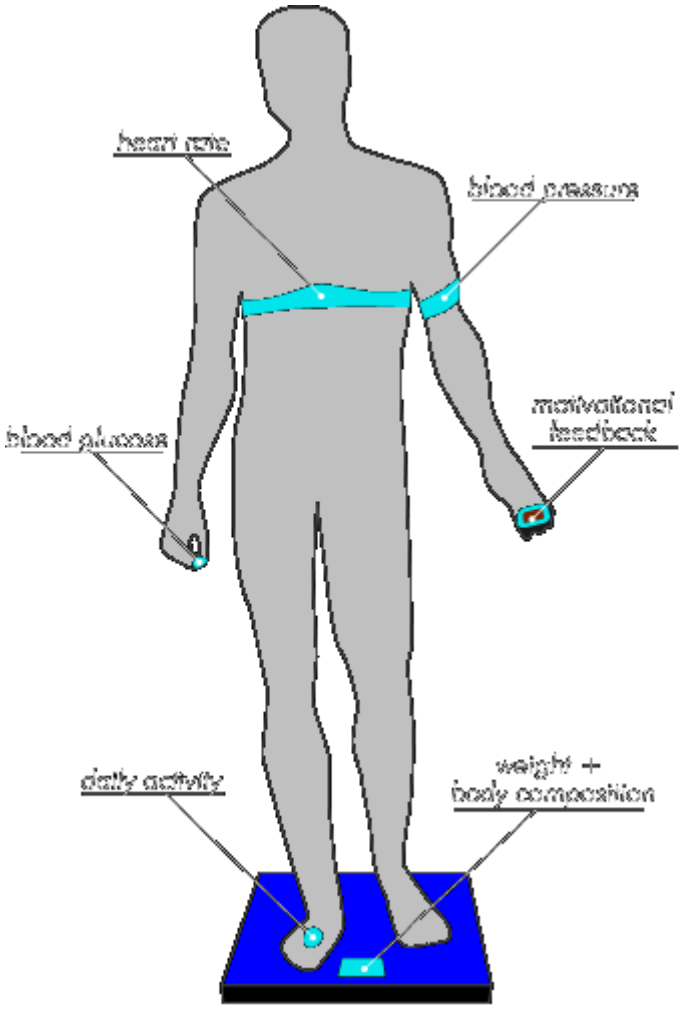
Εικόνα 13 WSN για την παρακολούθηση καταστροφικών ζημιών σε γέφυρες

Στην κατηγορία **των εφαρμογών στον χώρο της υγείας** έχουμε:

- 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, η πίεση κ.ά. 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.

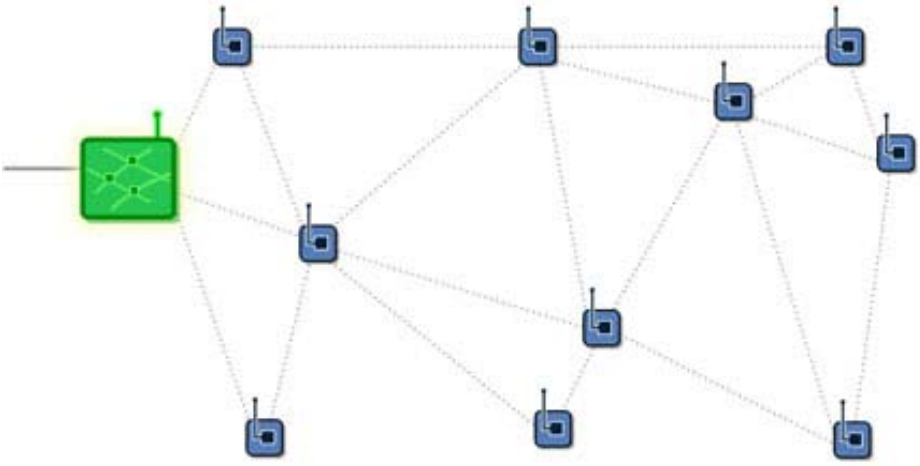
- Αισθητήρες για επιδημιολογικές μελέτες.
- Χορήγηση φαρμάκων.



Εικόνα 14 Εφαρμογές WSN στο χώρο της υγείας

1.6 Αρχές σχεδιασμού ασύρματων δικτύων αισθητήρων

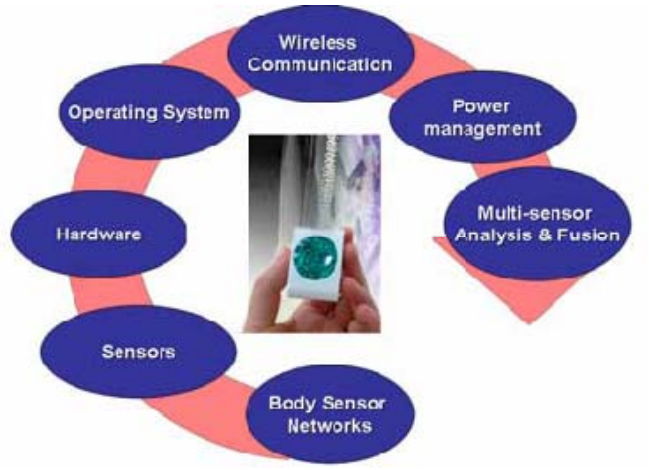
Όπως και με κάθε εφαρμογή, έτσι και στα WSN οι ιδιαιτερότητες της κάθε εφαρμογής καθορίζουν τις παραμέτρους σχεδιασμού του δικτύου. Οι παράμετροι αυτές είναι: η τοπολογία του δικτύου, η αξιοπιστία του, η κάλυψη και επεκτασιμότητα ποιότητας υπηρεσίας, ο συγχρονισμός και χρόνος απόκρισης, η ενεργειακή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



Εικόνα 15 WSN με multi-hop λειτουργίες όπου τα δεδομένα καταλήγουν σε ένα τελικό αποδέκτη

1.7 Προκλήσεις μελλοντικής εξέλιξης

Η εξελισσόμενη νανοτεχνολογία υπόσχεται 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, εφόσον 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 χαρακτηρίζονται προσωπικά και ανήκουν μόνο στο απόρρητο γιατρού ασθενούς. Άλλη μια παράμερος, που αποτελεί πρόκληση προς εξέλιξη, είναι η γνώση του εκάστοτε περιβάλλοντος και της κατάστασης του ασθενούς.



Εικόνα 16 Ένας κόμβος WSN

Κεφάλαιο 2

ΤΟ ΥΛΙΚΟ (HARDWARE), ΟΙ ΠΛΑΤΦΟΡΜΕΣ ΚΑΙ ΤΟ ΛΟΓΙΣΜΙΚΟ (SOFTWARE) ΤΩΝ WSN

2.1 Το υλικό (Hardware) των WSN

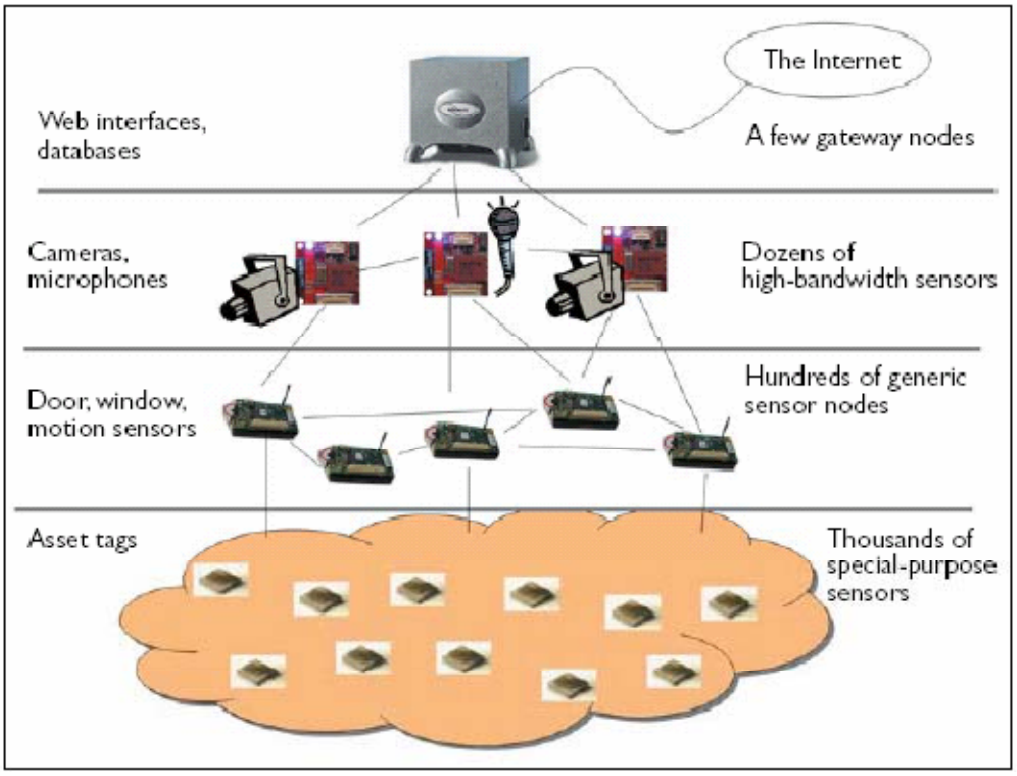
Όπως αναφέραμε στο προηγούμενο κεφάλαιο, τα ασύρματα δίκτυα αισθητήρων συνδυάζουν δυνατότητες επεξεργασίας, αίσθησης (sensing) και επικοινωνίας, σε μικροσκοπικά ενσωματωμένες συσκευές. Στη συνέχεια τα πρωτόκολλα επικοινωνίας συνδυάζουν κατάλληλα τις ανεξάρτητες συσκευές για τη δημιουργία ενός διασυνδεδεμένου 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 χρησιμοποιήθηκε 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.

2.1.1 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 17 Ιεραρχική ανάπτυξη ενός WSN

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, τοποθετημένους 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.

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

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

Στον πίνακα 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.

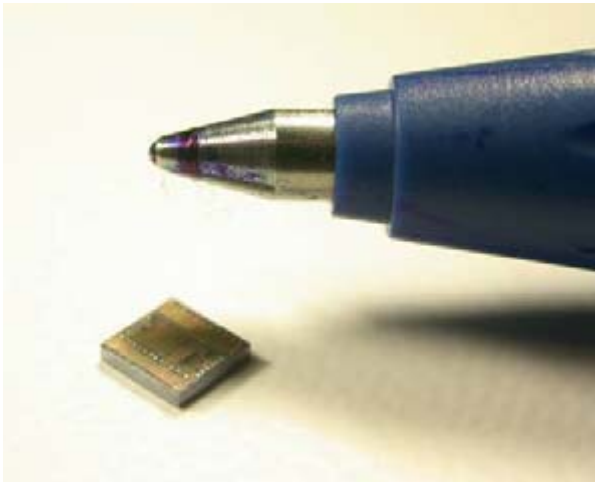
Node Type	Sample "Name" and Size	Typical Application Sensors	Radio Bandwidth (Kbps)	MIPS Flash RAM	Typical Active Energy (mW)	Typical Sleep Energy (uW)	Typical Duty Cycle (%)
Specialized sensing platform	Spec mm^3	Specialized low-bandwidth sensor or advanced RF tag	<50Kbps	<5 <0.1Mb <4Kb	1.8V*10–15mA	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 18 Τα τυπικά χαρακτηριστικά λειτουργίας των 4 κατηγοριών WSN

2.1.1.1 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.

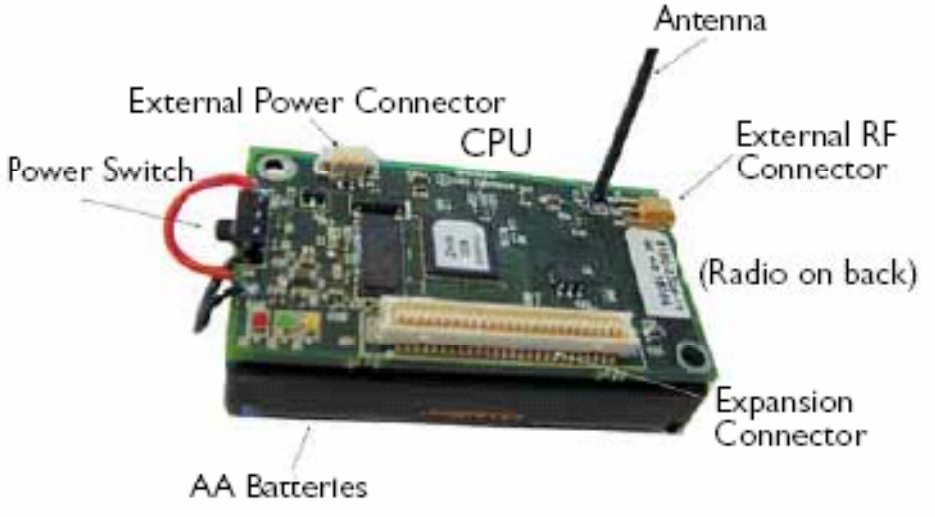


Εικόνα 19 Η μονάδα Spec

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.

2.1.1.2 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.



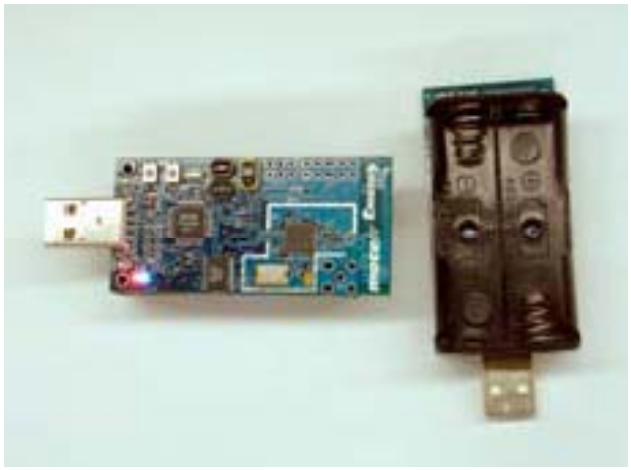
Εικόνα 20 Το Mica2

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.

2.1.1.3 Tmote sky

Το **tmote-sky** (το προηγούμενο μοντέλο ονομαζόταν Telosb) αποτελεί επίσης μια μονάδα που συνδυάζει ενσωματωμένους αισθητήρες, δυνατότητες ασύρματης

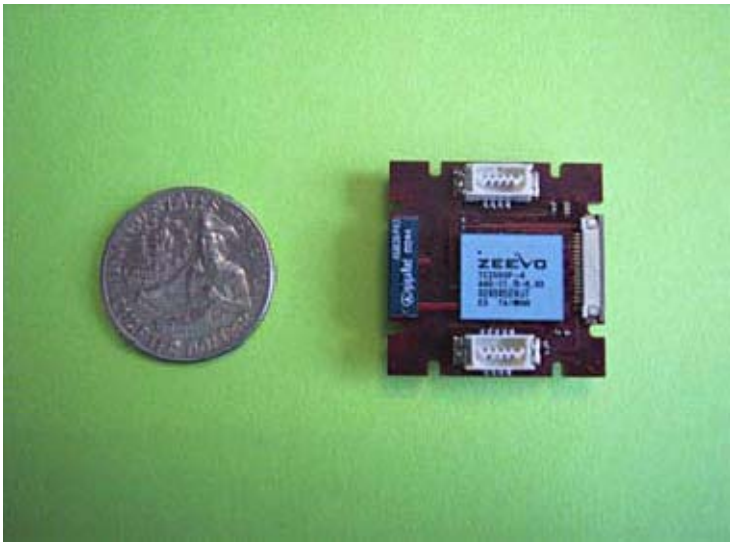
επικοινωνίας και προγραμματιστικές δυνατότητες. Τα χαρακτηριστικά του θα περιγραφούν αναλυτικά στη συνέχεια.



Εικόνα 21 Tmote-Sky

2.1.1.4 Το iMote

Το **iMote**, που δημιούργησε η Intel Research τον Μάιο του 2003, έχει σχεδιασθεί 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 22 Η πλατφόρμα της intel iMote

2.1.1.5 Η πλατφόρμα Stargate

Η πλατφόρμα **Stargate**, που ανέπτυξε η Intel και πούλησε η Crossbow Technology, είναι αντιπροσωπευτική των συσκευών κατηγορίας πύλης (gateway class 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 23 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.

2.1.1.6 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 Ζυρίχης από 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 Berkeley 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 24 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 τη χαμηλότερη δυνατή κατανάλωση ενέργειας από τους κόμβους και στοχεύει στην προσαρμοστικότητα, την εύκαμπτη και γρήγορη εφαρμογή. Για τον λόγο αυτό εκτός από την συμβατότητα της με το **TinyOS** χρησιμοποιεί 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.

2.1.2 Αρχιτεκτονικές διαφορές

Η συνολική αρχιτεκτονική δομή και στις τέσσερις 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 αφορούν το μέγεθος της on-board μνήμης, η χρήση ή μη μνήμης αναλαμπής (flash memory), το μέγεθος της ισχύος της κεντρικής μονάδας επεξεργασίας (CPU) καθώς επίσης και 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, συμπεριλαμβανομένων 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. Ομοίως, 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 τριών πρόσθετων αρχιτεκτονικών στοιχείων:

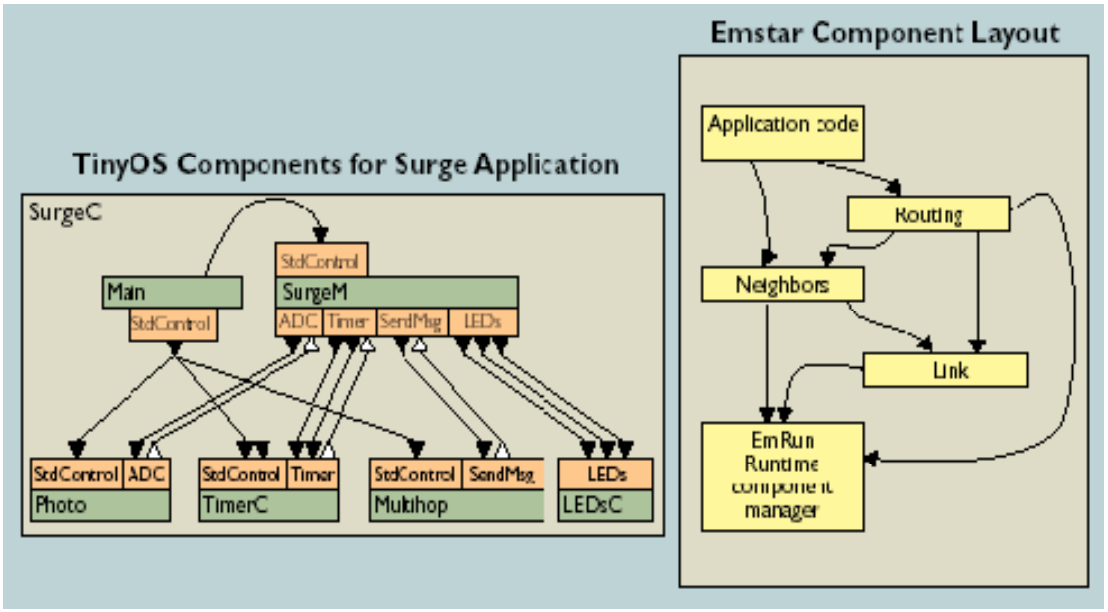
•Ένα πλαίσιο στοιχείων γενικού σκοπού που καταργεί τη διαστρωμάτωση

• 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 25 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.

2.2 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 οκταπλάσια 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 σχεδιασθεί 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.

Node	CPU	Power	Memory	I/O and Sensors	Radio	Remarks
Special-purpose Sensor Nodes						
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 Nodes						
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 Nodes						
BT Node 2001	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 1.0 2003	ARM 7TDMI 12-48MHz	1mW idle 120mW active	64KB SRAM 512KB Flash	UART, USB, GPIO, I^2C, SPI	Bluetooth 1.1	Multihop using 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.

Πίνακας 1 Σημερινές πλατφόρμες δικτύων αισθητήρων οργανωμένες κατά τάξη συσκευής

2.3 Πρότυπα λογισμικού και διεπαφών

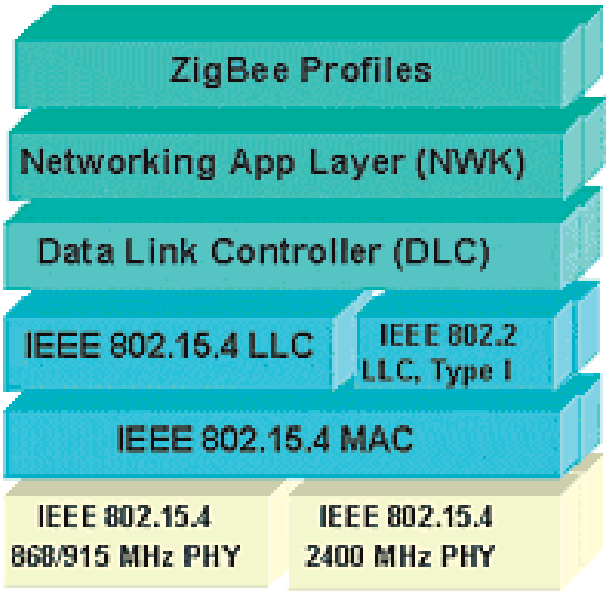
Μηχανικοί και ερευνητές που δραστηριοποιούνται στο χώρο της ασύρματης τεχνολογίας χαμηλής ισχύος χρησιμοποιούν όλο και περισσότερο το πρότυπο 802.15.4. Το πρότυπο αυτό παρέχει μια προδιαγραφή του καναλιού RF και του πρωτοκόλλου σηματοδοσίας. Το πρωτόκολλο **Zigbee**, που βασίζεται πάνω στο 802.15.4, 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.

2.3.1 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 26 Η δομή του πρωτοκόλλου ZigBee

Το πρωτόκολλο **Zigbee** αποσκοπεί σε εφαρμογές κτιριακού ελέγχου, τον 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, τη 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.

2.3.1.1.Αξιοπιστία και ασφάλεια δεδομένων

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

BAND	COVERAGE	DATA RATE	CHANNEL NUMBERS
2.4 GHz	ISM Worldwide	250 kbps	11-26
868 MHz	Europe	20 kbps	0
915 MHz	ISM Americas	40 kbps	1-10

Εικόνα 27 Το πρότυπο IEEE 802.15.4

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.

Όσον αφορά την ασφάλεια, 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. 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.

2.3.1.2. 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 μόνο 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 φωτοβολταϊκό 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 φωτοβολταϊκών μονάδων αυξάνει.

Το μείζον ζήτημα είναι πόση ενέργεια καταναλώνουν 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.

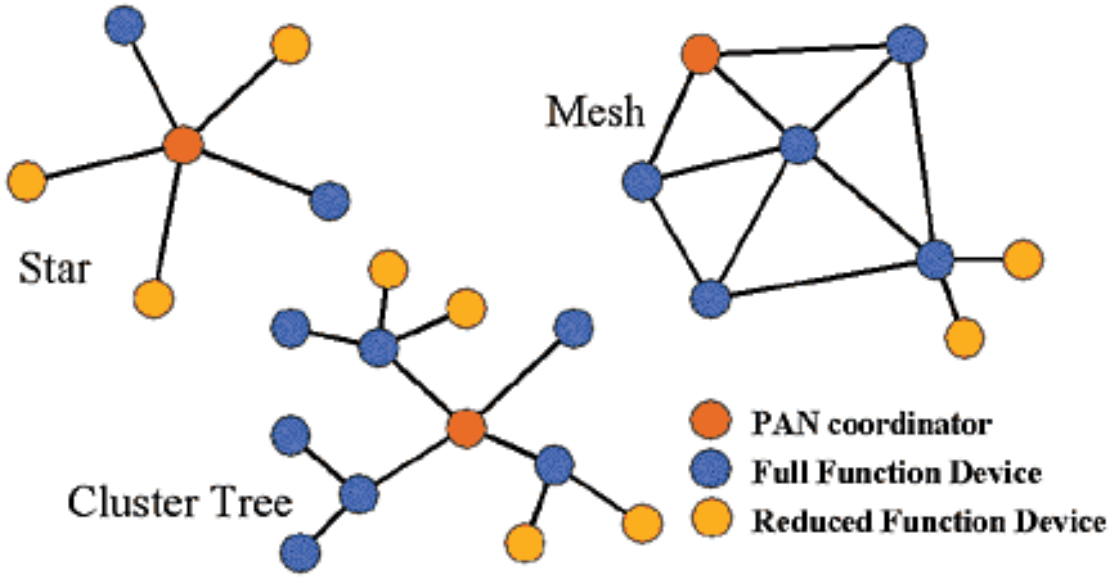
Εντούτοις, η λήψη γίνεται συνέχεια. Επιπλέον, δεδομένου ότι οι πομποί έχουν γίνει πιο σύνθετοι, το κόστος λήψης για ένα μήνυμα έχει αυξηθεί δραστικά. Το radio που χρησιμοποιήθηκε στα πρώτα motes UCB κατανάλωνε 9 mW ισχύ περιμένοντας τα μηνύματα. Στα νεότερα motes καταναλώνει 38 mW ισχύος κατά τη λήψη. 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/ανά 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 είναι δυνατό, χρησιμοποιούμε την επεξεργασία για να ελαττώσουμε το χρόνο εκπομπής εξοικονομώντας έτσι τεράστια ενέργεια. Αυτό επιβάλλει 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, εφόσον ένας 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 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 28 Μερικές μορφές τοπολογίας δικτύων

2.3.1.3 Κόστος

Το ZigBee και το 802.15.4 μεγιστοποιούν τη χρησιμότητά τους στον πολυδιάστατο τομέα του κόστους. Υπάρχει επαρκής ευελιξία και στα δύο πρότυπα ώστε να παρέχουν στο σχεδιαστή του συστήματος αισθητήρων μια ποικιλία τρόπων βελτιστοποίησης του κόστους, χωρίς να παραμελείται 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.

2.3.1.4 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

2.3.1.5 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 μέχρι χιλιάδες μέτρα. Τα δίκτυα μπορούν να έχουν δομή τύπου αστέρα, συστοιχίας δέντρων ή βροχωτού δικτύου, με την καθεμία να παρουσιάζει τις δικές της δυνατότητες.

2.3.1.6 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.

2.3.1.7 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. 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™ 802.15.4	Wi-Fi™ 802.11b	Bluetooth™ 802.15.1
Transmission Range (m)	1 – 100*	1 - 100	1 – 10
Battery Life (days)	100 – 1,000	0.5 – 5.0	1 - 7
Network Size (# of nodes)	> 64,000	32	7
Application	Monitoring & Control	Web, Email, Video	Cable Replacement
Stack Size (KB)	4 – 32	1,000	250
Throughput (kb/s)	20 – 250	11,000	720

Πίνακας 2 Βασικά χαρακτηριστικά του ZigBee και άλλων ασύρματων τεχνολογιών

2.4 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 29 Η ασύρματη μονάδα Tmote-Sky

2.4.1 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**.

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

Πίνακας 3 Κυριότερα τεχνικά χαρακτηριστικά του TinyOS

2.4.2 Τεχνικά χαρακτηριστικά μονάδας Tmote Sky

Η υλοποίηση και ανάπτυξη του tmote στηρίχθηκε σε τρεις βασικούς στόχους: την όσο το δυνατόν 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.

2.4.2.1 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.

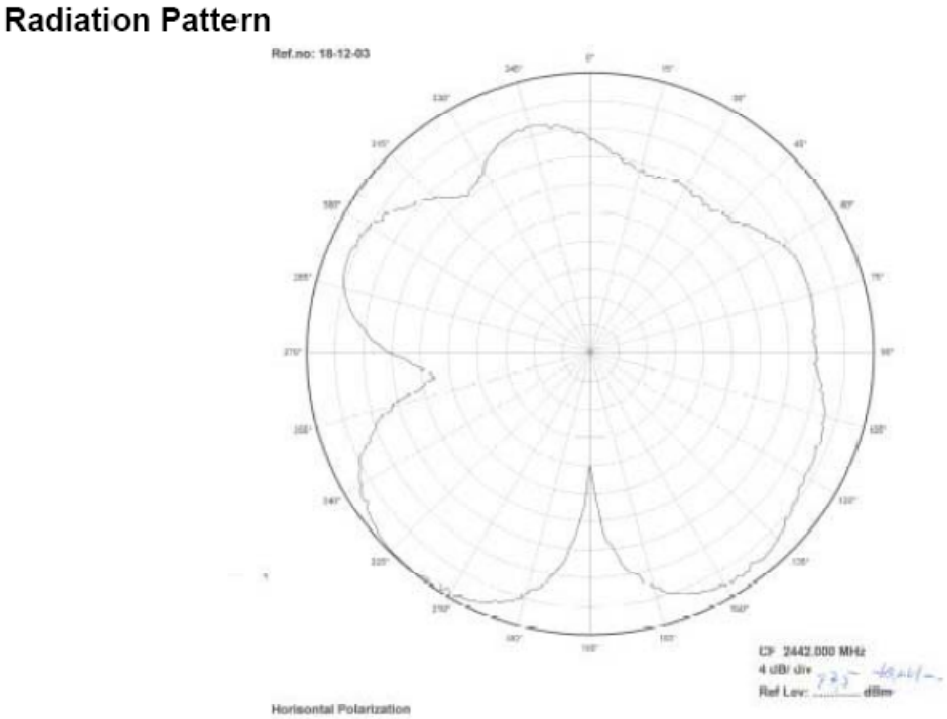


Figure 12 : Radiated pattern of the Inverted-F antenna with horizontal mounting (from Chipcon AS)

Εικόνα 30 Πολικό διάγραμμα της κεραίας σε οριζόντια εκπομπή

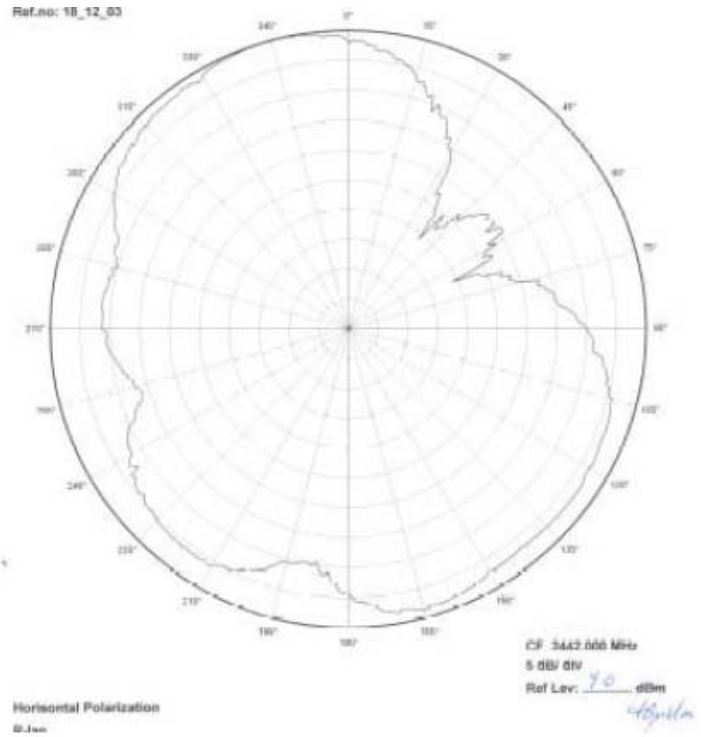


Figure 13 : Radiated pattern of the Inverted-F antenna with vertical mounting (from Chipcon AS)

Εικόνα 31 Πολικό διάγραμμα της κεραίας σε κατακόρυφη εκπομπή

Παρουσιάζονται 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 no.	**RFM TR1000**	**Chipcon CC1000**	**Chipcon CC2400**	**Nordic nRF2401**	**Chipcon CC2420**	**Motorola MC13191/92**	**Zeevo ZV4002**
Max Data rate (kbps)	115.2	76.8	1000	1000	250	250	723.2
RX power (mA)	3.8	9.6	24	18 (25)	19.7	37(42)	65
TX power (mA/dBm)	12/ 1.5	16.5 / 10	19/0	13/0	17.4 / 0	34(30)/ 0	65/ 0
Powerdown power (μA)	1	1	1.5	0.4	1	1	140
Turn on time (ms)	0.02	2	1.13	3	0.58	20	*
Modulation	OOK/ ASK	FSK	FSK,GFSK	GFSK	DSSS-O-QPSK	DSSS-O-QPSK	FHSS-GFSK
Packet detection	no	no	programmable	yes	yes	yes	yes
Address decoding	no	no	no	yes	yes	yes	yes
Encryption support	no	no	no	no	128-bit AES	no	128-bit SC
Error detection	no	no	yes	yes	yes	yes	yes
Error correction	no	no	no	no	yes	yes	yes
Acknowledgments	no	no	no	no	yes	yes	yes
Interface	bit	byte	packet/byte	packet/byte	packet/byte	packet/byte	packet
Buffering (bytes)	no	1	32	16	128	133	yes *
Time-sync	bit	SFD/byte	SFD/packet	packet	SFD	SFD	Blueto oth
Localization	RSSI	RSSI	RSSI	no	RSSI/LQI	RSSI/LQI	RSSI

Πίνακας 4 Χαρακτηριστικά σύγχρονων πομποδεκτών (COTS radios, commercial off the shelf) ιδανικών για WSN

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 αποθηκευμένο σε μια μνήμη flash αλλά, από τη στιγμή που δε στέλνεται κάπου ασύρματα μέσω του πομπού, η μονάδα κρυπτογράφησης του υλικού του πομπού δεν μπορεί να χρησιμοποιηθεί.

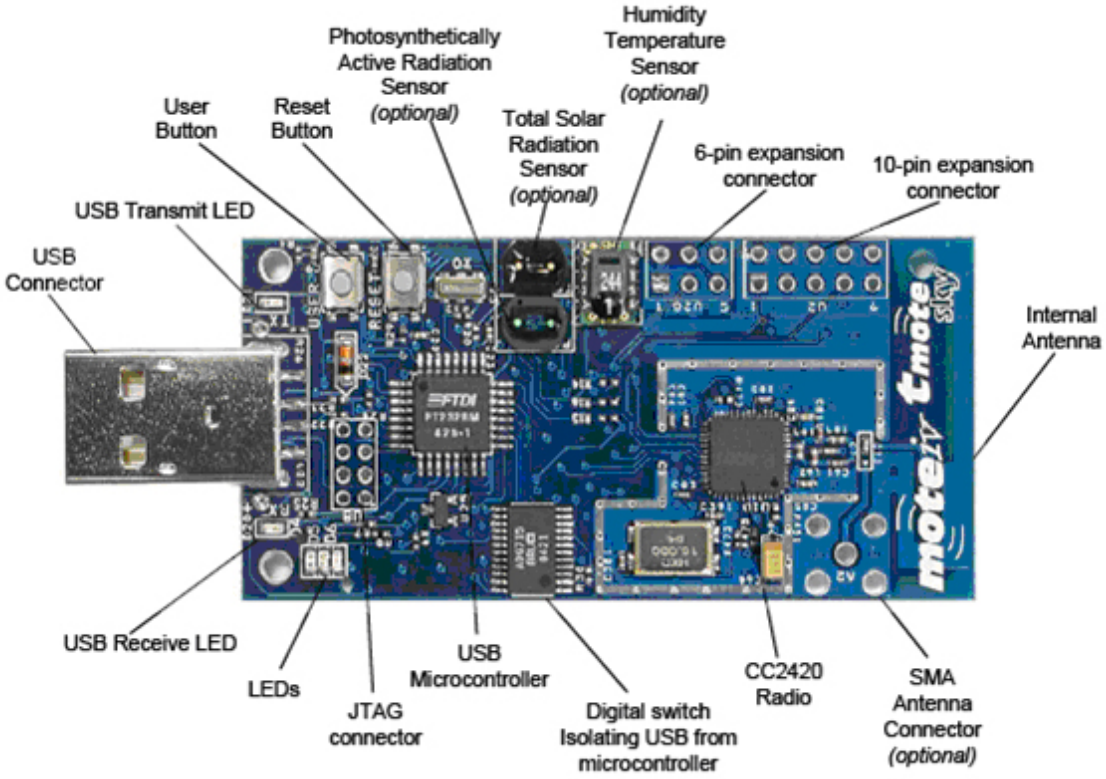
Οι τυπικές συνθήκες λειτουργίας του ασύρματου πομποδέκτη φαίνονται στον επόμενο πίνακα.

	MIN	NOM	MAX	Μονάδα
Τάση λειτουργίας κατά την ασύρματη εκπομπή (Vreg on)	2.1		3.6	V
Θερμοκρασία λειτουργίας	-40		85	°C
Εύρος συχνοτήτων RF	2400		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 σε 0 dBm		17.4		mA
Κατανάλωση ρεύματος: ασύρματη Λήψη		19.7		mA
Κατανάλωση ρεύματος: Radio on, ταλαντωτής on		365		μA
Κατανάλωση ρεύματος: κατάσταση αδράνειας, ταλαντωτής off		20		μA
Κατανάλωση ρεύματος: κατάσταση μη λειτουργίας, Vreg off			1	μA
Ρεύμα ρυθμιστή τάσης	13	20	29	μA
Χρόνος εκκίνησης ασύρματου ταλαντωτή		580	860	μs

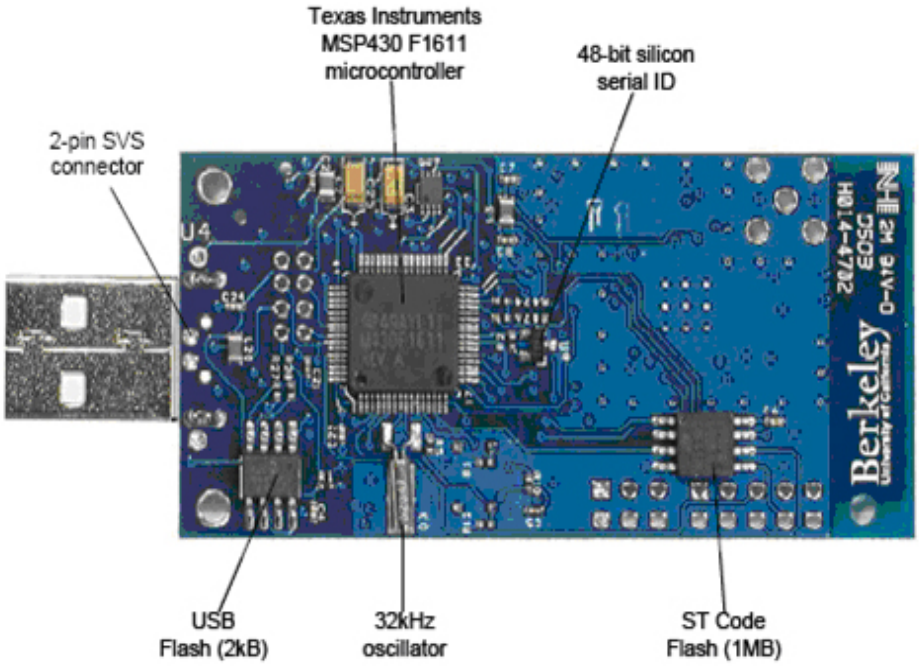
Πίνακας 5 Τυπικές συνθήκες λειτουργίας ασύρματου πομποδέκτη Chipcon CC2420

2.4.2.2 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 32 Εμπρόσθια όψη του Tmote-sky

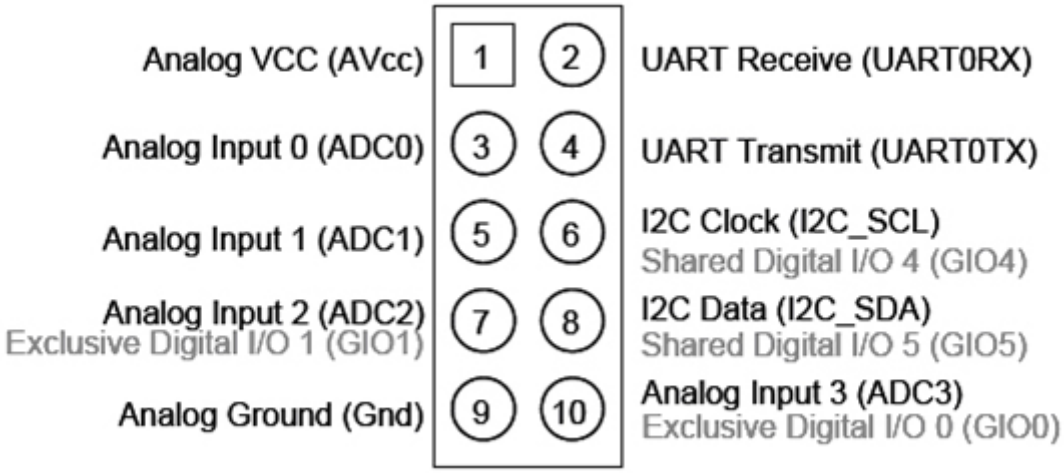


Εικόνα 33 οπίσθια όψη του Tmote-Sky

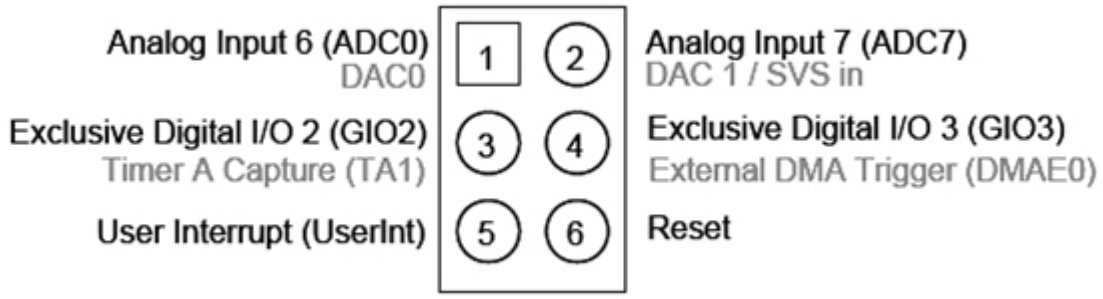
Το Tmote Sky χρησιμοποιεί μια ενσωματωμένη κεραία στα 2.4GHz, η οποία είναι μια μικροταινία σε σχήμα ανεστραμμένου F (Planar Inverted Folded Antenna – PIFA) και η οποία βρίσκεται τυπωμένη στην άκρη της πλακέτας, όπως φαίνεται και στην παραπάνω εικόνα (μπροστινή όψη). Η κεραία αυτή επιτυγχάνει εμβέλεια 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.

2.4.2.3 Συνδετήρας επέκτασης

Το tmote έχει δυο συνδετήρες επέκτασης, ένα των 10 ακροδεκτών (10-pin IDC header) και ένα 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 στις παρακάτω εικόνες:

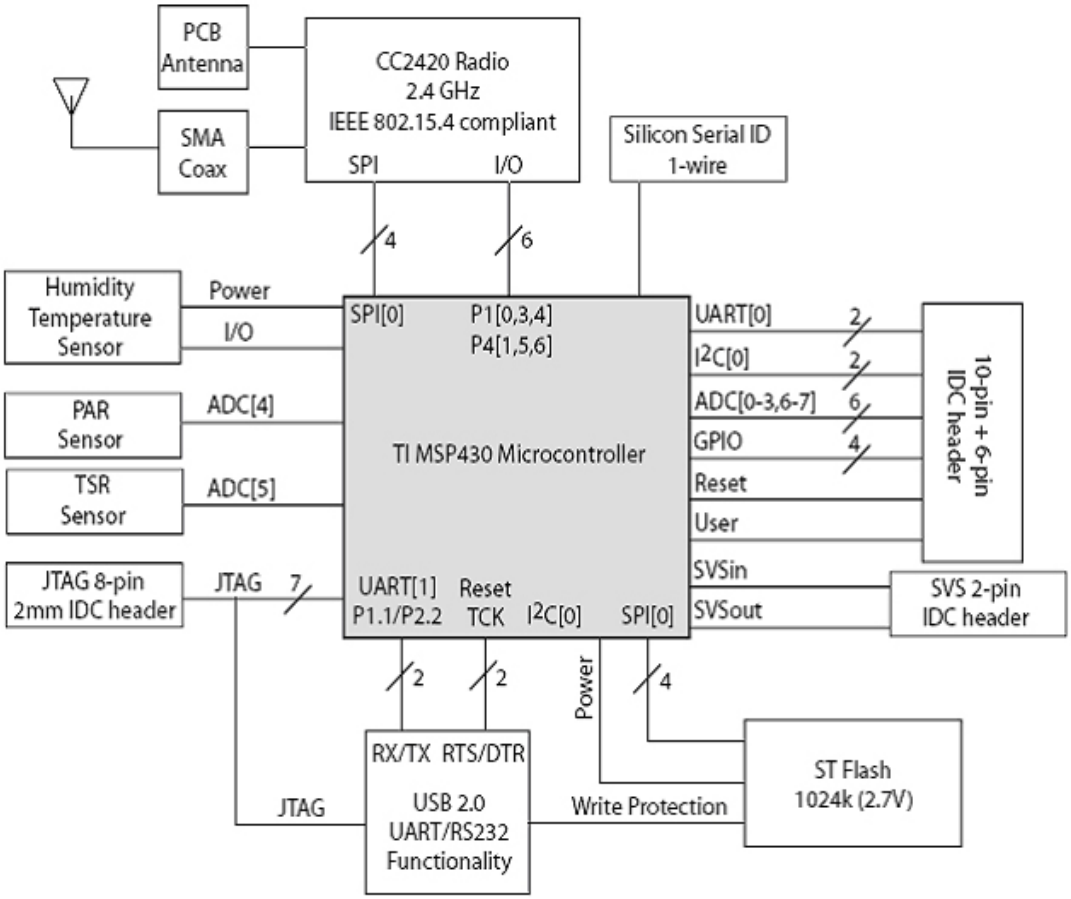


Εικόνα 34 Συνδετήρας επέκτασης 10 θέσεων (pins)



Εικόνα 35 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 36 Λειτουργικό μπλοκ διάγραμμα του Tmote

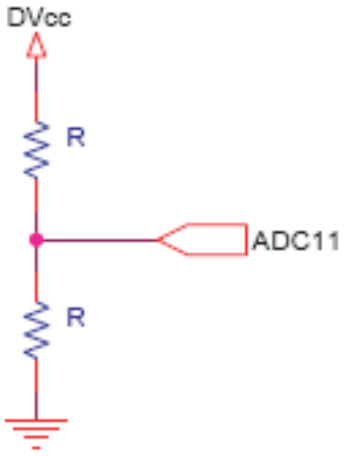
Το tmote είναι η πρώτη μονάδα που περιλαμβάνει ‘προστασία εγγραφής’ στο υλικό (hardware write-protection). Όταν συνδέεται στη USB θύρα, η προστασία εγγραφής απενεργοποιείται και ο πρώτος τομέας της μνήμης flash μπορεί να εγγραφεί. Όταν λειτουργεί με μπαταρίες (χωρίς USB), ο τομέας αυτός έχει προστασία εγγραφής. Η προστασία εγγραφής είναι σημαντική σε συστήματα που μπορούν 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 προήλθε 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 αναγνωρισθεί από το λογισμικό, η δυνατότητα της διακοπής της τροφοδοσίας στο συγκεκριμένο τμήμα της πλακέτας θα μπορούσε να διασώσει το όλο σύστημα.

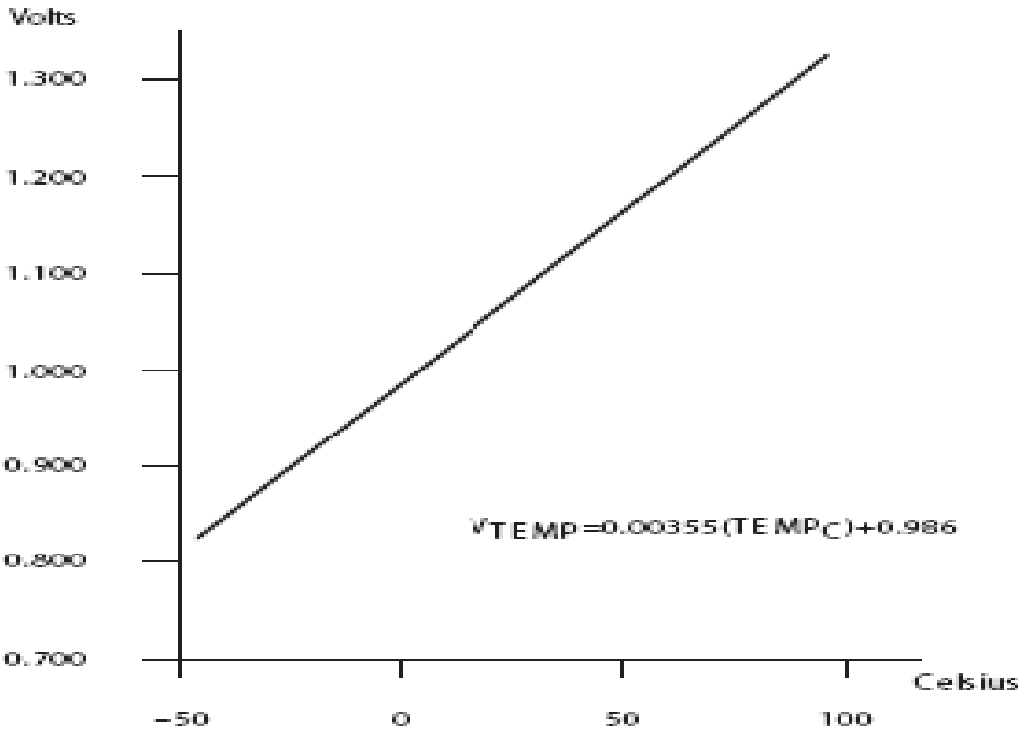
2.4.2.4 Αισθητήρες υγρασίας/θερμοκρασίας και φωτός

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 οποίες '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 37 Διάγραμμα του κυκλώματος αισθητήρα θερμοκρασίας

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



Εικόνα 38 Παράσταση τυπικής απόκλισης του εσωτερικού αισθητήρα φωτός

2.4.3 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 on)	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 (0dBm)	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

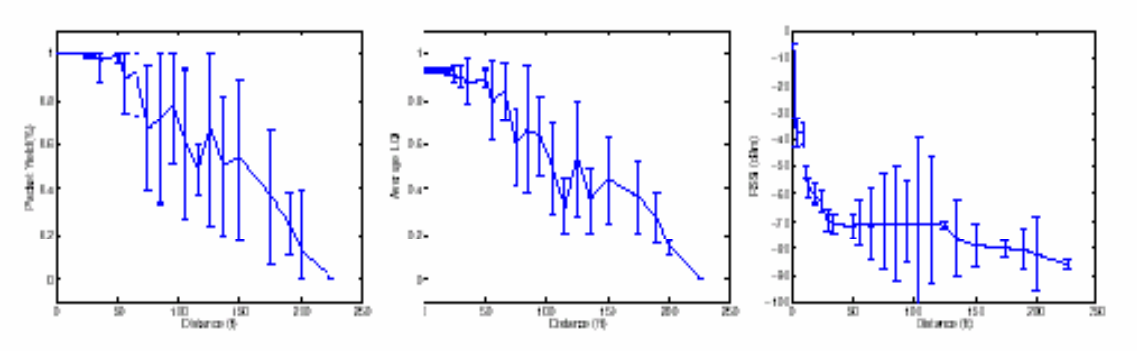
Πίνακας 6 Μετρούμενη κατανάλωση ρεύματος του Telos σε σύγκριση με τα Mica2 και MicaZ motes

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). 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 λαμβανόμενου 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.

2.5 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 επιλυθεί με αντικειμενοστρεφή 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 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. Η πρώτη αφορά στο ότι όλοι οι πόροι θεωρούνται στατικοί. Η δεύτερη στο ότι αντί της χρησιμοποίησης μιας γενικής εξυπηρέτησης λειτουργικού συστήματος, οι εφαρμογές κατασκευάζονται από ένα σύνολο στοιχείων συστήματος συσχετισμένων με συγκεκριμένο κώδικα. Τέλος, 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 και 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 ως 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**.

2.5.1 Το Λειτουργικό Σύστημα TinyOS

2.5.1.1 Γενικά

Το TinyOS είναι ένα λειτουργικό σύστημα ειδικά σχεδιασμένο για ασύρματα ενσωματωμένα συστήματα. Έχει ένα προγραμματιστικό μοντέλο προσαρμοσμένο για εφαρμογές 'οδηγούμενες από συμβάντα' (event-driven), ενώ ο πυρήνας του απαιτεί από κοινού μόνο 400bytes κώδικα και μνήμης δεδομένων. Όλη η οργανωτική δομή του TinyOS, οι βιβλιοθήκες και οι εφαρμογές είναι γραμμένες στη γλώσσα προγραμματισμού NesC. Το TinyOS έχει πολλές σημαντικές ιδιότητες που επηρέασαν το σχεδιασμό της γλώσσας NesC: αρχιτεκτονική κατανεμημένη σε επιμέρους στοιχεία (component-based), ένα απλό μοντέλο ταυτοχρονισμού βασισμένο σε συμβάντα (event-based concurrency model) και λειτουργίες διαχωρισμένες σε φάσεις.

2.5.1.2 Αρχιτεκτονική κατανεμημένη σε επιμέρους στοιχεία (Component based architecture)

Το λειτουργικό σύστημα TinyOS παρέχει ένα σύνολο από στοιχεία συστήματος τα οποία μπορούν να 'επαναχρησιμοποιηθούν'. Μια εφαρμογή συνδέει τα επιμέρους τμήματα-στοιχεία χρησιμοποιώντας μια καθορισμένη προδιαγραφή διασύνδεσης που είναι ανεξάρτητη από την υλοποίηση των στοιχείων. Κάθε εφαρμογή προσαρμόζει το σύνολο των στοιχείων που χρησιμοποιεί σύμφωνα με τις δικές του ανάγκες. Παρόλο που τα περισσότερα στοιχεία ενός λειτουργικού συστήματος είναι αυτοτελείς μονάδες λογισμικού, κάποια από αυτά αποτελούν απλά περιτυλίγματα γύρω από το υλικό, κάνοντας έτσι δύσκολη τη διάκριση σε κάποιον χρήστη που θέλει να αναπτύξει μια εφαρμογή. Αναλύοντας διαφορετικές υπηρεσίες ενός λειτουργικού σε ξεχωριστά στοιχεία-τμήματα μας δίνεται η δυνατότητα να αποκλείσουμε από την εφαρμογή μας υπηρεσίες και στοιχεία που δε θα μας χρησιμεύσουν.

2.5.1.3 Εργασίες και ταυτοχρονισμός βασισμένος σε γεγονότα (Tasks and event-based concurrency)

Το λειτουργικό σύστημα TinyOS εκτελεί μόνο ένα πρόγραμμα αποτελούμενο από επιλεγμένα στοιχεία συστήματος και στοιχεία κατασκευασμένα για την εφαρμογή που απαιτείται. Υπάρχουν δυο μηχανισμοί εκτέλεσης: οι εργασίες (tasks) και οι χειριστές συμβάντων υλικού (hardware event handlers). Οι εργασίες είναι λειτουργίες των οποίων η εκτέλεση μετατίθεται χρονικά. Εκτελούνται μέχρι την ολοκλήρωση τους και δεν προηγείται η μια της άλλης. Τα στοιχεία (components) μιας εφαρμογής μπορούν να θέσουν εργασίες προς εκτέλεση. Η χρήση αυτή των εργασιών από τα στοιχεία είναι επιθυμητή και λογική όταν οι χρονικές απαιτήσεις δεν είναι τόσο αυστηρές. Αυτό περιλαμβάνει ουσιαστικά τις περισσότερες λειτουργίες εκτός από την επικοινωνία σε χαμηλά επίπεδα (low-level). Για να εξασφαλίσουμε χαμηλή λανθάνουσα περίοδο κατά την εκτέλεση εργασιών, όλες οι ανεξάρτητες εργασίες πρέπει να είναι σχετικά σύντομες. Οι πολύπλοκες λειτουργίες θα έπρεπε να ‘διασκορπίζονται’ σε πολλαπλές μικρές εργασίες. Άλλωστε, οι απαιτήσεις των δικτύων αισθητήρων για μεγάλη χρονική διάρκεια ζωής απαγορεύουν βαρύ υπολογιστικό φορτίο, διατηρώντας το σύστημα αντιδραστικό.

Οι χειριστές συμβάντων υλικού εκτελούνται σε απάντηση μιας διακοπής υλικού (hardware interrupt) και επίσης εκτελούνται μέχρι ολοκλήρωσης, αλλά μπορούν να προηγηθούν της εκτέλεσης μιας εργασίας ή ενός άλλου χειριστή συμβάντος υλικού. Λέγοντας διακοπή υλικού ονομάζουμε ένα εξωτερικό συμβάν το οποίο προκαλεί τον επεξεργαστή να ξεκινήσει να εκτελεί μια προκαθορισμένη λειτουργία. Ο επεξεργαστής διαθέτει ένα πίνακα με διευθύνσεις λειτουργιών, μια για κάθε πιθανή διακοπή και αναμένει τον μεταγλωττιστή να συμπληρώσει κώδικα στην αρχή του μεταγλωττισμένου προγράμματος, ο οποίος θα συμπληρώνει κατάλληλα αυτόν τον πίνακα ανάλογα με το πρόγραμμα. Τα συμβάντα υποδηλώνουν είτε ολοκλήρωση μιας λειτουργίας διαχωρισμένης σε φάσεις είτε ένα άλλο συμβάν από το περιβάλλον (όπως η λήψη ενός μηνύματος ή η παρέλευση ενός χρονικού διαστήματος).

2.5.1.4 Λειτουργίες διαχωρισμένες σε φάσεις (Split-phase operations)

Επειδή οι εργασίες εκτελούνται χωρίς τη δυνατότητα η μια να προηγηθεί της άλλης, το TinyOS δεν έχει λειτουργίες φραγμού. Όλες οι λειτουργίες με αυξημένο

λανθάνοντα χρόνο εκτελούνται σε φάσεις: η αίτηση μιας λειτουργίας και η ολοκλήρωση της εκτελούνται ξεχωριστά. Οι εντολές (commands) είναι τυπικά αιτήσεις για εκτέλεση μιας λειτουργίας. Εάν η λειτουργία αυτή είναι διαχωρισμένη σε φάσεις, τότε η εντολή απαντά αμέσως, ενώ η ολοκλήρωσή της θα σηματοδοτήσει ένα συμβάν. Οι λειτουργίες που δε διαχωρίζονται σε φάσεις, όπως για παράδειγμα το άναμμα του led, δεν έχουν συμβάν ολοκλήρωσης. Ένα τυπικό παράδειγμα μιας λειτουργίας διαχωρισμένης σε φάσεις είναι η ασύρματη αποστολή ενός πακέτου. Κάποιο στοιχείο της εφαρμογής μπορεί να καλέσει την εντολή αποστολής send ώστε να ξεκινήσει την ασύρματη μετάδοση ενός μηνύματος, ενώ το στοιχείο που υλοποιεί την επικοινωνία σηματοδοτεί το συμβάν sendDone μόλις η μετάδοση του μηνύματος έχει ολοκληρωθεί. Κάθε στοιχείο υλοποιεί το μισό της συνολικής διφασικής λειτουργίας και καλεί το άλλο. Η διασύνδεση (wiring) συνδέει τόσο τις εντολές όσο και τα συμβάντα με κατάλληλο τρόπο ώστε να δράσουν από κοινού και ολοκληρωμένα.

Η διαμάχη πόρων τυπικά διαχειρίζεται μέσω απόρριψης σύγχρονων αιτήσεων. Στο παραπάνω παράδειγμα, εάν το στοιχείο που υλοποιεί την επικοινωνία δεν καταφέρει να χειριστεί πολλαπλές ταυτόχρονες λειτουργίες αποστολής, τότε σηματοδοτεί ένα μήνυμα λάθους όταν επιχειρηθεί μια ταυτόχρονη αποστολή. Εναλλακτικά, το στοιχείο που υλοποιεί την επικοινωνία θα μπορούσε να θέσει σε αναμονή την αίτηση για αποστολή προς μελλοντική επεξεργασία.

Το απλό μοντέλο ταυτοχρονισμού του TinyOS επιτρέπει υψηλό ταυτοχρονισμό με μικρό επίφορτο σε αντίθεση με το μοντέλο ταυτοχρονισμού βασισμένο σε νήματα στο οποίο οι στοίβες νημάτων καταναλώνουν πολύτιμη μνήμη. Παρόλα αυτά, όπως και κάθε ταυτόχρονο σύστημα, ο ταυτοχρονισμός και ο μη ντετερμινισμός μπορεί να είναι η αιτία πολύπλοκων προβλημάτων (bugs), συμπεριλαμβανομένου και του αδιεξόδου (deadlock).

2.5.2 Η γλώσσα προγραμματισμού NesC

2.5.2.1 Γενικά - Σχεδιασμός της γλώσσας NesC

Όπως αναφέρθηκε 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 της μεταγλώττισης.

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

Μερικές βασικές αρχές που χαρακτηρίζουν το σχεδιασμό της γλώσσας NesC είναι:

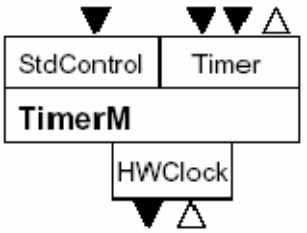
- Η γλώσσα NesC είναι μια προέκταση της C: Η C παράγει αποδοτικό κώδικα για όλους τους μικροελεγκτές που είναι δυνατό να χρησιμοποιηθεί σε ασύρματα δίκτυα αισθητήρων. Παρέχει όλες τις χαμηλού επιπέδου λειτουργίες που είναι απαραίτητες για την πρόσβαση στο υλικό και η αλληλεπίδραση με τον υπάρχον κώδικα C καθίσταται απλουστευμένη. Επιπλέον, πολλοί προγραμματιστές είναι εξοικειωμένοι με τη γλώσσα C. Η C έχει όμως και μερικά σημαντικά μειονεκτήματα: Παρέχει μικρή βοήθεια στη συγγραφή ασφαλούς κώδικα ή στη δόμηση των εφαρμογών.
- Συνολική προγραμματιστική ανάλυση: Τα προγράμματα στη NesC υπόκεινται σε συνολική προγραμματιστική ανάλυση (για ασφάλεια) και βελτιστοποίηση (για

επίδοση). Συνεπώς, δεν εξετάζεται ξεχωριστή σύνταξη-μεταγλώττιση στο σχεδιασμό της NesC. Το περιορισμένο μέγεθος προγράμματος στα motes κάνει αυτή την προσέγγιση δυνατή.

- Η NesC είναι μια 'στατική' γλώσσα: Δεν υπάρχει δυναμική παραχώρηση μνήμης και το διάγραμμα κλήσεων είναι πλήρως γνωστό κατά τη διάρκεια της μεταγλώττισης. Αυτοί οι περιορισμοί κάνουν την όλη προγραμματιστική ανάλυση και βελτιστοποίηση εξαιρετικά απλούστερη και ακριβέστερη. Ακούγονται πιο επαχθείς απ' όσο είναι στην πραγματικότητα: Το μοντέλο στοιχείων (component model) και οι παραμετροθετημένες διεπαφές (parameterized interfaces) της NesC ελαχιστοποιούν την ανάγκη για δυναμική παραχώρηση μνήμης και δυναμική διεκπεραίωση.
- Η NesC υποστηρίζει και αντανακλά το σχεδιασμό του λειτουργικού TinyOS: Η NesC βασίζεται στην ιδέα των επιμέρους στοιχείων, οπότε υποστηρίζει απευθείας το βασισμένο σε συμβάντα μοντέλο ταυτοχρονισμού του TinyOS. Επιπλέον, η NesC επιλαμβάνεται μοναδικά με το θέμα της ταυτόχρονης πρόσβασης σε μεριζόμενα δεδομένα. Στην πράξη, η γλώσσα NesC επέλυσε πολλά διφορούμενα στοιχεία σε σχέση με τις έννοιες του TinyOS περί τμημάτων-στοιχείων και ταυτοχρονισμού και το TinyOS εξελίχθηκε πάνω στο μοντέλο της NesC.

2.5.2.2 Προδιαγραφή στοιχείων-τμημάτων

Μια εφαρμογή γραμμένη σε γλώσσα NesC αποτελείται από ένα ή περισσότερα στοιχεία συνδεδεμένα κατάλληλα μεταξύ τους ώστε να σχηματίζουν ένα εκτελέσιμο αρχείο. Ένα στοιχείο παρέχει (provides) και χρησιμοποιεί (uses) διεπαφές (interfaces). Αυτές οι διεπαφές είναι τα μοναδικά σημεία πρόσβασης στο στοιχείο και είναι αμφίδρομες. Μια διεπαφή, γενικά, απεικονίζει μια υπηρεσία (όπως η αποστολή ενός μηνύματος) και προσδιορίζεται από ένα τύπο διεπαφής (interface type). Η εικόνα 40 δείχνει το στοιχείο TimerM, τμήμα της υπηρεσίας χρονομετρητή (timer) του TinyOS, το οποίο παρέχει τις διεπαφές StdControl και Timer και χρησιμοποιεί τη διεπαφή Clock (όλες οι διεπαφές φαίνονται στην εικόνα 41). Το στοιχείο TimerM παρέχει τη λογική που δημιουργεί την αντιστοιχία μεταξύ ενός ρολογιού υλικού (hardware clock) και της έννοιας του χρονομετρητή (timer) στο TinyOS.




```
module TimerM {
  provides {
    interface StdControl;
    interface Timer;
  }
  uses interface Clock as Clk;
} ...
```

Εικόνα 40 Προσδιορισμός και γραφική απεικόνιση του στοιχείου TimerM

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

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

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

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

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

Εικόνα 41 Μερικοί τύποι διεπαφών

Οι διεπαφές στη NesC είναι αμφίδρομες. Μια διεπαφή δηλώνει ένα σύνολο από συναρτήσεις, οι οποίες καλούνται εντολές (commands), τις οποίες ο πάροχος μιας διεπαφής πρέπει να υλοποιήσει, καθώς και ένα σύνολο συναρτήσεων, ονομαζόμενων συμβάντα (events), τις οποίες ο χρήστης μιας διεπαφής πρέπει να υλοποιήσει. Για να καλέσει κάποιο στοιχείο τις εντολές σε ένα interface, πρέπει να υλοποιήσει τα συμβάντα αυτής της διεπαφής. Ένα στοιχείο από μόνο του μπορεί να χρησιμοποιεί και να παρέχει περισσότερες από μια διεπαφές καθώς και πολλαπλά στιγμιότυπα της ίδιας διεπαφής. Για παράδειγμα, η διεπαφή Timer (Εικόνα 41) ορίζει τις εντολές start και stop και το συμβάν fired. Στην Εικόνα 40, οι παρεχόμενες διεπαφές είναι αυτές που φαίνονται πάνω από το στοιχείο TimerM ενώ, αυτές που χρησιμοποιούνται παριστάνονται κάτω από αυτό. Τα 'βέλη' που δείχνουν προς τα κάτω απεικονίζουν τις εντολές, ενώ αυτά που δείχνουν προς τα πάνω απεικονίζουν τα συμβάντα. Παρόλο που η ίδια αλληλεπίδραση μεταξύ του timer και του πελάτη (client) θα μπορούσε να πραγματοποιηθεί μέσω δυο ξεχωριστών διεπαφών (μια για τις εντολές start και stop

και μια για το συμβάν fired), η ομαδοποίηση αυτών των εντολών και συμβάντων στην ίδια διεπαφή κάνει τον προσδιορισμό πιο ξεκάθαρο και βοηθάει στην αποφυγή λαθών κατά τη διασύνδεση των στοιχείων μεταξύ τους. Οι εκτελούμενες σε δυο φάσεις λειτουργίες μπορούν εύκολα να μοντελοποιηθούν τοποθετώντας τις αιτούμενες εντολές και τα συμβάντα ανταπόκρισης στην ίδια διεπαφή. Στην Εικόνα 41 φαίνονται δυο τέτοιες περιπτώσεις. Η διεπαφή send έχει την εντολή send και το συμβάν sendDone μέσα στο διαχωρισμένο σε φάσεις πακέτο send. Όμοια και η διεπαφή ADC χρησιμοποιείται για την μοντελοποίηση της διαχωρισμένης σε φάσεις λειτουργίας ανάγνωσης των τιμών του αισθητήρα, όπως θα φανεί και παρακάτω στην εφαρμογή που υλοποιήθηκε.

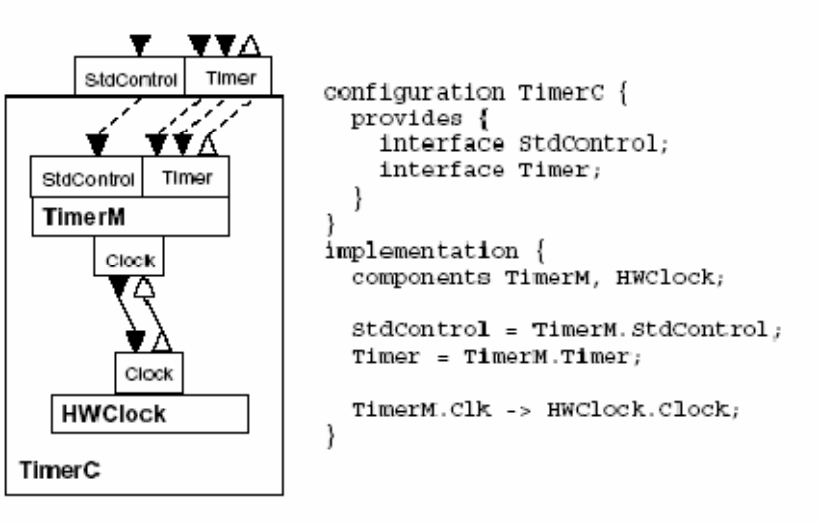
Ο διαχωρισμός αυτός στους ορισμούς των τύπων των διεπαφών από τη χρήση τους στα στοιχεία προωθεί τον ορισμό πρότυπων διεπαφών, κάνοντας τα στοιχεία (components) πιο ευέλικτα και επαναχρησιμοποιήσιμα. Αξίζει να σημειωθεί ότι ένα στοιχείο μπορεί να παρέχει και να χρησιμοποιεί τον ίδιο τύπο διεπαφής ή να παρέχει την ίδια διεπαφή περισσότερες από μια φορές. Σε αυτές τις περιπτώσεις το στοιχείο πρέπει να δώσει σε κάθε στιγμιότυπο της διεπαφής (interface instance) μια ξεχωριστή ονομασία χρησιμοποιώντας την χαρακτηριστική λέξη ‘as’, όπως φαίνεται και στο παράδειγμα της Εικόνας 40 για το Clk. Τέλος, ένα σημαντικό αλλά λεπτό σημείο είναι ότι οι αμφίδρομες διεπαφές μπορούν να υποστηρίξουν πολύ εύκολα τις διακοπές υλικού (hardware interrupts). Αντιθέτως, οι μονοσήμαντες διεπαφές που βασίζονται σε κλήσεις διαδικασιών υπαγορεύουν τη σταθμοσκόπηση υλικού (hardware polling) ή τη χρήση δυο ξεχωριστών διεπαφών για τις λειτουργίες υλικού και τις αντίστοιχες διακοπές (interrupts).

2.5.2.3 Υλοποίηση στοιχείων-τμημάτων

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

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

Το στοιχείο TimerC, η υπηρεσία δηλαδή χρονομετρητή (timer) του TinyOS, υλοποιείται σαν configuration έτσι όπως φαίνεται στην Εικόνα 42. Ο TimerC δημιουργείται από τη διασύνδεση των δυο υποστοιχείων που δίνονται κατά τη δήλωση των στοιχείων: του TimerM (Εικόνα 40) και του HWClock (πρόσβαση στο ρολόι του chip). Επίσης, ο TimerC ορίζει μέσα στον κώδικά του τις διεπαφές StdControl και Timer ως ισοδύναμες με τις αντίστοιχες διεπαφές που υλοποιούνται από τον TimerM (StdControl = TimerM.StdControl, Timer = TimerM.Timer) και συνδέει την διεπαφή του hardware ρολογιού (Clock) που χρησιμοποιείται από τον TimerM με αυτή που παρέχεται από τον HWClock (TimerM.Clk -> HWClock.Clock). Πρακτικά, τα βέλη στο configuration δείχνουν από το χρήστη μιας διεπαφής προς τον πάροχο της διεπαφής.



Εικόνα 42 Υπηρεσία χρονομετρητή (timer) του TinyOS: TimerC configuration

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

Τα περισσότερα στοιχεία στο TinyOS αντιπροσωπεύουν υπηρεσίες (όπως ο timer) ή τμήματα υλικού (όπως τα LEDs) και επομένως έχουν μια μοναδική υπόσταση. Παρόλα αυτά, κάποιες φορές είναι χρήσιμο να δημιουργήσουμε περισσότερες από μια υποστάσεις του ίδιου στοιχείου. Αυτό μπορεί να επιτευχθεί με τη χρήση ενός 'αφαιρετικού' στοιχείου και παραμέτρων.

2.5.2.4 Ταυτοχρονισμός (concurrency) και ατμητότητα (atomicity)

Στο περιβάλλον του TinyOS, ο κώδικας εκτελείται είτε ασύγχρονα σε απόκριση μιας διακοπής (interrupt) είτε σε μια προγραμματισμένη σύγχρονη λειτουργία. Για να διευκολυνθεί ο εντοπισμός συνθηκών ανταγωνισμού, διαχωρίζουμε τον κώδικα σε σύγχρονο και ασύγχρονο:

- Ασύγχρονος κώδικας (Asynchronous Code –AC): Κώδικας ο οποίος είναι προσβάσιμος από τουλάχιστον ένα χειριστή διακοπής υλικού (interrupt handler)
- Σύγχρονος κώδικας (Synchronous Code – SC): Κώδικας ο οποίος είναι προσβάσιμος μόνο από εργασίες (tasks).

Ο κανόνας που διέπει τις εργασίες, περί εκτέλεσης μέχρι ολοκλήρωσης και σε σειρά ακολουθίας οδηγεί άμεσα στην παρακάτω παραδοχή:

Ο σύγχρονος κώδικας είναι άτμητος (atomic) αναφορικά με άλλο σύγχρονο κώδικα.

Ο όρος ‘άτμητος – atomic’ σημαίνει ότι ο κώδικας αυτός θα εκτελείται χωρίς τη δυνατότητα ένας άλλος κώδικας να προηγηθεί. Ο κώδικας που περιλαμβάνει λειτουργίες διαχωρισμένες σε φάσεις, όπου εξ ορισμού θα περιλαμβάνει τουλάχιστον δύο εργασίες, δεν είναι άτμητος συνολικά, αλλά καθένα από τα τμήματά του είναι.

Παρόλο που η μη δυνατότητα υπερτέρησης (non-preemption) συντελεί στην αποφυγή ανταγωνισμών μεταξύ των εργασιών, υπάρχουν ακόμα κάποιες περιπτώσεις πιθανού ανταγωνισμού μεταξύ ασύγχρονου και σύγχρονου κώδικα, καθώς και μεταξύ δυο ασύγχρονων κωδίκων.

- 1ος ισχυρισμός: Κάθε πρόσβαση σε μια διαμοιραζόμενη παράμετρο από ασύγχρονο κώδικα είναι μια πιθανή περίπτωση ανταγωνισμού.
- 2ος ισχυρισμός: Κάθε πρόσβαση σε μια διαμοιραζόμενη παράμετρο που γίνεται παράλληλα τόσο από σύγχρονο όσο και ασύγχρονο κώδικα είναι πιθανή περίπτωση ανταγωνισμού.

Για να αποκατασταθεί η ατμητότητα (atomicity) σε αυτές τις περιπτώσεις. Ο προγραμματιστής έχει δύο επιλογές: είτε να μετατρέψει όλο το διαμοιραζόμενο κώδικα σε εργασίες (σύγχρονος κώδικας μόνο) είτε να κάνει χρήση των atomic sections (τμημάτων). Ένα atomic section είναι μια μικρή ακολουθία κώδικα που

εκτελείται χωρίς τη δυνατότητα υπερτέρησης. Με βάση τα παραπάνω, κάθε προσπάθεια πρόσβασης σε μια διαμοιραζόμενη παράμετρο πρέπει να πραγματοποιείται μέσα σε ένα atomic section. Αυτό μας οδηγεί στη διατύπωση της παρακάτω βασικής αρχής:

Κάθε πρόσβαση σε μια μεριζόμενη παράμετρο είτε δεν αποτελεί πιθανή κατάσταση ανταγωνισμού (σε περίπτωση σύγχρονου κώδικα μόνο) είτε συντελείται μέσα σε ένα atomic section.

Αυτή η αρχή απλά εγγυάται ότι κάθε μεμονωμένη πρόσβαση στην περίπτωση αυτή είναι ελεύθερη από ανταγωνισμούς. Αυτό όμως δε σημαίνει ότι μια λανθασμένη χρήση των atomic sections δεν μπορεί να οδηγήσει σε συνθήκες ανταγωνισμού.

2.5.2.5 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 διότι, 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).

2.5.2.6 Παραμετροθετημένες διεπαφές (Parameterized interfaces)

Στη NesC γίνεται ευρεία χρήση των παραμετροθετημένων διεπαφών. Μια παραμετροθετημένη διεπαφή (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. Οι παραμετροθετημένες διεπαφές χρησιμοποιούνται για τη μοντελοποίηση των Active Messages AM (Ενεργών Μηνυμάτων) του TinyOS. Στα Active Messages, τα πακέτα περιέχουν έναν αναγνωριστικό αριθμό ο οποίος προσδιορίζει ποιος χειριστής συμβάντων υλικού πρέπει να εκτελεστεί. Η διασύνδεση μιας parameterized διεπαφής πρέπει να ορίζει μια συγκεκριμένη διεπαφή μέσω μιας σταθεράς (τιμής).

Σε ένα module, οι υλοποιήσιμες εντολές και τα συμβάντα μιας parameterized διεπαφής λαμβάνουν επιπλέον παραμέτρους που προσδιορίζουν την επιλεγμένη διεπαφή 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.

2.5.3 Δομή Πακέτων και επικοινωνία στο TinyOS

2.5.3.1 Υπηρεσία επικοινωνίας – τρόπος μετάδοσης πακέτων

Μια από τις σημαντικότερες υπηρεσίες του TinyOS και γενικότερα των δικτύων αισθητήρων είναι αυτή της επικοινωνίας. Όπως και σε άλλα δίκτυα δεδομένων, η βασική μονάδα επικοινωνίας είναι τα πακέτα. Ένα πακέτο δεν είναι τίποτα άλλο από μια σειρά bits με συγκεκριμένη αρχή και τέλος. Το TinyOS ορίζει μια συγκεκριμένη δομή στα πακέτα που ανταλλάσσονται μεταξύ των κόμβων-αισθητήρων.

Το βασικό στοιχείο του TinyOS που παρέχει την επικοινωνία μεταξύ των κόμβων είναι το GenericComm. Το GenericComm παρέχει τις διεπαφές SendMsg και ReceiveMsg. Τα δεδομένα που αποστέλλονται είναι σε μορφή ‘raw data’ και η δομή τους, όπως χρησιμοποιούνται από την SendMsg.send(), είναι struct TOS_Msg, όπως ορίζεται στο αρχείο AM.h. Περιέχει πεδία για τη διεύθυνση αποστολής, τον τύπο μηνύματος (AM handler ID), το μήκος, το ωφέλιμο φόρτο δεδομένων (payload data) και άλλα που θα περιγραφούν και στην επόμενη παράγραφο. Η διασύνδεση (wiring) στο στοιχείο GenericComm γίνεται με τη χρήση παραμετροθετημένης διεπαφής (parameterized interface). Αυτό γιατί πολλά στοιχεία μπορεί να μοιράζονται το ίδιο κανάλι ταυτόχρονα αλλά το καθένα ενδιαφέρεται μόνο για τα δικά του δεδομένα. Κάθε πακέτο TinyOS περιέχει ένα πεδίο ‘group ID’ για αυτό το λόγο. Το μήκος του μηνύματος καταγράφεται επίσης μέσα σε ξεχωριστό πεδίο του πακέτου ώστε ο δέκτης να γνωρίζει πόσα bytes να περιμένει κατά τη λήψη του μηνύματος και να διευκολύνεται έτσι η επικοινωνία. Επίσης, το μέγιστο μήκος του ωφέλιμου πεδίου ‘payload’ είναι ορισμένο σε 29 bytes αλλά υπάρχει η δυνατότητα να αυξηθεί και παραπάνω. Συνήθως, για διευκόλυνση ορίζεται πάντα μια δομή για κάθε τύπο μηνύματος που επιθυμείται να σταλεί, ανάλογα με την εφαρμογή μας, και τοποθετείται στο πεδίο payload data του πακέτου TOS_Msg. Με αυτό τον τρόπο επιτυγχάνεται καλύτερη διαχείριση των μηνυμάτων που λαμβάνονται, ιδιαίτερα όταν η επικοινωνία γίνεται μεταξύ ενός υπολογιστή και ενός κόμβου-αισθητήρα.

Κάθε κόμβος-αισθητήρας που θέλει να στείλει ένα μήνυμα καταχωρεί ένα χώρο ενδιάμεσης μνήμης (buffer) για την αποθήκευση των δεδομένων του μηνύματος πριν το στείλει στο send(). Στη συνέχεια, εάν το στρώμα μηνυμάτων (message layer) έχει χώρο για να αναλάβει την κυριότητα αυτού του τμήματος ενδιάμεσης μνήμης, τότε

κάνει αποδεκτή την αίτηση αποστολής, παίρνει την κυριότητα και το στοιχείο που έκανε την αίτηση οφείλει να μην τροποποιήσει το χώρο αυτό ενδιάμεσης μνήμης μέχρι την ολοκλήρωση της αποστολής (SendDone). Για το λόγο αυτό, όπως θα φανεί και στην εφαρμογή που υλοποιήθηκε, είναι συνηθισμένη τακτική να χρησιμοποιείται κώδικας που κατά κάποιο τρόπο θα 'κλειδώνει' το buffer από κάθε απόπειρα πρόσβασης μέχρι ολοκλήρωσης της αποστολής.

Η διαχείριση μνήμης για τα εισερχόμενα μηνύματα είναι ενδογενώς δυναμική. Όταν το μήνυμα καταφθάνει, τότε συμπληρώνει ένα χώρο ενδιάμεσης μνήμης (buffer). Στη συνέχεια το στρώμα AM (Active Message layer) το αποκωδικοποιεί και το στέλνει προς διεκπεραίωση. Μόλις ολοκληρωθεί η επεξεργασία, το στοιχείο της εφαρμογής πρέπει να αποστείλει πίσω στο κατώτερο στρώμα ένα δείκτη που θα δείχνει στην ενδιάμεση αυτή μνήμη. Στην περίπτωση όμως που η εφαρμογή χρειάζεται να αποθηκεύσει τα περιεχόμενα του μηνύματος προς περαιτέρω χρήση, μπορεί να αντιγράψει το μήνυμα σε ένα νέο buffer ή, καλύτερα, να επιστρέψει ένα νέο ελεύθερο χώρο μνήμης (buffer) για χρήση από το στρώμα δικτύου (buffer swap).

Τέλος, θα πρέπει να αναφέρουμε ότι υπάρχουν κάποια άλλα στοιχεία που εξασφαλίζουν την αξιόπιστη ασύρματη μετάδοση των πακέτων και τα οποία προσαρτώνται σε πεδία στο πακέτο πριν την αποστολή του. Ένα από αυτά είναι και η χρήση του προοιμίου 'preamble'. Το προοίμιο 'preamble' αποτελεί μια συγκεκριμένη διάταξη από bit που αποστέλλονται πριν τη μετάδοση ενός πακέτου. Όπως είναι γνωστό, το μήνυμα φτάνει στο στρώμα ραδιοζεύξης και ξεκινά η αποστολή του από τον πομπό σύμφωνα με τα bit του μηνύματος και τον τρόπο πρόσβασης στο μέσο (Media Access Control) που έχει καθοριστεί. Ένας ραδιοδέκτης ακούει σταθερά ένα κανάλι και προσπαθεί να ξεχωρίσει αν το επίπεδο της ενέργειας είναι αρκετά υψηλό ή τα χαρακτηριστικά του σήματος είναι αρκετά ώστε να αποτελούν ένα μεταδιδόμενο bit. Εάν διαπιστώσει ότι όντως έτσι είναι, ξεκινά να δειγματοληπτεί με τη συχνότητα που έχει οριστεί από το πρότυπο ασύρματης μετάδοσης που χρησιμοποιεί. Οπότε εάν τα bit που αποτελούν το preamble είναι σωστά, ο δέκτης είναι σίγουρος ότι θα λάβει ένα πακέτο μηνύματος. Η διάρκεια που ο δέκτης θα δειγματοληπτεί το κανάλι, πριν σταματήσει, καθορίζεται από το πεδίο που ορίζει το μήκος του πακέτου. Βέβαια, επειδή ο θόρυβος είναι πιθανόν να προκαλέσει σφάλματα κατά την αποστολή του πακέτου, τα οποία και πρέπει να διορθωθούν ή να εντοπιστούν πριν το μήνυμα

προωθηθεί από το δέκτη σε ανώτερα στρώματα, γίνεται προσάρτηση στο σώμα του πακέτου ενός πεδίου CRC κώδικα. Με κατάλληλη χρήση του πεδίου αυτού από το δέκτη είναι δυνατός ο εντοπισμός σφαλμάτων στο πακέτο μηνύματος που έλαβε.

Για τη μείωση της πιθανότητας λάθους μετάδοσης μηνυμάτων και την όσο το δυνατόν πιο αξιόπιστη μετάδοση σε πολλούς κόμβους ταυτόχρονα, έχουν αναπτυχθεί διάφορα πρωτόκολλα Ελέγχου Πρόσβασης στο Μέσο (MAC protocols). Κάποια από αυτά μπορεί να καθορίζουν τη μετάδοση σε συγκεκριμένη συχνότητα για τον κάθε κόμβο (FDMA – Frequency Division Multiple Access). Πολλαπλή προσπέλαση με διαίρεση συχνότητας), άλλα μπορεί να ορίζουν στους κόμβους ένα συγκεκριμένο χρονικό διάστημα για να μεταδώσουν, έτσι ώστε κανένας να μην μεταδίδει ταυτόχρονα την ίδια στιγμή (TDMA – Time Division Multiple Access - Πολλαπλή προσπέλαση με διαίρεση χρόνου) ή να εφαρμόζουν πολλαπλή πρόσβαση διαίρεσης κώδικα (CDMA). Επίσης, συχνή είναι η χρήση πρωτοκόλλου CSMA (Carrier Sense Multiple Access) όπου οι κόμβοι ‘ακούν’ το κανάλι για ένα μικρό παράθυρο χρόνου πριν στείλουν το πακέτο, για την αποφυγή συγκρούσεων. Αυτός είναι και ο τρόπος πρόσβασης που ακολουθούν τα περισσότερα ασύρματα δίκτυα αισθητήρων που χρησιμοποιούν το TinyOS σαν λειτουργικό σύστημα.

Η ύπαρξη πεδίων με τις διευθύνσεις των κόμβων-αισθητήρων στα πακέτα που μεταδίδονται ασύρματα έδωσε τη δυνατότητα για βελτίωση της αξιοπιστίας μετάδοσης με χρήση πακέτων ACK για την επιβεβαίωση της λήψης πακέτων από τον παραλήπτη, καθώς και με χρήση του αμφίδρομου πρωτοκόλλου RTS-CTS, στο οποίο πομπός και δέκτης ανταλλάσουν πακέτα Request To Send και Clear To Send πριν ξεκινήσουν την μετάδοση των δεδομένων. Επίσης η δημιουργία σε κάθε κόμβο μιας λίστας με τους γείτονές του (‘πίνακας γειτόνων’ – ‘neighbor table’), η σωστή διαχείριση του ανάλογα με την εφαρμογή, και η χρήση αποδοτικών αλγορίθμων για τη διάδοση των πακέτων από τους κόμβους σε ένα κόμβο πηγή προσέφερε σημαντικές ελπίδες για ανάπτυξη αυτοδικτυούμενων συστημάτων αισθητήρων με δυνατότητα αποκατάστασης χαμένων κόμβων και επίσης δυνατότητες χωρικού εντοπισμού της θέσης τους, θέματα που αποτελούν ακόμα πεδία ερευνών για πολλές επιστημονικές ομάδες.

2.5.3.2 Γενική δομή πακέτων TinyOS

Όπως είδαμε και προηγουμένως, τα πακέτα στο TinyOS ακολουθούν μια γενική δομή η οποία διαφοροποιείται μερικώς, ανάλογα με την εφαρμογή που υλοποιεί ο κάθε χρήστης. Κάποια από τα σημεία που πρέπει να γνωρίζει κάποιος για να κατανοήσει καλύτερα τη σύνθεση των πακέτων αυτών φαίνονται παρακάτω:

• Ένα πακέτο δεδομένων στο TinyOS έχει μέγιστο μήκος 255 byte

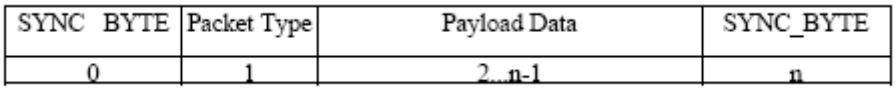
• Τα 'πρωτογενή δεδομένα' (raw data) περιβάλλονται και από τις δυο άκρες από ένα byte πλαισίου συγχρονισμού 0x7E. Αυτό χρησιμοποιείται για τη διαπίστωση της αρχής και του τέλους του κάθε πακέτου στη ροή των δεδομένων

• Το πακέτο 'raw data' χρησιμοποιεί ένα byte διαφυγής 0x7D. Αυτό χρειάζεται στην περίπτωση που ένα byte στο πεδίο ωφέλιμου φόρτου (payload data) είναι το ίδιο με ένα δεσμευμένου κώδικα byte, όπως το byte πλαισίου συγχρονισμού 0x7E. Σε αυτή την περίπτωση το byte διαφυγής θα προηγηθεί των δεδομένων του payload data και τα ίδια θα μπουν αλλά μετά την πράξη τους με 'ΑΠΟΚΛΕΙΣΤΙΚΟ Ή' με το 0x20. Για παράδειγμα ένα byte πεδίου ωφέλιμων δεδομένων 0x7E θα εμφανιστεί στο πακέτο δεδομένων ως 0x7D 0x5E

• Τα δεδομένα των bytes στέλνονται/παρουσιάζονται σε little-endian μορφή. Για παράδειγμα, το πεδίο διεύθυνσης UART των 2 bytes: 0x007E θα εμφανιστεί σαν '7E 00' στο πακέτο δεδομένων. Δηλαδή MSB: 0x00 και LSB: 0x7E

Πακέτο Raw Data

Το παρακάτω διάγραμμα και ο πίνακας περιγράφουν τη δομή του πακέτου ‘raw data’.



SYNC BYTE	Packet Type	Payload Data	SYNC_BYTE
0	1	2...n-1	n

Εικόνα 43 Δομή πακέτου raw data

Byte#	Πεδίο	Περιγραφή
0	SYNC_BYTE	Πάντα 0x7E
1	Packet Type	Υπάρχουν 5 γνωστοι τύποι πακέτων: • P_PACKET_NO_ACK (0x42) – πακέτο χρήστη χωρίς απαίτηση ACK. • P_PACKET_ACK (0x41) – Πακέτο χρήστη. Απαιτείται ACK. Περιλαμβάνει ένα byte πρόθεμα. Παραλήπτης πρέπει να στείλει P_ACK απάντηση με πρόθεμα byte για περιεχόμενο. • P_ACK (0x40) - Η ACK απόκριση σε ένα πακέτο P_PACKET_ACK. Περιλαμβάνει το byte πρόθεμα στα περιεχόμενα. • P_UNKNOWN (0xFF) - Αγνωστος τύπος πακέτου.
2...n-1	Payload Data	Στις περισσότερες περιπτώσεις θα είναι ένα TinyOS μήνυμα ποικίλου μήκους, όπως περιγράφεται παρακάτω
n	SYNC_BYTE	Πάντα 0x7E

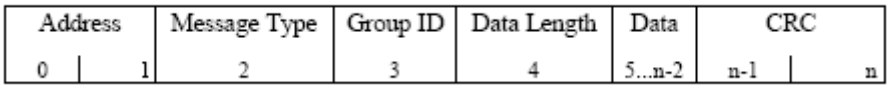
Πίνακας 7 Περιγραφή δομής πακέτου raw data

Πακέτο TinyOS

Το πεδίο ‘payload data’ θα είναι τυπικά τύπου TinyOS message, όπως καθορίζεται από τη δομή TOS_Msg στο αρχείο AM.h στου φακέλους της εγκατάστασης. Αυτή η δομή φαίνεται παρακάτω:

```
typedef struct TOS_Msg {
/* The following fields are transmitted/received on
the radio. */
uint16_t addr;
uint8_t type;
uint8_t group;
uint8_t length;
int8_t data[TOSH_DATA_LENGTH];
uint16_t crc;
/* The following fields are not actually transmitted
or received
*on the radio! They are used for internal accounting
only.
*The reason they are in this structure is that the AM
interface
*requires them to be part of the TOS_Msg that is
passed to
*send/receive operations.
*/
uint16_t strength;
uint8_t ack;
uint16_t time;
uint8_t sendSecurityMode;
uint8_t receiveSecurityMode;
} TOS_Msg
```

Πακέτο TOS_Msg



Εικόνα 44 Δομή πακέτου TOS_Msg

Byte#	Πεδίο	Περιγραφή
0-1	Address	Ένας απ'τους 3 τύπους τιμών: • Broadcast Address (0xFFFF) – μήνυμα σε όλους τους κόμβους. • UART Address (0x007e)- μήνυμα από ένα κόμβο σε μια σειριακή θύρα. Όλα τα εισερχόμενα μηνύματα θα έχουν αυτή τη διεύθυνση. • Node Address – Η μοναδική ID ενός κόμβου για τη λήψη μηνύματος.
2	Message Type	Active Message (AM) μοναδικό αναγνωριστικό για τον τύπο μηνύματος. Τυπικά κάθε εφαρμογή έχει το δικό του τύπο μηνύματος. Παραδείγματα: • AMTYPE_XUART = 0x00 • AMTYPE_MHOP_DEBUG = 0x03 • AMTYPE_SURGE_MSG =0x11 • AMTYPE_XSENSOR = 0x32 • AMTYPE_XMULTIHOP =0x33 • AMTYPE MH0P MSG = 0xFA
3	Group ID	Μοναδικά αναγνωρίσιμο για τον γκρούπ των κόμβων που συμμετέχουν στο δίκτυο. Η εξ' ορισμού τιμή είναι: 125 (0x7d). Μόνο τα motes με το ίδιο groupID θα επικοινωνούν μεταξύ τους
4	Data Length	Το μήκος *(l)*σε bytes του πεδίου data payload. Αυτό δεν περιλαμβάνει τον CRC ή bytes συγχρονισμού πλαισίου.
5…n-2	Payload data	Το πραγματικό περιεχόμενο του μηνύματος.. Τα δεδομένα βρίσκονται στα byte 5 εώς byte 5 σύν το μήκος των δεδομένων *(l)*. Τα δεδομένα είναι ανάλογα με τον τύπο μηνύματος. Συγκεκριμένοι τύποι φαίνονται στην επόμενη περιγραφή .
n-1, n	CRC	Κώδικας των 2 byte ο οποίος διασφαλίζει την ακεραιότητα του μηνύματος. Ο CRC περιέχει τον Packet Type συν ολόκληρο το unescaped TinyOS μήνυμα.

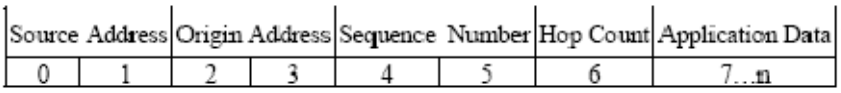
Πίνακας 8 Περιγραφή δομής πακέτου TOS_Msg

Multihop Μήνυμα

Το πεδίο payload data μέσα σε ένα TinyOS μήνυμα είναι raw data ειδικά για την εφαρμογή που έχει ενσωματωθεί στη μονάδα Tmote. Σε πολλές περιπτώσεις, ειδικότερα σε εφαρμογές που αξιοποιούν ad hoc δικτύωση βρόχου, η εφαρμογή χρησιμοποιεί το πρωτόκολλο Multihop μηνύματος. Ο ορισμός του μηνύματος Multihop φαίνεται παρακάτω:

```
typedef struct MultihopMsg {
uint16_t sourceaddr;
uint16_t originaddr;
int16_t seqno;
uint8_t hopcount;
uint8 t data[(TOSH DATA LENGTH - 7)];
}
```

Η μορφή και η περιγραφή του πακέτου φαίνονται παρακάτω:



Εικόνα 45 Δομή πακέτου Multihop Message

Byte#	Πεδίο	Περιγραφή
0,1	Source Address	Η διεύθυνση του κόμβου που έκανε την προώθηση.
2,3	Origin Address	Η διεύθυνση του κόμβου που δημιούργησε το μήνυμα.
4,5	Sequence Number	Χρησιμοποιείται για τον προσδιορισμό της διαδρομής και τον υπολογισμό χαμένων πακέτων
6	Hop Count	Χρησιμοποιείται για τον αριθμητικό υπολογισμό της διαδρομής. Αριθμός διασχιζόμενων κόμβων.
7...n	Application Data	Τα επακριβή δεδομένα. Ανάλογα με την εφαρμογή

Πίνακας 9 Περιγραφή της δομής του πακέτου Multihop Message

Τέλος, τα δεδομένα μέσα στο πεδίο application data του Multihop Message είναι raw data ειδικά ορισμένα για να ικανοποιούν την εφαρμογή του τελικού χρήστη. Ο τύπος των δεδομένων προσδιορίζεται από το πεδίο Message Type στο byte 2 του TinyOS μηνύματος

2.5.4 TOSSIM: Ένας εξομοιωτής για το TinyOS

Ο TOSSIM είναι ένας διακριτός εξομοιωτής γεγονότων (events) για ασύρματα δίκτυα αισθητήρων που βασίζονται στο TinyOS, ο οποίος αντιλαμβάνεται τη συμπεριφορά και τη διαδραστικότητα των motes σε επίπεδο bit, σε αντίθεση με άλλους εξομοιωτές που κάνουν το ίδιο σε επίπεδο πακέτου. Η αρχιτεκτονική του TOSSIM αποτελείται από πέντε μέρη:

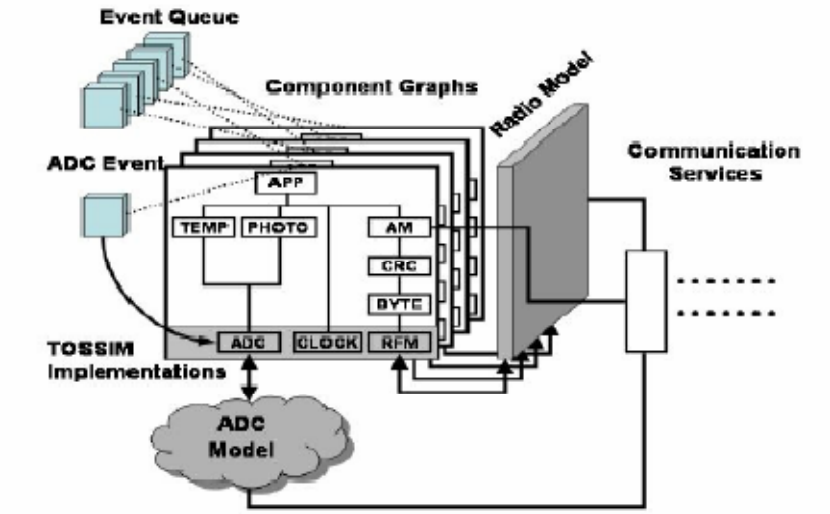
- Γράφοι συστατικών του TinyOS – υποστηρίζεται η μεταγλώττιση γράφων των συστατικών του TinyOS στην εξομοίωση από το χρήστη. Ένα συστατικό του TinyOS αποτελείται από πέντε αλληλοσχετιζόμενα μέρη: χειριστές εντολών (command handlers), χειριστές γεγονότων (event handlers), ένα συμπυκνωμένο προσωπικό πλαίσιο δεδομένων, μια δομή από προσωπικές μεταβλητές και μια ομάδα απλών tasks. Τα tasks, οι χειριστές εντολών και γεγονότων εκτελούνται στο περιβάλλον του πλαισίου. Οι εντολές και τα γεγονότα είναι μηχανισμοί επικοινωνίας μεταξύ των συστατικών και τα tasks χρησιμοποιούνται εσωτερικά.
- Μοντέλο εκτέλεσης – μια διακριτή ουρά γεγονότων υπεύθυνη για την ενεργοποίηση της εξομοίωσης. Τα μοντέλα του TOSSIM προκαλούν διακοπή μέσω γεγονότων εξομοίωσης, καθένα από τα οποία σχετίζεται με ένα mote, αφού ο προγραμματιστής (scheduler) εκτέλεσης των γεγονότων εκτελεί tasks στην ουρά των tasks του TinyOS και τα εκτελεί όλα (τα γεγονότα εξομοίωσης) σε σειρά FIFO. Η ουρά των tasks του σχετιζόμενου mote εκτελείται μαζί με το γεγονός, μέχρι ολόκληρη η ουρά να έχει τελειώσει.
- Μοντέλα - το RF και ο μετατροπέας αναλογικού-σε-ψηφιακό (ADC) χρησιμοποιούνται για να μοντελοποιήσουν τα χαρακτηριστικά των TinyOS motes. Τα RF μοντέλα ορίζουν τα χαρακτηριστικά που αφορούν στη μετάδοση από κόμβο σε κόμβο. Ένα ελάττωμα στο RF μοντέλο του TOSSIM είναι ότι η απόσταση δεν επηρεάζει την ισχύ του σήματος δημιουργώντας παρεμβολές γενικά χειρότερες από πραγματικές συμπεριφορές. Υπάρχουν δύο είδη RF μοντέλων.

Το «απλό» και αυτό που προκαλεί απώλειες ενέργειας, το «lossy». Το απλό RF μοντέλο τοποθετεί όλους τους κόμβους σε ένα μοναδικό «κελί», όπου κάθε bit που μεταδίδεται, λαμβάνεται χωρίς καθόλου σφάλμα. Ενώ κανένα bit δεν καταστρέφεται λόγω σφάλματος, δύο motes που εκπέμπουν ταυτόχρονα οδηγούν στο πρόβλημα της επικάλυψης των σημάτων, αλλά η πιθανότητα να εκπέμπουν δύο motes είναι πολύ χαμηλή εξαιτίας του πρωτοκόλλου CSMA του TinyOS.

Το «lossy» RF μοντέλο τοποθετεί τους κόμβους σε έναν κατευθυνόμενο γράφο. Κάθε ακμή (x,y) του γράφου σημαίνει ότι το σήμα του κόμβου x μπορεί να ακουστεί στον κόμβο y. Κάθε ακμή έχει μια τιμή που αναπαριστά την πιθανότητα ένα bit που αποστέλλεται από τον κόμβο x να είναι κατεστραμμένο (αντεστραμμένο), όταν ο y το ακούσει.

Ο TOSSIM παρέχει δύο ADC μοντέλα: το τυχαίο (random) και το γενικής χρήσης (generic). Ο ADC έχει διάφορα κανάλια που μπορούν να δειγματοληφθούν. Στο μοντέλο αυτό, όποτε ένα κανάλι δειγματοληπτείται επιστρέφει μια τυχαία 10-bit τιμή. Το γενικό μοντέλο παρέχει τις τυχαίες τιμές. Επιπλέον, παρέχει την πιθανότητα να ενεργοποιηθεί από εξωτερικές εφαρμογές, θέτοντας την τιμή για οποιαδήποτε ADC θύρα σε οποιοδήποτε mote.

- Συστατικά «γενίκευσης» υλικού (hardware abstraction components) – το TinyOS θεωρεί κάθε πόρο υλικού ως συστατικό (component). Ο TOSSIM αντικαθιστά ένα μικρό μόνο αριθμό από αυτά τα συστατικά, όπως το ADC, το ρολόι, το EEPROM, κτλ., και προσομοιώνει τη συμπεριφορά του υποκείμενου υλικού. Ο TOSSIM μοντελοποιεί αυτά τα συστατικά στο κομμάτι των συστατικών «γενίκευσης» υλικού της αρχιτεκτονικής.
- Υπηρεσίες επικοινωνίας – για να επιτρέψει σε εφαρμογές που τρέχουν στο PC να επικοινωνήσουν με τον TOSSIM μέσω TCP/IP.



Εικόνα 46 Η αρχιτεκτονική του TOSSIM: πλαίσια, γεγονότα, μοντέλα και υπηρεσίες

Ο TOSSIM διαβαθμίζεται σε χιλιάδες κόμβους, που μεταγλωττίζονται κατευθείαν από το TinyOS. Ο TOSSIM εξομοιώνει σε επίπεδο bit και αντικαθιστά το υλικό με συστατικά λογισμικού καθιστώντας δυνατές κλιμακούμενες εξομοιώσεις δικτύων αισθητήρων. Ο TOSSIM στοχεύει στην επίτευξη τεσσάρων στόχων:

Κλιμάκωση (scalability) – να είναι δυνατόν να χειριστεί μεγάλα δίκτυα από χιλιάδες κόμβους σε μεγάλο εύρος ρυθμίσεων.

Αρτιότητα (completeness) – να αντιλαμβάνεται με ακρίβεια τη συμπεριφορά ολόκληρου του συστήματος.

Αξιοπιστία (fidelity) – να αντιλαμβάνεται τη συμπεριφορά του δικτύου σε λειτουργία fine grain αποκαλύπτοντας απροσδόκητες αλληλεπιδράσεις. Ο TOSSIM βοήθησε στην εκσφαλμάτωση (debugging) των προβλημάτων στοίβας των δικτύων του TinyOS. Εξομοιώνει το δίκτυο σε επίπεδο bit.

Γεφύρωση (bridging) – γεφυρώνει το κενό ανάμεσα στον αλγόριθμο και την υλοποίηση, επιτρέποντας στους σχεδιαστές να ελέγξουν και να επιβεβαιώσουν τον κώδικα που θα τρέξει σε πραγματικό υλικό. Ο ελεγμένος κώδικας μπορεί να τοποθετηθεί κατευθείαν χωρίς καμία αλλαγή στην εφαρμογή.

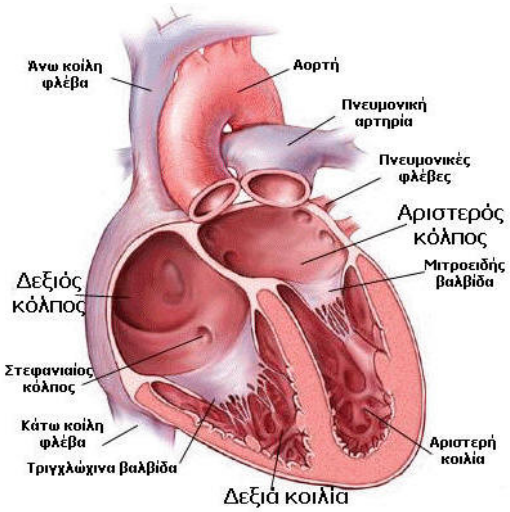
Κεφάλαιο 3

ΟΙ ΤΕΧΝΙΚΕΣ ΑΝΙΧΝΕΥΣΗΣ ΣΥΜΠΛΕΓΜΑΤΟΣ QRS ΣΕ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑ (ΗΚΓ)

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

3.1 Η καρδιά και ο καρδιακός κύκλος

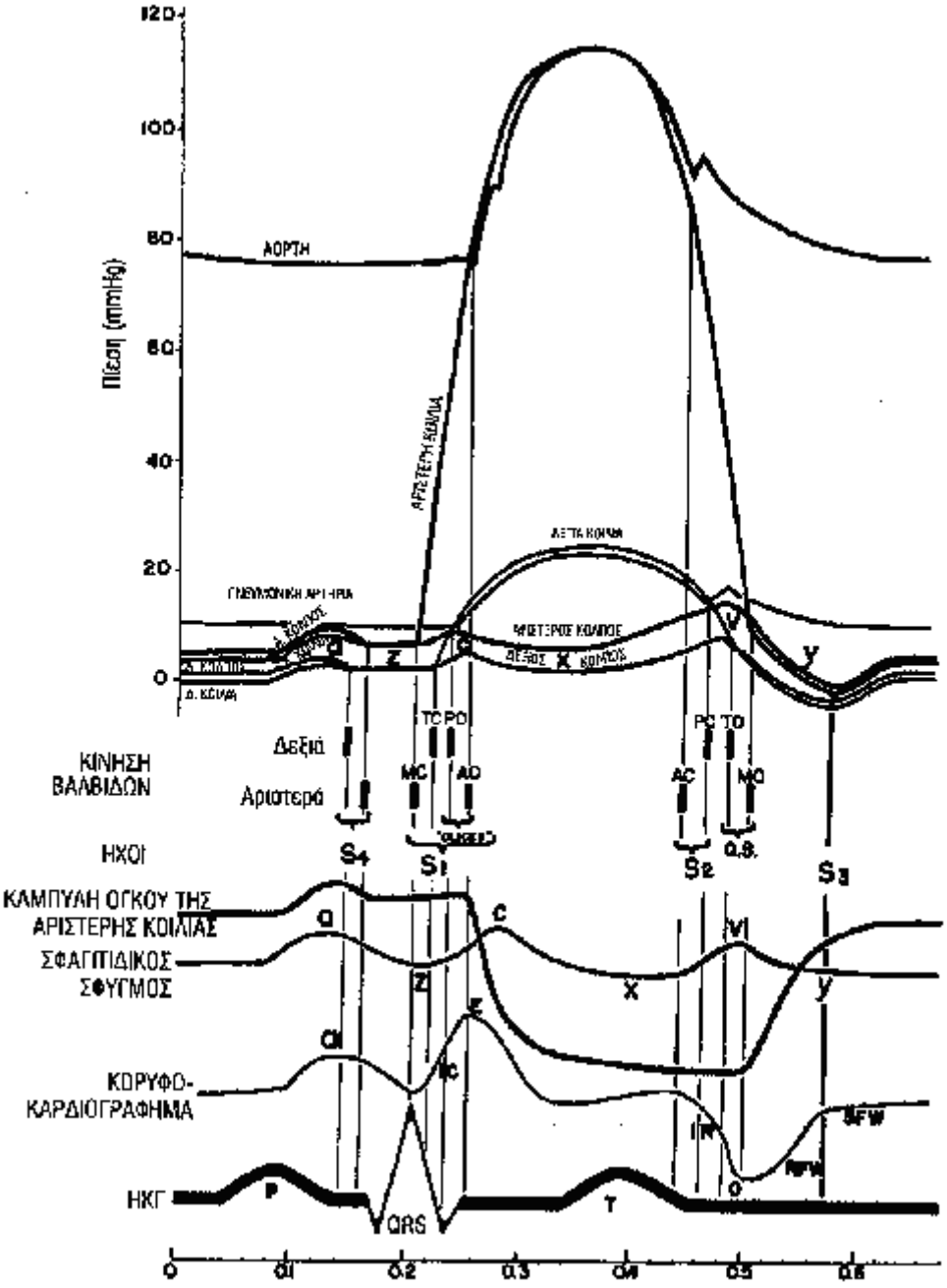
Η καρδιά είναι ένα μυώδες όργανο που συστέλλεται ρυθμικά λειτουργώντας σαν αντλία αίματος με τη βοήθεια του κυκλοφορικού συστήματος. Η καρδιά αποτελείται από τέσσερις ξεχωριστές αντλίες: δύο προαντλίες, τους κόλπους, και δύο προωθητικές αντλίες, τις κοιλίες. Η χρονική περίοδος από το τέλος μιας συστολής της καρδιάς μέχρι το τέλος της επόμενης συστολής ονομάζεται καρδιακός παλμός ή καρδιακός κύκλος. Ο κάθε καρδιακός παλμός αρχίζει με την αυτόματη γένεση ενός δυναμικού ενέργειας στο φλεβόκομβο. Αυτός ο κόμβος εντοπίζεται στο πρόσθιο τμήμα του δεξιού κόλπου, κοντά στην εκβολή της άνω κοίλης φλέβας, το δε δυναμικό ενέργειας επεκτείνεται με ταχύτητα και στους δυο κόλπους, και από εκεί, μέσα από το κολποκοιλιακό δεμάτιο, προς τις κοιλίες. Όμως, εξαιτίας της ειδικής διαρρύθμισης του συστήματος αγωγής από τους κόλπους στις κοιλίες, παρατηρείται καθυστέρηση κάπως μεγαλύτερη από 1/10 sec για τη δίοδο της διέγερσης από τους κόλπους στις κοιλίες. Με αυτόν τον τρόπο παρέχεται στους κόλπους η ευκαιρία να συστέλλονται πριν από τις κοιλίες, με αποτέλεσμα την άντληση αίματος προς τις κοιλίες πριν από την έντονη κοιλιακή συστολή.



Εικόνα 47 Η καρδιά

Ο καρδιακός παλμός αποτελείται από μια περίοδο χάλασης που ονομάζεται διαστολή, κατά τη διάρκεια της οποίας η καρδιά γεμίζει με αίμα, η οποία ακολουθείται από περίοδο συστολής, που ονομάζεται συστολή.

Στο ηλεκτροκαρδιογράφημα της εικόνας 48 καταγράφονται τα επάρματα (κύματα) P, QRS και T. Πρόκειται περί ηλεκτρικών δυναμικών, τα οποία παράγονται από την καρδιά και καταγράφονται με τον ηλεκτροκαρδιογράφο, από την επιφάνεια του σώματος. Το έπαρμα P προκαλείται από την επέκταση της εκπόλωσης στο μυοκάρδιο των κόλπων, η οποία ακολουθείται από τη συστολή των κόλπων, με συνέπεια την ελαφρά ανύψωση της καμπύλης της ενδοκολπικής πίεσης, αμέσως μετά το έπαρμα P.



Εικόνα 48 ηλεκτροκαρδιογράφημα στο οποίο φαίνονται τα διάφορα επάρματα

Μετά από 0,16 sec περίπου από την έναρξη του επάρματος P, εμφανίζονται τα επάρματα QRS τα οποία οφείλονται στην εκπόλωση των κοιλιών, η οποία προκαλεί την έναρξη της συστολής των κοιλιών και την ανιούσα φορά της ενδοκοιλιακής

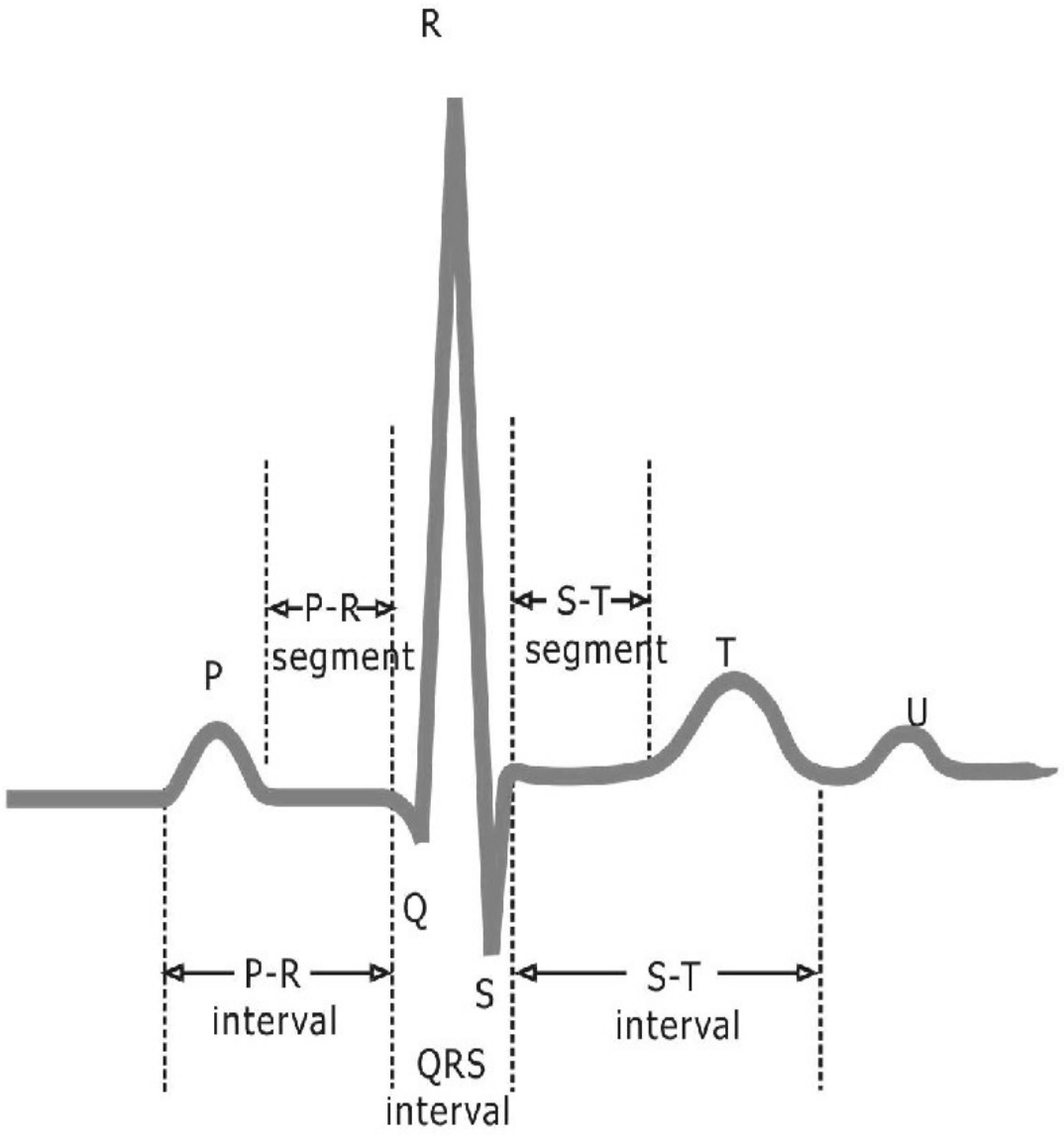
πίεσης. Κατά συνέπεια, το σύμπλεγμα QRS αρχίζει ελάχιστο χρόνο πριν από τη συστολή των κοιλιών.

Ακόμα, παρατηρείται στο ηλεκτροκαρδιογράφημα το κοιλιακό έπαρμα Τ. Αυτό αντιπροσωπεύει την περίοδο επαναπόλωσης των κοιλιών, κατά τη διάρκεια της οποίας οι μυϊκές ίνες του μυοκαρδίου των κοιλιών αρχίζουν να χαλαρώνουν. Γι' αυτό και το έπαρμα Τ εμφανίζεται ελάχιστο χρονικό διάστημα πριν από το τέλος της συστολής των κοιλιών.

Για τη γένεση και την αγωγή της διέγερσης μέσα από την καρδιά, η διέγερση προέρχεται από τον φλεβόκομβο. Κάθε φορά που διεγείρεται ο φλεβόκομβος, η διέγερση άγεται τόσο στις ίνες του κολποκοιλιακού κόμβου όσο και στις ίνες Purkinje, με αποτέλεσμα την εκπόλωση της διεγέρσιμης μεμβράνης τους. Στη συνέχεια, αυτές οι ίνες, όπως και οι ίνες του φλεβόκομβου, αναλαμβάνουν από το δυναμικό ενέργειας και υφίστανται υπερπόλωση. Οι ίνες, όμως, του φλεβόκομβου αποβάλλουν αυτήν την υπερπόλωση με πολύ ταχύτερο ρυθμό από τις άλλες δύο κατηγορίες ινών και εκπέμπουν μια καινούρια διέγερση πριν οι άλλες ίνες προλάβουν να φτάσουν σε κατάσταση για αυτοδιέγερση. Η καινούρια διέγερση προκαλεί και πάλι εκπόλωση τόσο των ινών του καλποκοιλιακού κόμβου όσο και των ινών Purkinje. Η διεργασία αυτή επαναλαμβάνεται συνεχώς με τον φλεβόκομβο να προκαλεί πάντα τη διέγερση των δυνητικά διεγέρσιμων αυτών ινών πριν να είναι δυνατή η αυτοδιέγερσή τους.

3.2 Το ηλεκτροκαρδιογράφημα

Κατά την επέκταση του επάρματος της διέγερσης στα διάφορα τμήματα της καρδιάς, ηλεκτρικά ρεύματα διατρέχουν τους ιστούς γύρω από την καρδιά, ένα μικρό δε μέρος από αυτά φτάνει μέχρι την επιφάνεια του σώματος. Εάν τοποθετηθούν ηλεκτρόδια πάνω στο δέρμα από τη μια και την άλλη πλευρά της καρδιάς, καθίσταται δυνατή η καταγραφή των ηλεκτρικών δυναμικών που παράγονται από αυτήν. Η καμπύλη που λαμβάνεται με αυτόν τον τρόπο ονομάζεται ηλεκτροκαρδιογράφημα και ένα φυσιολογικό ηλεκτροκαρδιογράφημα φαίνεται στην εικόνα 49.

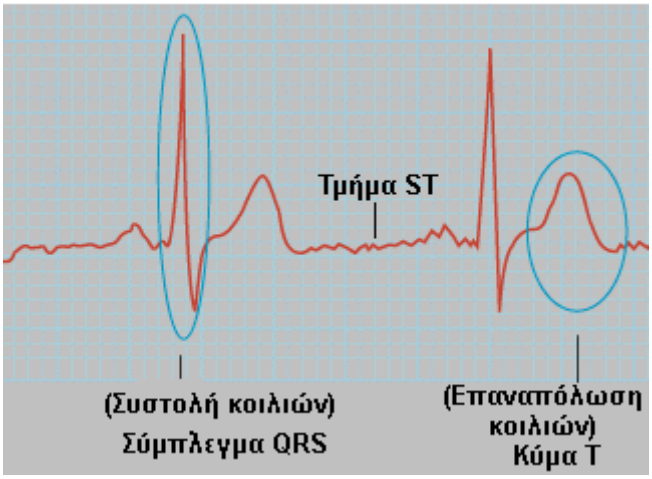


Εικόνα 49 Έπαρμα QRS σε φυσιολογικό ΗΚΓ

3.2.1 Τα χαρακτηριστικά του ηλεκτροκαρδιογραφήματος

Το φυσιολογικό ηλεκτροκαρδιογράφημα αποτελείται από ένα έπαρμα P, ένα «σύμπλεγμα QRS» και ένα έπαρμα T. Το σύμπλεγμα QRS συνήθως αποτελείται από τρία διαφορετικά επάρματα, το έπαρμα Q, το έπαρμα R και το έπαρμα S. Το έπαρμα P προκαλείται από ηλεκτρικά ρεύματα τα οποία παράγονται κατά την εκπόλωση των κόλπων πριν από τη συστολή τους, ενώ το σύμπλεγμα QRS προκαλείται από

ηλεκτρικά ρεύματα τα οποία παράγονται κατά την εκπόλωση των κοιλιών πριν από τη συστολή τους, δηλαδή κατά την επέκταση της εκπόλωσης στο μυοκάρδιο των κοιλιών. Κατά συνέπεια, τόσο το έπαρμα P, όσο και τα επάρματα που αποτελούν το σύμπλεγμα QRS, είναι επάρματα εκπόλωσης. Το έπαρμα T προκαλείται από ηλεκτρικά ρεύματα τα οποία παράγονται κατά την ανάνηψη των κοιλιών από την κατάσταση της εκπόλωσης. Η διεργασία αυτή επιτελείται στο μυοκάρδιο των κοιλιών 0,25 ως 0,35 sec μετά την εκπόλωση, αυτό δε το έπαρμα χαρακτηρίζεται ως έπαρμα εκπόλωσης. Δηλαδή, το ηλεκτροκαρδιογράφημα αποτελείται τόσο από επάρματα εκπόλωσης όσο και από επάρματα επαναπόλωσης.



Εικόνα 50 Επάρματα εκπόλωσης και επαναπόλωσης σε ΗΚΓ

3.2.2 Η συσχέτιση της συστολής των κόλπων και κοιλιών προς τα επάρματα του ηλεκτροκαρδιογραφήματος

Πριν να είναι δυνατή η συστολή του μυός, είναι απαραίτητη η επέκταση της εκπόλωσης στο μυοκάρδιο, για την έναρξη των χημικών διεργασιών της συστολής. Το έπαρμα P προκαλείται από την επέκταση της εκπόλωσης στους κόλπους, τα δε επάρματα QRS από την επέκταση της εκπόλωσης στις κοιλίες. Κατά συνέπεια, το

έπαρμα P εμφανίζεται αμέσως πριν από την έναρξη της συστολής των κόλπων, ενώ το σύμπλεγμα QRS αμέσως πριν από την έναρξη της συστολής των κοιλιών. Οι κοιλίες παραμένουν σε κατάσταση συστολής για μερικά χιλιοστά του sec μετά την επαναπόλωση, δηλαδή μετά το τέλος του επάρματος Τ.

Το έπαρμα επαναπόλωσης των κοιλιών στο φυσιολογικό ηλεκτροκαρδιογράφημα είναι το έπαρμα Τ. Φυσιολογικά, ορισμένες μυϊκές ίνες του μυοκαρδίου των κοιλιών αρχίζουν να επαναπολούνται 0,2 sec περίπου μετά την έναρξη του επάρματος εκπόλωσης, πολλές όμως άλλες ίνες αρχίζουν να επαναπολούνται βραδύτερα, μέχρι και 0,35 sec. Έτσι, η διεργασία της επαναπόλωσης επεκτείνεται μέσα σε σχετικά μεγάλο χρονικό διάστημα, περίπου 0,15 sec. Κατά συνέπεια, το έπαρμα Τ στο φυσιολογικό ηλεκτροκαρδιογράφημα συχνά είναι παρατεταμένο, η ηλεκτρική του τάση όμως είναι σημαντικά μικρότερη από την τάση του συμπλέγματος QRS, αυτό δε μερικώς οφείλεται στη μεγάλη του διάρκεια.

Οι κόλποι επαναπολούνται περίπου 0,15 ως 0,20 sec μετά το έπαρμα εκπόλωσης. Ο χρόνος όμως αυτός συμπίπτει με την εμφάνιση του συμπλέγματος QRS στο ηλεκτροκαρδιογράφημα. Συνεπώς, το έπαρμα επαναπόλωσης των κόλπων, γνωστό ως Τ των κόλπων, συνήθως επικαλύπτεται από το πολύ μεγαλύτερο σύμπλεγμα QRS. Για αυτό το λόγο, σπάνια μόνο είναι δυνατόν να παρατηρηθεί κολπικό έπαρμα Τ στο ηλεκτροκαρδιογράφημα.

3.2.3 Οι φυσιολογικές ηλεκτρικές τάσεις στο ηλεκτροκαρδιογράφημα

Η ηλεκτρική τάση των κυμάτων στο φυσιολογικό ηλεκτροκαρδιογράφημα εξαρτάται από τον τρόπο με τον οποίο τα ηλεκτρόδια τοποθετούνται στην επιφάνεια του σώματος. Όταν το ένα ηλεκτρόδιο τοποθετείται αμέσως πάνω από την καρδιά, και το δεύτερο ηλεκτρόδιο τοποθετείται σε κάποιο άλλο σημείο του σώματος, η ηλεκτρική τάση του συμπλέγματος QRS μπορεί να φτάνει τα 3 ή 4 mV. Όταν το ηλεκτροκαρδιογράφημα καταγράφεται με ηλεκτρόδια τοποθετημένα στα δύο άνω άκρα, είτε σε ένα άνω και σε ένα κάτω άκρο, η ηλεκτρική τάση του συμπλέγματος QRS είναι συνήθως 1 mV από την κορυφή του επάρματος R μέχρι το κάτω μέρος του επάρματος S. Εξάλλου η ηλεκτρική τάση του επάρματος P είναι 0,1 ως 0,3 mV και του επάρματος Τ από 0,2 ως 0,3 mV.

Το διάστημα P-Q ή P-R: Το χρονικό διάστημα που μεσολαβεί μεταξύ του επάρματος P και της αρχής του συμπλέγματος QRS είναι ο χρόνος που παρέρχεται από την έναρξη της συστολής των κόλπων μέχρι την έναρξη της συστολής των κοιλιών. Το χρονικό αυτό διάστημα ονομάζεται διάστημα P-Q. Το φυσιολογικό διάστημα P-Q είναι περίπου 0,16 sec. Αυτό το διάστημα σε μερικές περιπτώσεις ονομάζεται διάστημα P-R γιατί το Q συχνά απουσιάζει.

Το διάστημα Q-T : Η συστολή των κοιλιών πρακτικά διαρκεί από την αρχή του επάρματος Q μέχρι το τέλος του επάρματος T. Το χρονικό αυτό διάστημα ονομάζεται διάστημα Q-T και η φυσιολογική του διάρκεια είναι 0,35 sec.

3.3 Οι αρχές της ανίχνευσης του QRS με υπολογιστικές μεθόδους

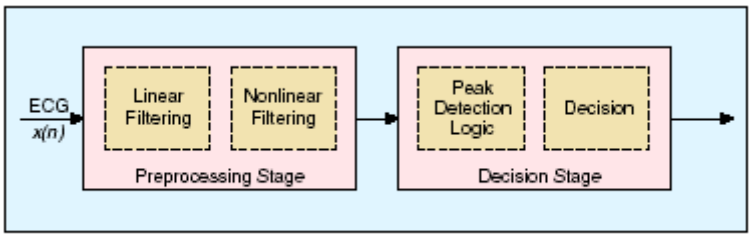
Το σύμπλεγμα QRS είναι η πιο χαρακτηριστική κυματομορφή μέσα στο ηλεκτροκαρδιογράφημα (ΗΚΓ). Καθώς αντιπροσωπεύει την ηλεκτρική δραστηριότητα της καρδιάς κατά τη συστολή των κοιλιών, ο χρόνος εμφάνισής του καθώς και το σχήμα του παρέχουν πολλές και σημαντικές πληροφορίες για την κατάσταση της καρδιάς. Λόγω του χαρακτηριστικού του σχήματος αποτελεί τη βάση για τον αυτοματοποιημένο προσδιορισμό του ρυθμού της καρδιάς, είναι σημείο αναφοράς για την κατηγοριοποίηση του καρδιακού κύκλου και, πολλές φορές, χρησιμοποιείται στην δημιουργία αλγορίθμων συμπίεσης των δεδομένων του ΗΚΓ. Με αυτή την έννοια, το σύμπλεγμα QRS παρέχει τους βασικούς άξονες για σχεδόν όλους τους αυτοματοποιημένους αλγόριθμους ανάλυσης του ΗΚΓ.

Το λογισμικό της ανίχνευσης του QRS αποτελεί πεδίο έρευνας για περισσότερα από 30 χρόνια. Η εξέλιξη των αλγόριθμων αυτών αντικατοπτρίζει ξεκάθαρα τις μεγάλες προόδους στην τεχνολογία των υπολογιστών. Παλαιότερα, το υπολογιστικό φορτίο ήταν αυτό που καθόριζε την πολυπλοκότητα και, κατά συνέπεια, την απόδοση του αλγόριθμου. Σήμερα, ωστόσο, ο βασικός στόχος είναι η υπολογιστική απόδοση. Το υπολογιστικό φορτίο γίνεται όλο και πιο ασήμαντο. Η μόνη, ίσως, εξαίρεση στην τάση αυτή είναι η ανάπτυξη αλγορίθμων ανίχνευσης QRS για συσκευές με λειτουργά μπαταρίας.

Μέσα στην τελευταία δεκαετία έχουν προταθεί πολλές νέες προσεγγίσεις στην ανίχνευση QRS. Για παράδειγμα, αλγόριθμοι από το πεδίο των τεχνητών νευρωνικών δικτύων, γενετικοί αλγόριθμοι, μετασχηματισμοί κυματομορφής, τράπεζες φίλτρων καθώς και μέθοδοι αυτοεκμάθησης, βασιζόμενες, κυρίως, σε μη γραμμικούς μετασχηματισμούς.

Η γρήγορη ανάπτυξη ισχυρών μικροϋπολογιστών είχε ως αποτέλεσμα την εκτεταμένη εφαρμογή λογισμικού αλγορίθμων ανίχνευσης QRS σε καρδιολογικές συσκευές. Τα τελευταία 35 χρόνια, το λογισμικό ανίχνευσης QRS αντικαθιστά όλο και περισσότερες συσκευές ανίχνευσης QRS.

Ήδη από τα πρώτα χρόνια της αυτοματοποιημένης ανίχνευσης QRS, είχε αναπτυχθεί μια αλγοριθμική δομή, η οποία ακολουθείται και σήμερα σε πολλούς αλγορίθμους. Όπως φαίνεται στο σχ. 2, χωρίζεται σε δύο στάδια. Το πρώτο στάδιο είναι αυτό της προεπεξεργασίας και της εξαγωγής χαρακτηριστικών τιμών χρησιμοποιώντας τόσο γραμμικά όσο και μη γραμμικά φίλτρα. Το δεύτερο, το στάδιο της απόφασης, είναι αυτό της ανίχνευσης των αιχμών και της λογικής της απόφασης. Πολλές φορές, προστίθεται και ένα ακόμη στάδιο επεξεργασίας για τον ακριβή προσδιορισμό και το χρονικό εντοπισμό του πιθανού QRS. Στη συνέχεια γίνεται παρουσίαση και διαχωρισμός των διαφόρων αλγορίθμων, με ιδιαίτερη προσοχή στα στάδια προεπεξεργασίας τους, διότι τα κυρίως στάδια επεξεργασίας βασίζονται κυρίως στην αυτοεκμάθηση και εξαρτώνται από τα αποτελέσματα της προεπεξεργασίας.



2. Common structure of the QRS detectors.

Εικόνα 51 Τυπική δομή αλγορίθμων ανίχνευσης QRS

3.3.1 Προσεγγίσεις βασισμένες στις παραγώγους και σε ψηφιακά φίλτρα

Οι τυπικές τιμές συχνοτήτων ενός συμπλέγματος QRS κυμαίνονται περίπου από 10 ως 25 Hz. Κατά συνέπεια, όλοι οι αλγόριθμοι ανίχνευσης QRS χρησιμοποιούν ένα στάδιο φιλτραρίσματος πριν από την πραγματική ανίχνευση, με σκοπό να μετριασθούν άλλα στοιχεία και παρεμβολές (artifacts) του σήματος όπως τα επάρματα P και T, η μετατόπιση του σήματος (baseline drift) και ο θόρυβος που περικλείει (incoupling noise). Ενόσω η μείωση της παρεμβολής των επαρμάτων P και T καθώς και της μετατόπισης απαιτούν υψιπερατά φίλτρα, ο περιορισμός του θορύβου επιτυγχάνεται, συνήθως, με την εφαρμογή βαθυπερατού φίλτρου. Ο συνδυασμός των δύο οδηγεί στην αποτελεσματική εφαρμογή ενός ζωνοπερατού φίλτρου, στην περίπτωσή μας με συχνότητες αποκοπής περίπου 10 και 25 Hz.

Σε πολλούς αλγορίθμους το βαθυπερατό και το υψιπερατό φίλτρο εφαρμόζονται χωριστά. Κάποιοι αλγόριθμοι χρησιμοποιούν μόνο το υψιπερατό φίλτρο. Στη συνέχεια, τα φιλτραρισμένα σήματα χρησιμοποιούνται για τη δημιουργία ενός χαρακτηριστικού σήματος, στο οποίο η ανίχνευση τoy συμπλέγματος QRS πραγματοποιείται συγκρίνοντας το χαρακτηριστικό αυτό σήμα με κάποια σταθερά ή προσαρμοζόμενα κατώφλια τιμών. Σχεδόν όλοι οι αλγόριθμοι χρησιμοποιούν επιπλέον κριτήρια επιλογής για την αποφυγή λανθασμένων ανιχνεύσεων.

3.3.1.1 Αλγόριθμοι με βάση τις παραγώγους

Το υψιπερατό φίλτρο συχνά υλοποιείται σαν διαφοριστής, και ιδιαίτερα στους παλαιότερους αλγόριθμους. Το στοιχείο αυτό καταδεικνύει την σημασία της απότομης κλίσης του συμπλέγματος QRS στη διαδικασία ανίχνευσής του.

Παραδείγματα εξισώσεων διαφορών πιθανών φίλτρων διαφορισμού είναι τα εξής:

$$y_1(n) = x(n+1) - x(n-1)$$
$$y_1(n) = 2x(n+2) + x(n+1) - x(n-1) - 2x(n-2)$$
$$y_1(n) = x(n) - x(n-1)$$
$$y_1(n) = \tilde{x}(n) - \tilde{x}(n-1)$$

όπου

$$\tilde{x}(n) = \begin{cases} |x(n)| & |x(n)| \geq \Theta \\ \Theta & |x(n) < \Theta| \end{cases}$$

και Θ είναι το κατώφλι του πλάτους όπως έχει προσδιορισθεί από το μετρούμενο σήμα ΗΚΓ x(n), από συναρτήσεις όπως η $\Theta_x = 0.3 \ldots 0.4 \cdot \max[x]$ όπου το μέγιστο (*max*) υπολογίζεται στο εκάστοτε τμήμα του σήματος. Μερικοί αλγόριθμοι υπολογίζουν και τη δεύτερη παράγωγο. Χαρακτηριστικά μεγέθη *z(n)* αυτών των αλγορίθμων είναι η ίδια η παράγωγος του σήματος ή ένας γραμμικός συνδυασμός της πρώτης και δεύτερης παραγώγου.

Η ανίχνευση ενός συμπλέγματος επιτυγχάνεται συγκρίνοντας το φιλτραρισμένο σήμα με ένα κατώφλι πλάτους. Συνήθως, τα επίπεδα κατωφλίου υπολογίζονται σε συνάρτηση με το σήμα, έτσι ώστε αν είναι δυνατή η προσαρμογή του σε αλλαγές των χαρακτηριστικών του σήματος.

Η λογική της ανίχνευσης των κορυφών του σήματος ολοκληρώνεται με την εφαρμογή επιπλέον κριτηρίων επιλογής, με σκοπό τη μείωση των λανθασμένων ανιχνεύσεων. Τέτοια κριτήρια συνήθως επιβάλλουν περιορισμούς στη διάρκεια και τη μορφή του σήματος ή εφαρμόζουν δευτερεύοντα κατώφλια για να αποκλείσουν τμήματα εκτός του QRS του ΗΚΓ που έχουν, όμως, παρόμοια χαρακτηριστικά.

3.3.1.2 Αλγόριθμοι βασισμένοι σε ψηφιακά φίλτρα

Ένας τέτοιος αλγόριθμος, προτείνει το φιλτράρισμα του ΗΚΓ παράλληλα από δύο διαφορετικά βαθυπερατά φίλτρα με διαφορετικές συχνότητες αποκοπής. Η διαφορά μεταξύ των εξόδων των δύο φίλτρων δίνει με αποτελεσματικό τρόπο το

φιλτραρισμένο σήμα $y_1(n)$, την έξοδο του ζωνοπερατού φίλτρου, από το οποίο προκύπτει, τελικά το $y_2(n) = y_1(n)\left[\sum_{k=-m}^{m} y_1{}^2(n+k)\right]^2$

Αυτή η μη γραμμική διαδικασία οδηγεί σε μια σχετική συμπίεση των μικρών τιμών και μια ομαλοποίηση των κορυφών του σήματος. Το τελικό προς ανίχνευση σήμα $z(n)$ προκύπτει από το $y_2(n)$ με εφαρμογή επιπλέον περιορισμών στην έξοδο του βαθυπερατού φίλτρου με την υψηλότερη συχνότητα αποκοπής. Τελικά, το κατώφλι υπολογίζεται δυναμικά από τη σχέση $\Theta = \max[z(n)]/8$.

Ένα άλλο παράδειγμα είναι ο αλγόριθμος MOBD (multiplication of backwards difference). Στην ουσία πρόκειται για ένα συνδυασμό λογικού ΚΑΙ παρακείμενων τιμών πλάτους της παραγώγου. Ο πολλαπλασιασμός των όπισθεν διαφορών μιας ακολουθίας N ορίζεται ως εξής:

$$z(n) = \prod_{k=0}^{N-1} |x(n-k) - x(n-k-1)|$$

Για την αποφυγή ακραίων τιμών σε τμήματα με θόρυβο, εφαρμόζονται επιπλέον περιορισμοί. Το κατώφλι Θ ορίζεται αρχικά στην μέγιστη τιμή του σήματος z_{max} και στη συνέχεια μειώνεται κατά το ήμισυ ανά τακτά χρονικά διαστήματα. Το κατώφλι περιορίζεται από ένα κατώτατο όριο, το οποίο ορίζεται επίσης δυναμικά.

Άλλοι αλγόριθμοι χρησιμοποιούν την ίδια προεπεξεργασία. Το ΗΚΓ περνάει από ζωνοπερατό φίλτρο και μετά διαφορίζεται. Το τελικό προς ανίχνευση σήμα υπολογίζεται υψώνοντας στο τετράγωνο και εξάγοντας τη μέση τιμή της εξόδου του διαφοριστή. Το φίλτρο και ο διαφοριστής χρησιμοποιούν συντελεστές καθορισμένους για μια εφαρμογή, σε επεξεργαστές σταθερών σημείων με μικρό μήκος λέξης. Για την ανίχνευση των κορυφών, εισάγεται μια μεταβλητή $\boldsymbol{\nu}$ που περιέχει την τιμή του τελευταίου χρονικά μεγίστου. Οι κορυφές του σήματος ανιχνεύονται συγκρίνοντας το σήμα με την τιμή του $\boldsymbol{\nu}$. Η κορυφή ανιχνεύεται όταν η τιμή του σήματος πέσει κάτω από την τιμή $\nu/2$. Τότε, ως πλάτος της κορυφής θεωρείται η τιμή της ν, και στη συνέχεια η ν παίρνει την τρέχουσα τιμή του σήματος. Η θέση καθώς και το ύψος της αιχμής (πλάτος του σήματος), τοποθετούνται σε ένα διάνυσμα το οποίο, στη συνέχεια, υφίσταται περαιτέρω επεξεργασία στο στάδιο της απόφασης. Στο στάδιο απόφασης,

γίνεται αναδρομικός υπολογισμός μιας τιμής της αιχμής και μιας τιμής του θορύβου καθώς και ο υπολογισμός του κατωφλίου βασισμένος στις παραπάνω τιμές.

Σε μια άλλη εφαρμογή, το σήμα *z(n)* υπολογίζεται με τρόπο παρόμοιο αλλά χρησιμοποιώντας διαφορετικά φίλτρα. Εδώ, το σήμα χωρίζεται σε τμήματα των 15 σημείων (δειγμάτων). Η μέγιστη τιμή κάθε τμήματος συγκρίνεται με μια μεταβαλλόμενη τιμή θορύβου και μια εκτιμώμενη τιμή κορυφής και στη συνέχεις καταχωρείται σύμφωνα με τη διαφορά με κάθε μία από αυτές. Ως θέση της κορυφής του QRS μαρκάρεται το δείγμα εκείνο του τμήματος στο οποίο εμφανίζονται ταυτόχρονα η μέγιστη τιμή του ΗΚΓ και ο μηδενισμός της πρώτης παραγώγου του.

Η ανίχνευση του QRS επιτυγχάνεται χρησιμοποιώντας ψηφιακά φίλτρα με διάφορους άλλους τρόπους. Το σήμα *z(n)* παράγεται περνώντας το ΗΚΓ από δύο διαφορετικά ζωνοπερατά φίλτρα και στη συνέχεια πολλαπλασιάζοντας τις εξόδους τους.

Η διαδικασία βασίζεται στην υπόθεση ότι ένα σύμπλεγμα QRS χαρακτηρίζεται από στοιχεία συχνότητας που εμφανίζονται ταυτόχρονα και στις δύο εξόδους των ζωνοπερατών φίλτρων. Ο πολλαπλασιασμός τους (λογικό ΚΑΙ) εξασφαλίζει την ύπαρξη και των δύο. Αυτό σημαίνει ότι μόνο όταν και οι δύο έξοδοι έχουν υψηλή τιμή έχει και το σήμα υψηλή τιμή και υποδεικνύει την ύπαρξη QRS. Ως θέση του μεγίστου θεωρείται η θέση του κύματος R.

Σε άλλες εφαρμογές παρουσιάζεται η χρήση επαναληπτικών και μη επαναληπτικών φίλτρων που εξάγουν την κεντρική τιμή, όπως:

$$y(n) = median[y(n-m),...,y(n-1),$$
$$x(n),x(n+1),...,x(n+m)]$$
$$\kappa\alpha\iota$$
$$y(n) = median[x(n-m),...,x(n-1),$$
$$x(n),x(n+1),...,x(n+m)]$$

Πολλές φορές τα φίλτρα αυτά χρησιμοποιούνται σε συνδυασμό με ένα φίλτρο εξομάλυνσης (smoothing) για την δημιουργία ζωνοπερατού φίλτρου.

3.3.2 Ανίχνευση QRS βασισμένη στην κυματομορφή

3.3.2.1 Μετασχηματισμός κυματομορφής και ανίχνευση ιδιομορφιών.

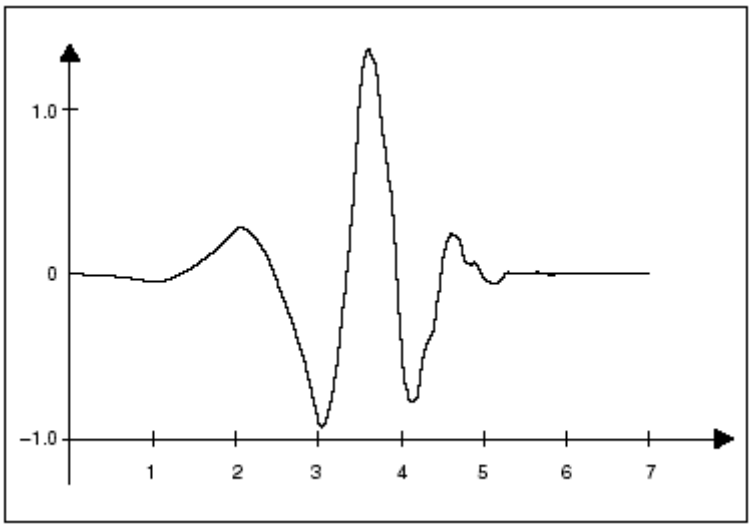
Ο μετασχηματισμός κυματομορφής (WT) μιας συνάρτησης είναι ένας ακέραιος μετασχηματισμός που ορίζεται από τη σχέση

$$Wf(a,b) = \int_{-\infty}^{+\infty} f(t)\psi^{*}{}_{a,b}(t)dt$$

όπου $\psi^*(t)$ η συζυγής της $\psi(t)$. Ο μετασχηματισμός αποδίδει μια απεικόνιση χρόνου-κλίμακας παρόμοια με την απεικόνιση χρόνου-συχνότητας του μετασχηματισμού Fourier σύντομου χρόνου (STFT). Σε αντίθεση με τον STFT ο WT χρησιμοποιεί ένα ζεύγος συναρτήσεων που επιτρέπουν διαφορετική ανάλυση τιμών χρόνου και συχνότητας για διαφορετικές ζώνες συχνοτήτων. Το ζεύγος συναρτήσεων $\psi_{a,b}$ προκύπτει από την αρχική κυματομορφή $\psi(t)$ μέσω του τύπου

$$\psi_{a,b}(t) = \frac{1}{\sqrt{2}} \cdot \psi\left(\frac{t-b}{a}\right)$$

όπου a και b οι παράμετροι διαστολής και μετάφρασης αντίστοιχα. Η παράμετρος a είναι η αντίστοιχη της παραμέτρου συχνότητας του STFT. Η αρχική κυματομορφή είναι ένα μικρό κύμα με μηδενική μέση τιμή. Ενα παράδειγμα φαίνεται στην εικόνα 52.



4. Example of a wavelet function (Daubechies-4 wavelet).

Εικόνα 52 Παράδειγμα συνάρτησης κυματομορφής

Ο διακριτός μετασχηματισμός κυματομορφής (DWT) προκύπτει από διακριτές παραμέτρους *α* και *b*. Για παράδειγμα $\alpha=2^j$ και $b=n\cdot 2^j$, όπου *n* και *j* ακέραιοι. Αυτή η επιλογή των *α* και *b* οδηγεί στον δυαδικό DWT

$$Wf(2^j,b)=\int_{-\infty}^{+\infty} f(t)\cdot\psi^*{}_{2^j,b}(t)dt$$

$$\mu\varepsilon$$

$$\psi_{2^j,b}(t)=\frac{1}{2^{j/2}}\cdot\psi\left(\frac{t-b}{2^j}\right)=\frac{1}{2^{j/2}}\cdot\psi\left(\frac{t}{2^j}-n\right)$$

$$\kappa\alpha\iota\ \ j,n\in Z$$

Παρόλο που ορίζεται σαν ολοκληρωτικός μετασχηματισμός, συνήθως εφαρμόζεται χρησιμοποιώντας δυαδικό φίλτρο με χαρακτηριστικά που προκύπτουν απευθείας από τη συνάρτηση της κυματομορφής. Το σήμα εισόδου στο φίλτρο αυτό είναι η δειγματοληψία του ΗΚΓ.

Σχεδόν όλες οι μέθοδοι ανίχνευσης κορυφών με βάση την κυματομορφή βασίζονται στις προσεγγίσεις των Mallat και Hwang για ανίχνευση και κατηγοριοποίηση με χρήση του τοπικού μεγίστου της κυματομορφής. Σε αυτές αναζητείται η αντιστοιχία μεταξύ των ιδιομορφιών μιας συνάρτησης *f(t)* και των τοπικών μεγίστων της

μετασχηματισμένης κυματομορφής *Wf(α,t)*. Φαίνεται ότι οι ιδιομορφίες αντιστοιχούν σε ζεύγη μέγιστων συντελεστών σε διάφορες κλίμακες. Η κατάταξη των κορυφών επιτυγχάνεται με τον υπολογισμό του βαθμού της ιδιομορφίας. Ένα παράδειγμα αποτελεί η τοπική ιδιομορφία Lipschitz *α*, που εκτιμάται από την πτώση του πλάτους της κυματομορφής μέσω του τύπου

$$a_j = \log_2 \left| Wf(2^{j+1}, n^{j+1}) \right| - \log_2 \left| Wf(2^j, n^j) \right|$$

και

$$a = \frac{a_1 + a_2}{2}$$

3.3.2.2 Προσεγγίσεις βασισμένες στην ανίχνευση ιδιομορφίας

Στον αλγόριθμο που προτείνεται από τους Mallat και Hwang οι αιχμές R του συμπλέγματος εντοπίζονται με την ταυτόχρονη ύπαρξη μέγιστων συντελεστών σε σχετικές κλίμακες του WT. Για να θεωρηθεί μια αιχμή ως έγκυρη αιχμή R, η εκτιμώμενη ιδιομορφία Lipschitz θα πρέπει να είναι μεγαλύτερη του μηδέν ($\alpha>0$). Εκτός από την προϋπόθεση για τον δείκτη Lipschitz, εφαρμόζονται και περαιτέρω κριτήρια απόφασης, όπως προϋποθέσεις για το πρόσημο και τη χρονική στιγμή εμφάνισης της κορυφής σε διαφορετικές κλίμακες.

Σε άλλους αλγόριθμους βασισμένους σε τοπικά μέγιστα γίνεται ανίχνευση χαρακτηριστικών δειγμάτων μέσω της σύγκρισης του WT επιλεγμένης κλίμακας με συγκεκριμένα κατώφλια πλάτους. Σε κάποιες εφαρμογές, το σήμα του ΗΚΓ χωρίζεται σε τμήματα συγκεκριμένου μεγέθους. Η ανίχνευση της αιχμής R του QRS πραγματοποιείται μέσω σύγκρισης των μέγιστων συντελεστών με κατώφλι που υπολογίζεται για το εκάστοτε τμήμα.

Υπάρχουν και εφαρμογές όπου η κυματική αναπαράσταση των μηδενισμών του σήματος χρησιμοποιείται για αναγνώριση προτύπων. Αυτή η μέθοδος αποτελείται από ένα στάδιο εκμάθησης και ένα στάδιο αναγνώρισης. Στο στάδιο εκμάθησης, δημιουργείται ένα σύνολο γενικευμένων χαρακτηριστικών διανυσμάτων από ένα σύνολο προτύπων, μέσω της αναπαράστασης των μηδενισμών τους. Στο στάδιο της αναγνώρισης, χαρακτηριστικά διανύσματα υπολογίζονται για τμήματα συγκεκριμένου μεγέθους του ΗΚΓ και συγκρίνονται με τα γενικευμένα διανύσματα. Αν ένα συγκεκριμένο ποσοστό ομοιομορφίας ξεπερνά το κατώφλι, ανιχνεύεται μια αιχμή R.

3.3.3 Προσεγγίσεις με νευρωνικά δίκτυα

3.3.3.1 Νευρωνικά δίκτυα

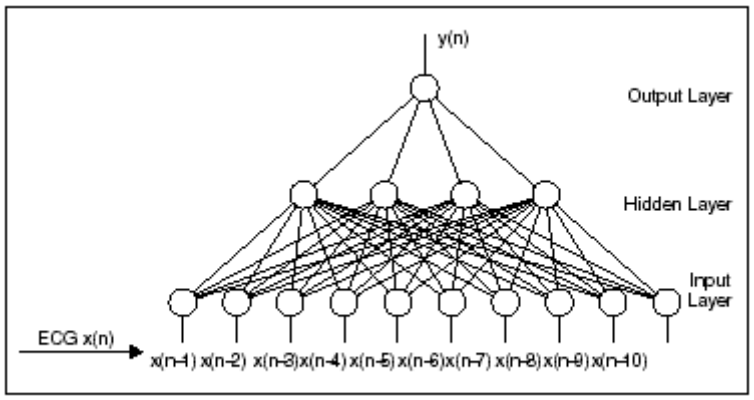
Τα τεχνητά νευρωνικά δίκτυα έχουν ευρύ φάσμα εφαρμογών στη μη γραμμική επεξεργασία σήματος, στην κατάταξη και στην οπτικοποίηση. Σε πολλές εφαρμογές έχει αποδειχθεί ότι η απόδοσή τους είναι σαφώς ανώτερη των κλασσικών γραμμικών προσεγγίσεων.

Στην επεξεργασία του σήματος του ΗΚΓ χρησιμοποιούνται κυρίως τα δίκτυα multiplayer percepton (MLP), radial basis function (RBF) και learning vector quantization (LVQ). Το δίκτυο MLP αποτελείται από αρκετά στρώματα διασυνδεδεμένων νευρώνων, όπου ο κάθε νευρώνας αντιπροσωπεύει μια συνάρτηση επεξεργασίας

$$y = f\left(w_0 + \sum_{i=1}^{N} w_i x_i \right)$$

με w_i το συντελεστή βάρους που αντιστοιχεί στην είσοδο x_i και *f()* μια γραμμική ή μη γραμμική συνάρτηση. Στην περίπτωση μη γραμμικής συνάρτησης, η *f()* ορίζεται συχνά ως η λογιστική συνάρτηση $f(n) = 1/(1+e^{-u})$ ή $f(n) = \tanh(u)$. Τα δίκτυα RBF είναι μια υλοποίηση της συνάρτησης

$y(n) = \sum_{i=1}^{N} w_i \exp\left(-\frac{\vec{x}(n) - \vec{c}_i}{\sigma_i} \right)$ όπου ***x(n)*** είναι ένα διάνυσμα δεδομένων εισόδου. Ο αριθμός N των νευρώνων, οι συντελεστές w_i, τα κεντρικά διανύσματα $\boldsymbol{c}_i$ και οι τυπικές αποκλίσεις σ_i είναι όλες παράμετροι του συστήματος. Η εκθετική συνάρτηση μπορεί να αντικατασταθεί από άλλες συναρτήσεις, όπως κυματομορφές. Τα δίκτυα RBF είναι στενά συνδεδεμένα με τις μεθόδους της ασαφούς λογικής (fuzzy logic). Το πλεονέκτημα των δικτύων RBF έναντι των MLP είναι ότι, όπως και στην ασαφή λογική, υπάρχει η πιθανότητα ερμηνείας των παραμέτρων. Αυτό καθιστά τα αποτελέσματα πιο προβλέψιμα και, συνεπώς, πιο αξιόπιστα.



6. Multilayer perceptron.

Εικόνα 53 Δομή νευρωνικού δικτύου

Το δίκτυο LVQ αποτελείται από ένα στρώμα εισόδου (input), ένα ανταγωνιστικό στρώμα (competitive) και ένα γραμμικό στρώμα (linear). Το ανταγωνιστικό στρώμα μαθαίνει αυτόματα να κατηγοριοποιεί τα διανύσματα εισόδου σε υποκατηγορίες, με το μέγιστο αριθμό τους N να ισούται με τον αριθμό των νευρώνων του στρώματος. Εδώ, η κατάταξη επιτυγχάνεται με βάση την Ευκλείδεια απόσταση μεταξύ του διανύσματος εισόδου και του διανύσματος βάρους καθενός από τους νευρώνες. Τελικά, το γραμμικό στρώμα συνδυάζει τις υποκατηγορίες του πρώτου στρώματος στις τελικές κατηγορίες που έχουν οριστεί από το χρήστη.

Για την επίτευξη της εργασίας που απαιτεί η εκάστοτε εφαρμογή (για παράδειγμα κατάταξη ή προσέγγιση), οι παράμετροι του συστήματος πρέπει να εκπαιδεύονται. Τα δίκτυα MLP και BRF εκπαιδεύονται από επιβλεπόμενους αλγόριθμους εκμάθησης, ενώ τα LVQ δίκτυα προσαρμόζονται με μη επιβλεπόμενο τρόπο. Σχετικοί αλγόριθμοι εκπαίδευσης περιγράφονται στη βιβλιογραφία

3.3.4 Παρόμοιες Προσεγγίσεις

3.3.4.1 Προσαρμοστικά φίλτρα

Η εφαρμογή της χρήσης προσαρμοστικών φίλτρων για την ανίχνευση QRS έχει ερευνηθεί. Παρόμοιος με τη μη γραμμική περίπτωση, ο στόχος του φίλτρου είναι να κερδίσει μια εκτίμηση $\hat{x}(n) = \sum_{i=1}^{P} a_i(n)x(n-i)$ με τους χρονομεταβλητούς

συντελεστές $a_i(n)$, $i=1,2,...,P$, προσαρμοστικά ρυθμισμένους ανάλογα με τις μεταβαλλόμενες στατιστικές σημάτων. Από τη βιβλιογραφία, διάφοροι κανόνες προσαρμογής για τους συντελεστές είναι γνωστοί, π.χ. ο αλγόριθμος των ελάχιστων μέσων τετραγώνων (LMS): $a(n+1) = a(n) + \lambda e(n)y(n)$, όπου το $a(n) = [a_1(n), a_2(n), ..., a_p(n)]^T$ δείχνει το διάνυσμα συντελεστή στο χρόνο n, το λ είναι η παράμετρος μεγέθους βημάτων, το $e(n) = y(n) - \hat{y}(n)$ δείχνει την πρόβλεψη λάθους και το $x(n) = [x(n-1), x(n-2),...,x(n-P)]^T$ είναι το διάνυσμα των χρονοκαθυστερημένων δειγμάτων σημάτων ΗΚΓ.

Έχουν προταθεί τα βασικά χαρακτηριστικά για τα προσαρμοστικά φίλτρα, καθώς και η χρήση των διαφορών μεταξύ των διανυσμάτων συντελεστών *a* στο χρόνο *n* και στο χρόνο *n-1*, π.χ.

$D(n) = \sum_{i=1}^{P}(|\,a_i(n) - a_i(n-1)|)^2$ και ενός συνδυασμού της διαφοράς μεταξύ των σύντομων χρονικών ενεργειών του υπολοίπου λάθους δύο διαδοχικών τμημάτων $D(n) = \sum_{i=n}^{n+m} e^2(i) - \sum_{i=n-m}^{n} e^2(i)$. Έχει αποδειχθεί ότι σε μια δειγματοληψία με συχνότητα $f_T = 500\,Hz$ δύο φίλτρα είναι ικανοποιητικά για μια καλή απόδοση πρόβλεψης.

3.3.4.2 Τα μοντέλα Hidden Markov (HHMs)

Τα HHMs μοντέλα διαμορφώνουν την ακολουθία στοιχείων με χρήση μιας συνάρτησης πιθανότητας σύμφωνα με την κατάσταση της Markov αλυσίδας. Με τη βοήθεια της Markov αλυσίδας τα δομικά χαρακτηριστικά της διαδικασίας συντηρούνται, ενώ οι παράμετροι της συνάρτησης πυκνότητας πιθανότητας μετράνε για τις ποικίλες στατιστικές ιδιότητες των παρατηρηθέντων στοιχείων. Ο στόχος του αλγορίθμου είναι να συμπεράνει την ελλοχεύουσα κατάσταση της ακολουθίας από το παρατηρηθέν σήμα. Στην περίπτωση των σημάτων ΗΚΓ, πιθανές καταστάσεις είναι το P-έπαρμα, το QRS και το T-έπαρμα. Το πλεονέκτημα αυτής της μεθόδου ανίχνευσης είναι ότι όχι μόνο το σύμπλεγμα QRS καθορίζεται, αλλά και τα P- και T-επάρματα. Τα προβλήματα της μεθόδου περιλαμβάνουν μια απαραίτητη και μη-αυτοματοποιημένη, αλλά χειρωνακτική κατάτμηση του αρχείου πριν από την ανάλυσή του και τη σημαντικού βαθμού υπολογιστική πολυπλοκότητα ακόμα και όταν εφαρμόζεται ο υπολογιστικά αποδοτικός αλγόριθμος Viterby.

3.3.4.3 Μαθηματική μορφολογία

Η μαθηματική μορφολογία προέρχεται από την επεξεργασία εικόνας και προτάθηκε για την αύξηση σημάτων ΗΚΓ και την επιτυχή αφαίρεση ωστικού θορύβου από το ΗΚΓ. Η μαθηματική μορφολογία είναι βασισμένη στους όρους *διάβρωση (erosion)* και *διαστολή (dilation)*. Έστω ότι f:F $\longrightarrow I$ και k:K$\longrightarrow I$ δηλώνουν διακριτές συναρτήσεις, όπου τα σύνολα F και K δίνονται από F = {0,1,...,N-1} και K = {0,1,…,M-1}. Το I είναι το σύνολο των ακεραίων αριθμών. Η διάβρωση της συνάρτησης f από τη συνάρτηση k ορίζεται ως: $(\mathrm{f} \ominus k) = \min_{n=0,1,..,M-1} f(m+n) - k(n)$ για $N >$ M και m = 0,..,N-M και το k αναφέρεται επίσης ως δομικό στοιχείο. Οι τιμές των $(f \ominus k)$ είναι πάντα λιγότερες από αυτές της f.

Η διαστολή της συνάρτησης f από την συνάρτηση k ορίζεται ως: $(f \oplus k)(m) = \max_{n=0,1,..,M-1} f(n) + k(m-n)$ για $N >$ M και m = M-1,...,N-1. Οι τιμές των (f $\oplus k$) είναι πάντα μεγαλύτερες από αυτές της f.

Η διάβρωση και η διαστολή συνδυάζονται για πρόσθετες διαδικασίες. Το opening («άνοιγμα»), δηλωμένο ως ∘, ορίζεται ως η διάβρωση που ακολουθείται από τη διαστολή. Το closing («κλείσιμο»), δηλωμένο ως ·, ορίζεται ως η διαστολή που ακολουθείται από μια διάβρωση. Και οι δύο αυτοί φορείς χειρίζονται τα σήματα με έναν συγκρίσιμο τρόπο. Αυτός είναι ο εξής: το να «ανοίγεις» μια συνάρτηση f με ένα επίπεδο δομικό στοιχείο k θα αφαιρέσει όλες τις κορυφές. Το να «κλείσεις» την ακολουθία με το ίδιο δομικό στοιχείο θα αφαιρέσει όλα τα κοίλα (αρνητικές κορυφές). Η γενίκευση ενός χαρακτηριστικού σήματος για QRS συμπλέγματα υλοποιείται ως εξής: $\mathrm{z} = \tilde{x} - [(\tilde{x} \circ B) \cdot B]$ όπου B είναι ένα κορυφώδες δομικό στοιχείο. Μια QRS κορυφή, τελικά, βρίσκεται συγκρίνοντας το χαρακτηριστικό σήμα με ένα προσαρμοστικό κατώφλι.

3.3.4.4 Αντιστοιχισμένα φίλτρα (Matched filters)

Πέρα από τα αντιστοιχισμένα φίλτρα που βασίζονται σε νευρωνικά δίκτυα, υπάρχουν και προσεγγίσεις γραμμικών αντιστοιχισμένων φίλτρων. Μετά από κάποια αναλογικά προεπεξεργαστικά βήματα, όπως ο αυτόματος έλεγχος κέρδους, το σήμα ΗΚΓ ψηφιοποιείται και έπειτα περνά μέσα από ένα βαθυπερατό φίλτρο με συχνότητα αποκοπής στα 50 Hz και μέσα από ένα ζωνοπερατό φίλτρο με συχνότητα αποκοπής

στα 15Hz και τα 40Hz. Αυτό το στάδιο ψηφιακών φίλτρων ακολουθείται από ένα αντιστοιχισμένο φίλτρο για περαιτέρω βελτίωση του σηματοθορυβικού λόγου (SNR). Το αντιστοιχισμένο φιλτράρισμα ακολουθείται από: y(n) = $\sum_{i=0}^{N-1} h(i)x(n-i)$, όπου η κρουστική απόκριση h(n) είναι το φέρον (βασισμένο στο χρόνο) της κυματομορφής που θα ανιχνευθεί. Η κρουστική απόκριση ενός αντιστοιχισμένου φίλτρου h(n) λαμβάνεται από τους πρώτους καρδιακούς κύκλους των τρεχουσών μετρήσεων. Για περαιτέρω βελτίωση της χρονικής ακρίβειας, η έξοδος του φίλτρου παρεμβάλλεται μέχρι τέσσερις φορές στην αρχική συχνότητα δειγματοληψίας. Η τελική απόφαση για την QRS κορυφή λαμβάνεται συγκρίνοντας το φιλτραρισμένο σήμα με ένα σταθερό κατώφλι. Μάλιστα σε αυτήν την εφαρμογή αναφέρεται πως το αντιστοιχισμένο φίλτρο βελτιώνει και τη χρονική ακρίβεια του ανιχνευμένου R-επάρματος.

Μια παρόμοια προσέγγιση προτείνει αντί για υπολογισμό της συνάρτησης ετεροσυσχέτισης (cross-correlation) ανάμεσα στο φέρον και στο σήμα όπως και στην προαναφερθείσα εξίσωση: y(n) = $\sum_{i=0}^{N-1} h(i)x(n-i)$, ο αλγόριθμος να ψάχνει για το ελάχιστο των μέσων διαφορών μεγέθους (AMCD): AMCD = $\sum_{i-1}^{N} |x(n-i) - h(i)|$ όπου x(n) είναι το ΗΚΓ σήμα και h(i) είναι το φέρον (βασισμένο στο χρόνο). Αυτός ο αλγόριθμος δε χρειάζεται πολλαπλασιασμούς και είναι συνεπώς υπολογιστικά ανέξοδος.

3.3.4.5 Γενετικοί Αλγόριθμοι

Γενετικοί αλγόριθμοι εφαρμόζονται σε ένα συνδυασμένο σχεδιασμό βέλτιστων πολυωνυμικών φίλτρων για την προεπεξεργασία των ΗΚΓ και των παραμέτρων του σταδίου απόφασης.

Τα πολυωνυμικά φίλτρα ορίζονται ως εξής: $y_n = \sum_{k_1}^{M} \sum_{k_2}^{M} \dots \sum_{k_N}^{M} a_{k_1 k_2 \dots k_N} x_{n-d_1}^{k_1} x_{n-d_2}^{k_2} \dots x_{n-d_N}^{k_N}$, όπου τα d_j είναι οι καθυστερήσεις ως προς το χρόνο n. Τρεις διαφορετικές περιπτώσεις πολυωνυμικού φίλτρου ερευνώνται:

- Σχεδόν γραμμικά φίλτρα με διαδοχικά δείγματα (M=1 και N=10)
- Σχεδόν γραμμικά φίλτρα με επιλεγμένα δείγματα (M=1 και N=5)
- Τετραγωνικά φίλτρα με επιλεγμένα δείγματα (M=2 και N=3)

Το στάδιο της απόφασης αποτελείται από ένα προσαρμοστικό κατώφλι το οποίο συγκρίνεται με το φιλτραρισμένο ΗΚΓ σήμα. Οι παράμετροι προσαρμογής του

κατωφλίου βελτιστοποιούνται σε συνδυασμό με το πολυωνυμικό φίλτρο μέσω της βελτιστοποίησης του γενετικού αλγορίθμου.

3.3.4.6 Ανίχνευση QRS βασισμένη στο μετασχηματισμό Hilbert

Προτείνεται ακόμα η χρήση του μετασχηματισμού Hilbert για την ανίχνευση QRS. Ο μετασχηματισμός Hilbert ενός πραγματικού σήματος x ορίζεται ως: $x_H(t) = \mathcal{H}\{x\} = \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{x(\tau)}{t-\tau}d\tau = x_H(t) = x(t) * \frac{1}{\pi t}$ και μπορεί να υπολογισθεί στο πεδίο συχνοτήτων ως: $x_H(j\omega) = X(j\omega)\cdot[-j\cdot sgn(\omega)] = X(j\omega)\cdot H(j\omega)$ όπου η συνάρτηση μεταφοράς του μετασχηματισμού Hilbert H(jω) δίνεται: H(jω) $\begin{cases} -j, 0 \le \omega < \pi \\ +j, -\pi \le \omega < 0 \end{cases}$.

Με χρήση του ταχέος μετασχηματισμού Fourier (FFT), ο μετασχηματισμός Hilbert μπορεί εύκολα να υπολογισθεί. Ο ιδανικός μετασχηματισμός Hilbert προσεγγίζεται από ένα FIR φίλτρο με ζώνη περιορισμού (2N+1) και με κρουστική απόκριση h(n).

Ο μετασχηματισμός Hilbert $x_H(n)$ του ΗΚΓ σήματος x(n) χρησιμοποιείται για τον υπολογισμό του πλάτους του σήματος, ο οποίος δίνεται για σήματα περιορισμένου εύρους από τον τύπο: $x_e(n) = \sqrt{x^2(n) + x_H^2(n)}$. Μια λιγότερο «ακριβή» υπολογιστικά προσέγγιση μπορεί να δοθεί από: $x_e(n) \approx |x(n)|+|x_H(n)|$.

Για να αφαιρεθούν κυματισμοί από το παραπάνω σήμα και να αποφευχθούν ασάφειες στο επίπεδο ανίχνευσης κορυφής, φιλτράρεται με βαθυπερατό φίλτρο. Επιπλέον, προτείνεται ένα σχέδιο προσαρμοστικής κυματομορφής για την αφαίρεση ΗΚΓ στοιχείων με χαμηλή συχνότητα.

3.3.4.7 Μετασχηματισμοί μήκους και ενέργειας

Ερευνάται επιπλέον η χρήση των μετασχηματισμών μήκους και ενέργειας στην ανίχνευση QRS. Οι μετασχηματισμοί αυτοί χρησιμοποιούνται για τα πολυδιαυλικά σήματα ΗΚΓ, αλλά και για την ανάλυση μονοδιαυλικού σήματος ΗΚΓ. Δίνονται από τους τύπους: L(n,q,i) = $\sum_{k=i}^{i+q-1}\sqrt{\sum_{j=1}^{n}(\Delta x_{j,k})^2}$, ο μετασχηματισμός μήκους και E(n,q,i) = $\sum_{k=i}^{i+q-1}\sum_{j=1}^{n}(\Delta x_{j,k})^2$, ο μετασχηματισμός ενέργειας, όπου n ο αριθμός των ΗΚΓ καναλιών, i ο χρονικός δείκτης, q το μήκος του παραθύρου και $\Delta x_{j,k} = x_{j,k} - x_{j,k-1}$. Αυτοί οι τύποι βασίζονται στην υπόθεση ότι οι παράγωγοι των ΗΚΓ καναλιών

μπορούν να θεωρηθούν ως στοιχεία ενός διανύσματος. Το μήκος του διανύσματος καθορίζεται από την τετραγωνική ρίζα του δεύτερου ποσού στην εξίσωση του μετασχηματισμού μήκους.

Ο μετασχηματισμός μήκους αντιπροσωπεύει μια χρονική πορεία διανύσματος μήκους και ο μετασχηματισμός ενέργειας μπορεί να ερμηνευθεί ως μια βραχυπρόθεσμη εκτίμηση ενέργειας ενός διανύσματος. Μάλιστα, ο μετασχηματισμός μήκους λειτουργεί ιδιαιτέρως καλά σε περιπτώσεις μικρών συμπλεγμάτων QRS.

3.3.4.8 Συντακτικές Μέθοδοι

Το σήμα που πρόκειται να αναλυθεί με συντακτική μέθοδο θεωρείται μια αλληλουχία από πρότυπα γλωσσολογικά σχήματα, όπως strings. Χρησιμοποιώντας μια γραμματική, αυτή η απεικόνιση των strings αναλύεται γραμματικά κωδικοποιώντας μια πατέντα αναζήτησης. Έτσι, ένας συντακτικός αλγόριθμος χρειάζεται τον ορισμό των πρότυπων γλωσσολογικών σχημάτων, μια κατάλληλη γλωσσολογική απεικόνιση (αλφάβητο) των πρότυπων αυτών σχημάτων και μια διατύπωση της πρότυπης γραμματικής.

Στην επεξεργασία ΗΚΓ σημάτων, το σήμα χωρίζεται σε μικρά τμήματα μεταβλητού ή σταθερού μήκους. Κάθε τμήμα, έπειτα, απεικονίζεται με μια αρχική και μια κωδικοποιημένη χρησιμοποίηση του προκαθορισμένου αλφάβητου. Λόγω της υπολογιστικής αποδοτικότητας, οι περισσότεροι αλγόριθμοι χρησιμοποιούν τμήματα γραμμών ως πρότυπα για την απεικόνιση των σημάτων. Μάλιστα, αυτό το σύνολο των πρότυπων γραμμών εκτείνεται έτσι ώστε να συμπεριλαμβάνει και κορυφές, παραβολικές καμπύλες και επιπρόσθετες ιδιότητες.

Στον παρακάτω πίνακα απεικονίζονται τα αλφάβητα, τα πρότυπα γλωσσολογικά σχήματα και τα σύμβολα κάποιων συντακτικών αλγορίθμων.

Table 1. Sets Used in Syntactic Approaches for ECG Event Detection.				
	Set (alphabet)	Primitives	Symbols	Reference
(1)	$\Sigma = \{a,b,c\}$	square of the first derivative y_1^2	a,b,c depend on amplitude and duration of peaks of y_1^2	[9]
(2)	$\Sigma = \{(a,b) \mid a \in \{/,\backslash,0\}, b \in \{+,-,*\}\}$	line segments	a: slope of the line segment, i.e. positive slope (/), negative slope (\) or zero slope (0). b: start point of the line segment, i.e. above (+), below (–) or on the baseline (*)	[46]
(3)	$\Sigma = \{h,s,s_n,i,i_n,l,l_n\}$	line segments	horizontal (h), small (s), negative small (s_n), intermediate (i), negative intermediate (i_n), large (l) and negative large (l_n) slope	[117]
(4)	$\Sigma = \{(l_P,i,n),(s_P,i,n),(s_N,i,n),(l_N,i,n)\}$	line segments	$\{l_P,s_P,s_N,l_N\}$: slope of line segment; i,n: time coordinate and duration of line segment (attribute values)	[87]
(5)	$\Sigma = \{K^+,K^-,E,\Pi\}$	peak, line and parabolic segment	K^+ (positive peak), K^- (negative peak), E (line segment), Π (parabolic segment), and additional attribute values describing the primitives.	[113]

Πίνακας 10 Συντακτικές μέθοδοι ανίχνευσης QRS

3.3.4.9 Ανίχνευση QRS που βασίζεται στην εκτίμηση MAP

Η εκτίμηση του *maximum a posteriori* (MAP) μπορεί να θεωρηθεί μια ειδική περίπτωση στο Bayesian πλαίσιο, που παρέχει μια πολύ γενική βάση για την εκτίμηση παραμέτρων λαμβάνοντας υπόψη προγενέστερη γνώση. Η εκτίμηση της παραμέτρου $\angle\theta_{MAP}$ της MAP, δοθέντος του x, ορίζεται ως: $\angle\theta_{MAP} = \tan^{-1}\max_\theta f_\theta(\theta|x)$, όπου $f_\theta(\theta|x) = \frac{f_x(x|\theta)f_\theta(\theta)}{f_x(x)}$ είναι η a posteriori συνάρτηση πυκνότητας πιθανότητας της θ, δοθέντος του x. Η a priori συνάρτηση πυκνότητας πιθανότητας της θ απεικονίζει τη διαθέσιμη προγενέστερη γνώση.

3.3.4.10 Ανίχνευση QRS με τη μέθοδο zero-crossing

Έπειτα από ένα ζωνοπερατό φιλτράρισμα, μια υψηλής συχνότητας ακολουθία b(n) = k(n) $(-1)^{\prime\prime}$ προστίθεται στο φιλτραρισμένο σήμα $y_1(n)$. Το εύρος της υψηλής συχνότητας ακολουθίας k(n) καθορίζεται από τον τρέχοντα μέσο όρο των σταθερών συντελεστών του ζωνοπερατά φιλτραρισμένου ΗΚΓ $|y_1(n)|$. Εφόσον το εύρος του k(n) είναι μικρότερο από το εύρο του QRS συμπλέγματος, ο αριθμός των αποτελεσμάτων των zero crossings είναι μεγάλος κατά τη διάρκεια των μη-QRS συμπλεγμάτων και μικρός κατά τη διάρκεια του συμπλέγματος QRS. Υπολογίζοντας

έναν τρέχοντα μέσο όρο του αριθμού των zero crossings οδηγείται σε ένα χαρακτηριστικό *z(n)* για QRS κορυφές. Το χαρακτηριστικό αυτό z(n) συγκρίνεται με ένα προσαρμοστικό κατώφλι για την ανίχνευση QRS. Η χρονική τοποθέτηση του R-επάρματος βρίσκεται με μια αναζήτηση μεγίστου σε ένα ζωνοπερατά φιλτραρισμένο σήμα γύρω από ένα ανιχνευμένο υποψήφιο QRS.

Κεφάλαιο 4

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΤΩΝ ΤΕΧΝΙΚΩΝ ΑΝΙΧΝΕΥΣΗΣ QRS ΜΕ ΧΡΗΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ MATLAB

4.1 Προτυποποιημένες βάσεις δεδομένων ΗΚΓ

Διάφορες τυποποιημένες βάσεις δεδομένων ΗΚΓ είναι διαθέσιμες για την αξιολόγηση του λογισμικού των αλγορίθμων ανίχνευσης QRS. Αυτές οι βάσεις δεδομένων περιέχουν ένα μεγάλο αριθμό επιλεγμένων σημάτων, αντιπροσωπευτικών της μεγάλης ποικιλίας των ΗΚΓ όπως και σημάτων που σπάνια παρατηρούνται αλλά κλινικώς είναι σημαντικά.

Οι διαθέσιμες τυποποιημένες βάσεις δεδομένων περιλαμβάνουν:

1. MIT-BIH Βάση Δεδομένων

Η MIT-BIH βάση δεδομένων από το MIT και το Boston's Israel Hospital αποτελείται από δέκα βάσεις δεδομένων, δηλαδή τη βάση δεδομένων αρρυθμίας (arrythmia database), τη βάση δεδομένων δοκιμής πίεσης θορύβου, τη βάση δεδομένων κοιλιακής ταχυρρυθμίας από το καρδιακό κέντρο του Creighton University, τη βάση δεδομένων ST Change, τη βάση δεδομένων της καλοήθους κοιλιακής αρρυθμίας, τη βάση δεδομένων του ενδοσκοπικού ινιδισμού/κυματισμού, τη βάση δεδομένων δοκιμής συμπίεσης του ΗΚΓ, τη βάση δεδομένων υπερκοιλιακής αρρυθμίας, την long-term βάση δεδομένων και τη βάση δεδομένων κανονικού ρυθμού κόλπων. Εκτός από την AHA βάση δεδομένων και την Ευρωπαϊκή ST-T βάση δεδομένων, οι τρεις πρώτες MIT-BIH βάσεις δεδομένων από το ANSI για τη δοκιμή των βαδιστικών συσκευών ΗΚΓ.

Η συχνότερα χρησιμοποιούμενη MIT-BIH βάση δεδομένων είναι η βάση δεδομένων αρρυθμίας. Συνολικά υπάρχουν 116137 συμπλέγματα QRS σε αυτή τη βάση δεδομένων. Σε κάποια αρχεία περιέχονται σαφή R-επάρματα ενώ για κάποια άλλα

αρχεία η ανίχνευση QRS είναι πολύ δύσκολη λόγω των ανώμαλων μορφών και του θορύβου.

2. AHA Βάση Δεδομένων

Η AHA βάση δεδομένων για την αξιολόγηση των κοιλιακών ανιχνευτών αρρυθμίας της American Heart Association περιέχει 155 αρχεία ΗΚΓ. Τα αρχεία αυτά ομαδοποιούνται σε οχτώ ομάδες που απεικονίζουν διαφορετικά επίπεδα εκτοπικής διέγερσης.

3. ANN Arbor Electrogram Libraries

Οι ANN Arbor Electrogram Libraries αποτελούν μια συλλογή από περισσότερα από 800 ενδοκαρδιακά ηλεκτρογραφήματα και επιφανειακά ΗΚΓ. Κάθε καταγραφή αποτελείται από ενδοκαρδιακά μονοπολικά και διπολικά ηλεκτροκαρδιογραφήματα και μια επιφάνεια ΗΚΓ. Αυτή η βάση δεδομένων είναι ιδιαίτερα πολύτιμη για την αξιολόγηση των αλγορίθμων για εμφυτεύσιμες καρδιακές συσκευές.

4. CSE Βάση Δεδομένων

Η CSE Βάση Δεδομένων χρησιμοποιείται συχνά για την αξιολόγηση των διαγνωστικών συσκευών ανάλυσης ΗΚΓ.

5. Άλλες Τυποποιημένες Βάσεις Δεδομένων

Περισσότερες βιβλιοθήκες διαθέσιμες για την αξιολόγηση αλγορίθμων ανίχνευσης και ταξινόμησης είναι η Ευρωπαϊκή ST-T βάση δεδομένων, η QT βάση δεδομένων, η MGH βάση δεδομένων, η IMPROVE βιβλιοθήκη στοιχείων και το σύνολο στοιχείων αναφοράς ΗΚΓ του Physikalisch-Technische Bundesanstalt. Αυτή η βάση δεδομένων επιλέγεται για την αξιολόγηση ΗΚΓ συσκευών που αναλύουν τα επίπεδα ST και T-επάρματα. Η QT βάση δεδομένων σχεδιάστηκε για την αξιολόγηση των αλγορίθμων που ανιχνεύουν τα όρια κυματομορφών στο ΗΚΓ. Η MGH βάση δεδομένων και η IMPROVE βάση δεδομένων είναι πολυδιαυλικές καταγραφές που περιέχουν και πρόσθετα σήματα, όπως πνευμονική αρτηριακή πίεση, κεντρική φλεβική πίεση, CO_2, O_2 και αναπνοή. Το σύνολο των στοιχείων αναφοράς ΗΚΓ του PTB δεν έχει ολοκληρωθεί ακόμα. Στη διπλωματική αυτή, ως προτυποποιημένη βάση δεδομένων χρησιμοποιήθηκε η MIT-BIH Arrythmia Database.

4.2 Προτυποποιημένα σήματα

Από τη βάση δεδομένων MIT-BIH Arrythmia Database μελετήθηκαν τα σήματα 100-107, τα οποία έχουν δειγματοληφθεί με συχνότητα 360Hz και περιέχουν 30000 δείγματα το καθένα.

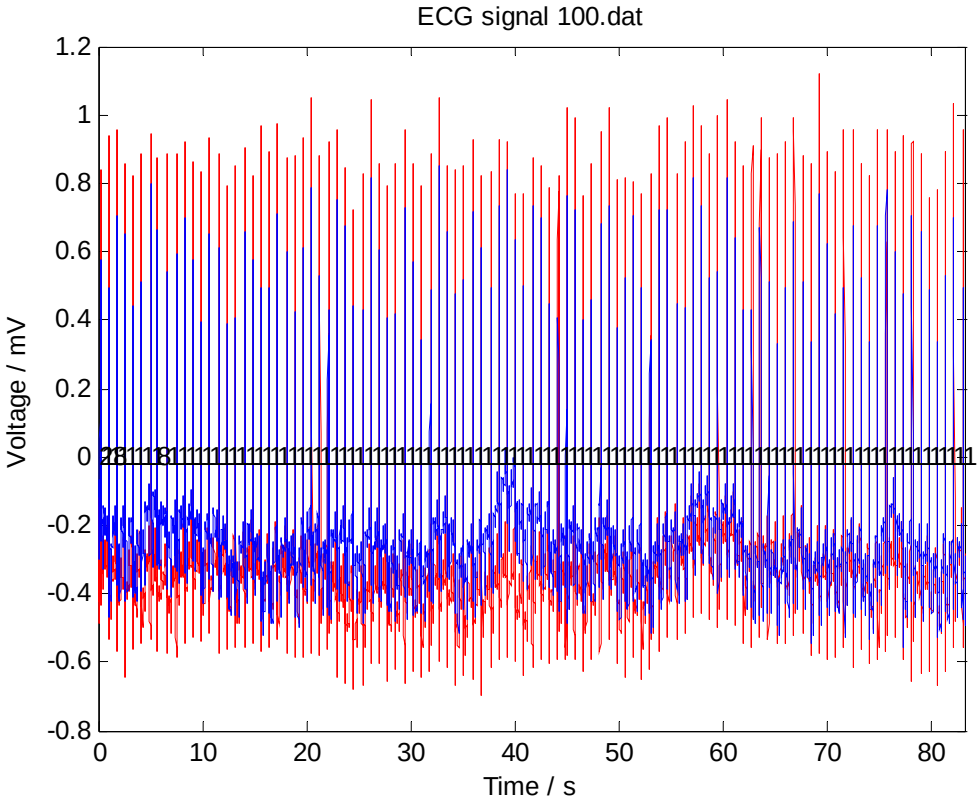
Η μελέτη και η οπτικοποίηση των σημάτων έγινε στο περιβάλλον του MATLAB με τη βοήθεια του αλγορίθμου readMIT που παρουσιάζεται στο Παράρτημα Α. Ο αλγόριθμος αυτός δημιουργήθηκε από τους Robert Tratnig και Klaus Rheinberger. Πρόκειται για ένα πρόγραμμα σε περιβάλλον MATLAB που διαβάζει αρχεία ενός ή δύο διαύλων από τη PhysioBank αποθηκευμένα σε μορφή 212 (format 212). Στη μορφή αυτή κάθε δείγμα έχει δώδεκα bits. Το πρώτο δείγμα προέρχεται από τα δώδεκα λιγότερο σημαντικά bits του πρώτου ζεύγους byte και το δεύτερο δείγμα από τα υπόλοιπα τέσσερα bits του πρώτου ζεύγους byte και από οκτώ bits του επόμενου ζεύγους. Η διαδικασία αυτή επαναλαμβάνεται για κάθε διαδοδικό ζεύγος δειγμάτων. Το πρόγραμμα αυτό απεικονίζει τα αρχεία σε MATLAB figure. Οι ακόλουθες γραμμές

```
HEADERFILE= '107.hea';   % header-file in text format
ATRFILE= '107.atr';     % attributes-file in binary format
DATAFILE='107.dat';     % data-file
```

προσδιορίζουν το αρχείο (record) που θα επεξεργαστεί.

- Σήμα 100

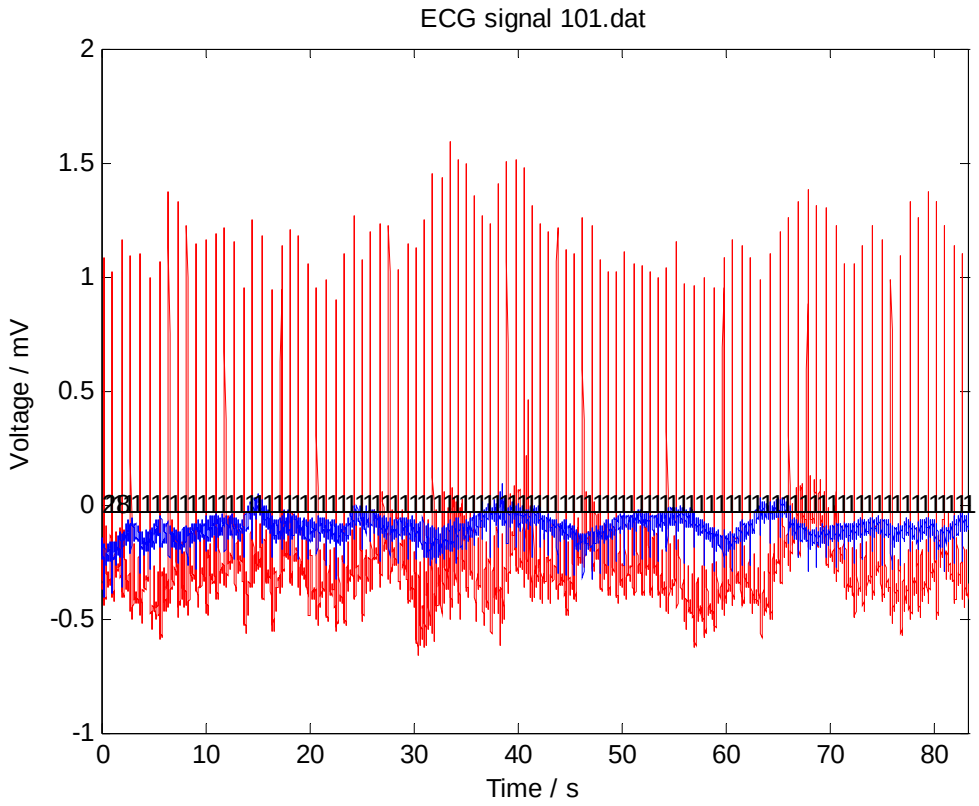
Το σήμα αυτό περιέχει 103 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 54. Πρόκειται για ΗΚΓ άνδρα ηλικίας 69 ετών.



Εικόνα 54 Κυματομορφή του σήματος 100

- Σήμα 101

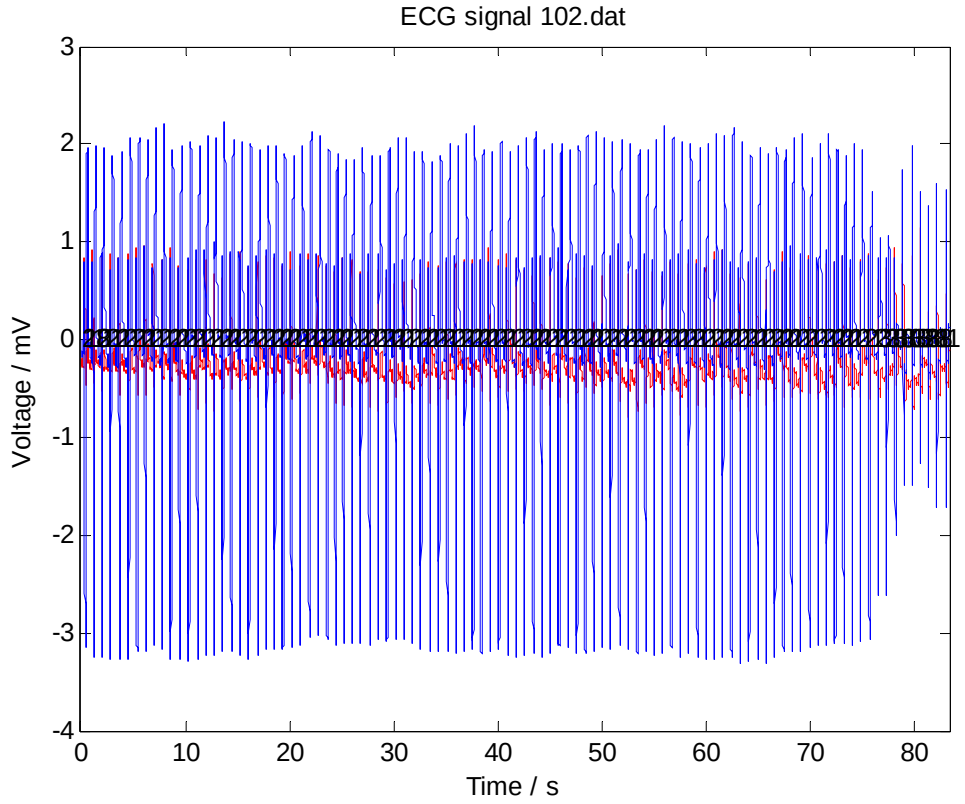
Το σήμα αυτό περιέχει 97 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 55. Πρόκειται για ΗΚΓ γυναίκας ηλικίας 75 ετών.



Εικόνα 55 Κυματομορφή του σήματος 101

- Σήμα 102

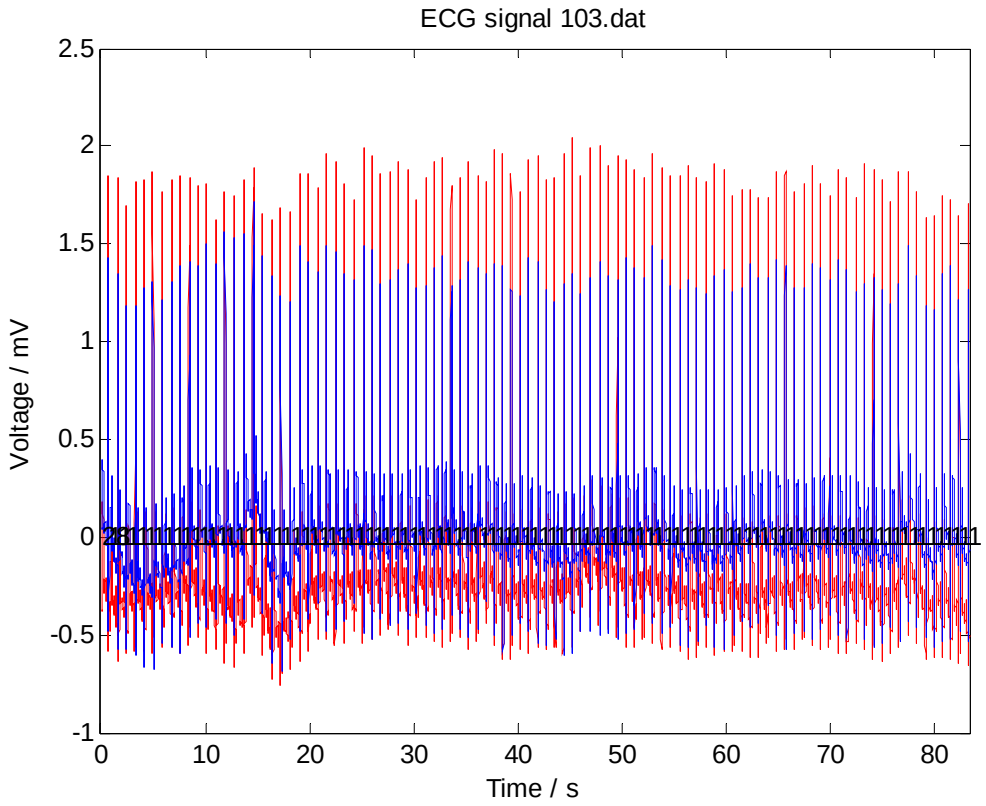
Το σήμα αυτό περιέχει 101 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 56. Πρόκειται για ΗΚΓ γυναίκας ηλικίας 84 ετών.



Εικόνα 56 Κυματομορφή του σήματος 102

- Σήμα 103

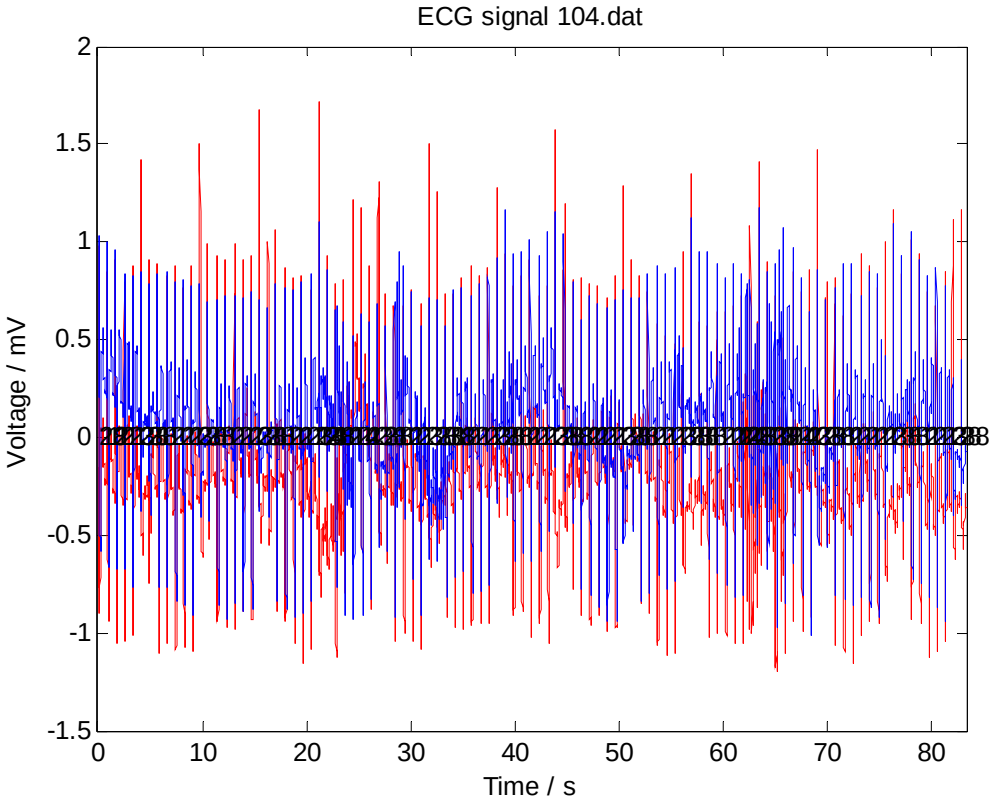
Το σήμα αυτό περιέχει 98 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 57. Πρόκειται για ΗΚΓ άνδρα αγνώστου ηλικίας.



Εικόνα 57 Κυματομορφή του σήματος 103

- Σήμα 104

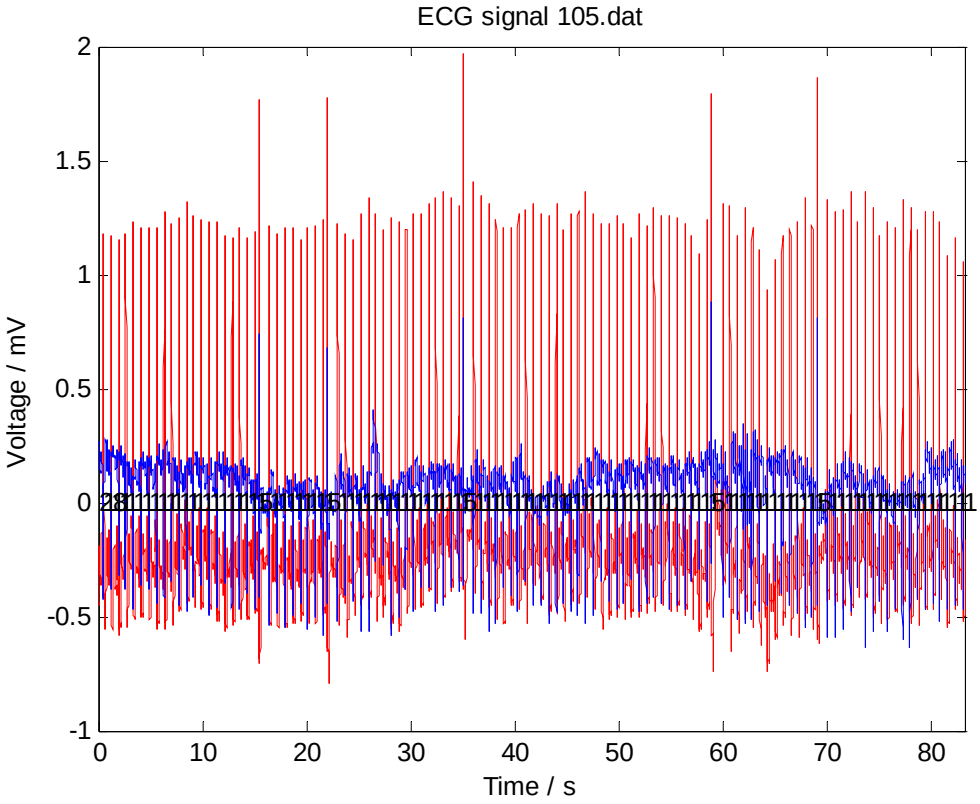
Το σήμα αυτό περιέχει 103 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 58. Πρόκειται για ΗΚΓ γυναίκας ηλικίας 66 ετών.



Εικόνα 58 Κυματομορφή του σήματος 104

- Σήμα 105

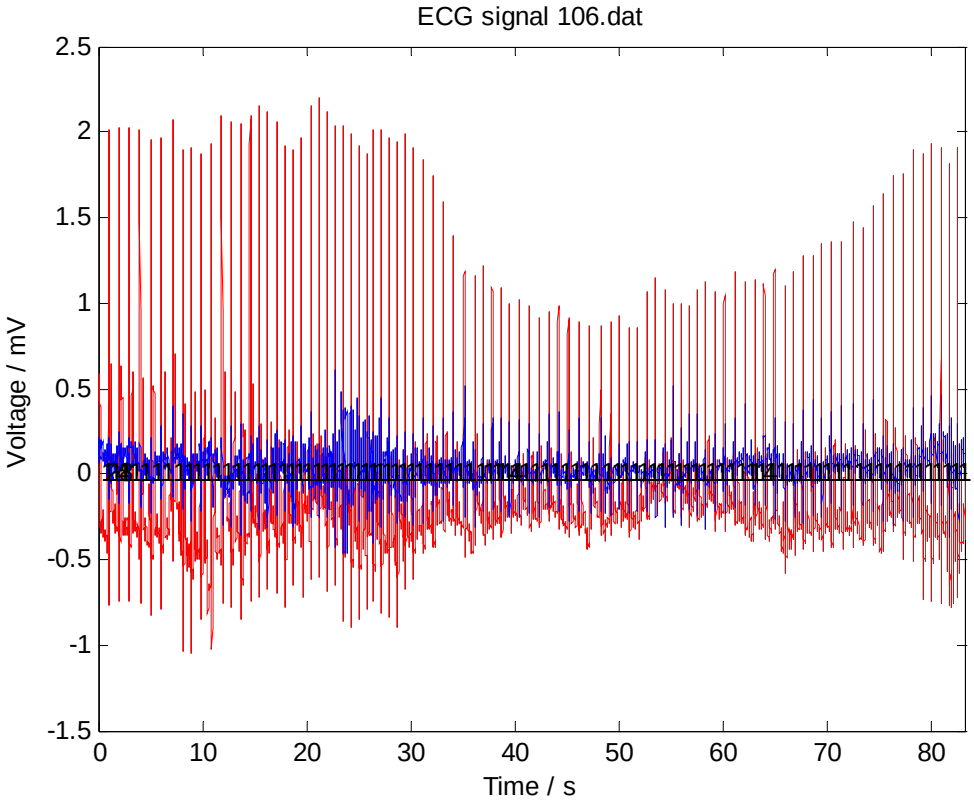
Το σήμα αυτό περιέχει 116 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 59. Πρόκειται για ΗΚΓ γυναίκας ηλικίας 73 ετών.



Εικόνα 59 Κυματομορφή του σήματος 105

- Σήμα 106

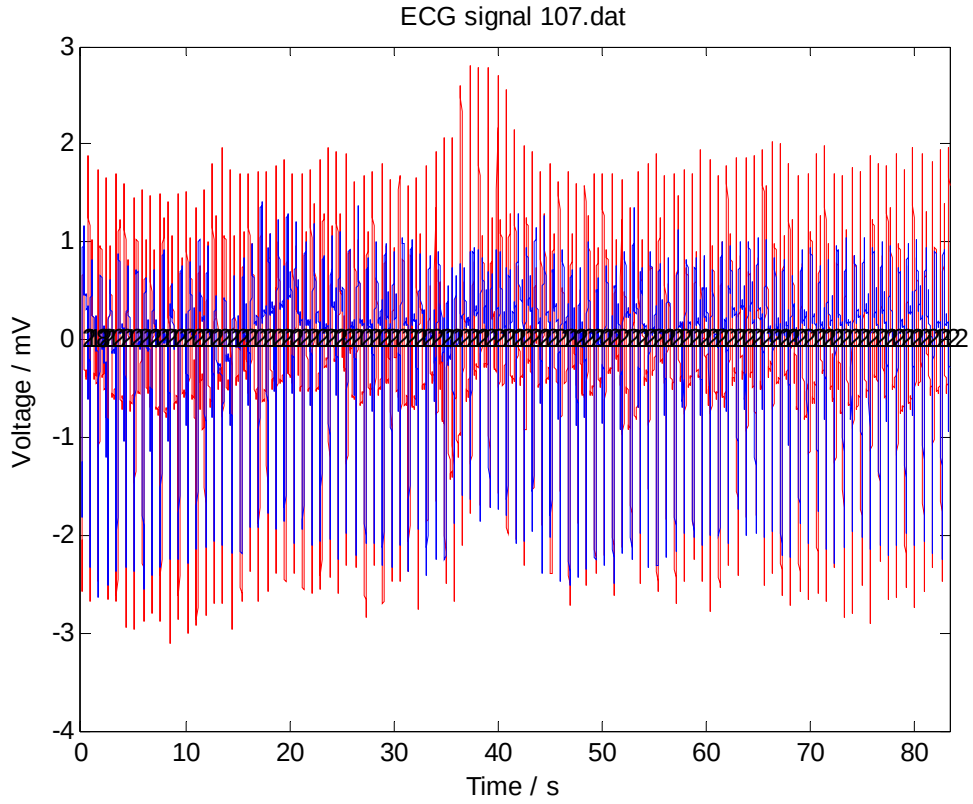
Το σήμα αυτό περιέχει 92 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 60. Πρόκειται για ΗΚΓ γυναίκας ηλικίας 24 ετών.



Εικόνα 60 Κυματομορφή του σήματος 106

- Σήμα 107

Το σήμα αυτό περιέχει 98 πραγματικά συμπλέγματα QRS. Η κυματομορφή του φαίνεται στην εικόνα 61. Πρόκειται για ΗΚΓ άνδρα ηλικίας 63 ετών.



Εικόνα 61 Κυματομορφή του σήματος 107

Όπως προκύπτει και με απλή παρατήρηση των ανωτέρω γραφικών παραστάσεων, ορισμένα από τα σήματα παρουσιάζουν αρκετές ιδιαιτερότητες ως προς τη μορφή τους. Πιο συγκεκριμένα, αφενός, τα σήματα 100 έως 103 καθώς και το 105 παρουσιάζουν μια επαναλαμβανόμενη κυματομορφή με τιμές χρόνου και πλάτους που δεν διαφοροποιούνται σημαντικά καθ' όλη τη διάρκεια του σήματος. Αφετέρου, τα

σήματα 104, 106 και 107 παρουσιάζουν σημαντικές διαφοροποιήσεις. Το πλάτος τους διαφέρει ανά χρονικές στιγμές, ενώ ορισμένα εμφανίζουν στοιχεία εντελώς διαφορετικά από τα αναμενόμενα ενός υγιούς ΗΚΓ σε κάποια σημεία τους. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι τα σήματα προέρχονται από διαφορετικά πρόσωπα, κάποια από τα οποία, ενδεχομένως εμφανίζουν καρδιακές παθήσεις ή ακόμη και κάποιο επεισόδιο κατά τη διάρκεια καταγραφής του ΗΚΓ.

4.3 Παρουσίαση και αξιολόγηση των τεχνικών ανίχνευσης QRS

Στο κεφάλαιο αυτό, θα γίνει παρουσίαση και αξιολόγηση τριών βασικών τεχνικών ανίχνευσης συμπλέγματος QRS σε ΗΚΓ. Οι τεχνικές αυτές βασίζονται στους γενικότερους αλγορίθμους που παρουσιάστηκαν στο προηγούμενο κεφάλαιο και συγκεκριμένα είναι οι εξής:

1. Τεχνική με φίλτρο Savitzky-Golay και σταθερό κατώφλι
2. Τεχνική με προσαρμοστικό φίλτρο και κατώφλι
3. Τεχνική βασισμένη σε παραγώγους με φίλτρο Savitzky-Golay και σταθερό κατώφλι

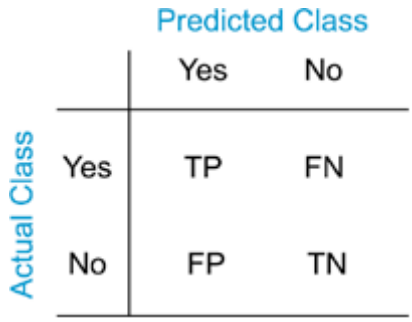
Αναλυτικά, οι κώδικες που υλοποιούν τους αλγόριθμους αυτούς φαίνονται στο Παράρτημα Α.

Για την αξιολόγηση των τεχνικών αυτών θα εφαρμόσουμε μια στατιστική ανάλυση που βασίζεται στις εξής τρεις κατηγορίες ανιχνεύσεων

- **TP** – True Positive. Αφορά τους παλμούς που υπάρχουν στο σήμα και ανιχνεύονται από την εκάστοτε μέθοδο.

- **FP** – False Positive.Αφορά τους παλμούς που δεν υπάρχουν στο σήμα αλλά ανιχνεύονται από την εκάστοτε μέθοδο.

- **FN** –False Negative.Αφορά τους παλμούς που υπάρχουν στο σήμα αλλά δεν ανιχνεύονται από την εκάστοτε μέθοδο.



Εικόνα 62 Οι διάφορες κατηγορίες ανιχνεύσεων συμπλέγματος QRS

Για την αξιολόγηση του κάθε αλγορίθμου τα αποτελέσματα καταγράφονται 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 που ανιχνεύονται ορθά.

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

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

$$+P = \frac{TP}{TP + FP}$$

Οι παράμετροι αυτές παίρνουν τιμές από 0 έως 1. Όσο πιο κοντά στο 1 είναι οι τιμές τους, τόσο πιο αποδοτικός είναι ο αλγόριθμος ανίχνευσης QRS.

4.3.1 Τεχνική με φίλτρο Savitzky-Golay και σταθερό κατώφλι (1η τεχνική)

Στην τεχνική αυτή ακολουθήσαμε τα εξής βήματα:

- **Φιλτράρισμα**

Αρχικά, το σήμα περνάει μέσα από το φίλτρο Savitzky-Golay, το οποίο αποτελεί μία υποκατηγορία του φίλτρου κινούμενου μέσου όρου (moving average). Η μέθοδος Savitzky-Golay εφαρμόζει μια πολυωνυμική παλινδρόμηση σε μια κατανομή για να εξάγει μια τιμή με λιγότερες διακυμάνσεις (smoothed value), οι οποίες οφείλονται σε θόρυβο.

Ο κινούμενος μέσος όρος είναι μια 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. Το FIR θεωρείται πεπερασμένο (finite) λόγω του ότι δεν έχει ανάδραση (feedback). Έχει γραμμική φάση, είναι πάντα σταθερό και γραμμικά υλοποιήσιμο. Το μειονέκτημα τους είναι ότι πολλές φορές μαζί με το θόρυβο, φιλτράρουν ένα σημαντικό ποσοστό των υψηλών συχνοτήτων του σήματος.

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 πλεονέκτημα των Savitzky-Golay σε σχέση με τα προαναφερθέντα φίλτρα, είναι ότι διατηρούν τα χαρακτηριστικά του σήματος, όπως τα στοιχεία υψηλής συχνότητας.

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

- **Κατώφλι**

Στη συνέχεια, γίνεται υπολογισμός της μέσης τιμής (mean) και της τυπικής απόκλισης (std) του φιλτραρισμένου σήματος:

$$mean = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$std = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Έπειτα, το κατώφλι υπολογίζεται μέσω της σχέσης:

$$thresh = mean + l * std$$

Όπου l ένας συντελεστής με τιμές l=1, 1.4, 1.8, 2.2, 2.6, 3.

Το φιλτραρισμένο σήμα συγκρίνεται με το κατώφλι. Τα δείγματα των οποίων η τιμή το ξεπερνά τοποθετούνται στον πίνακα possibleQRS. Στη συνέχεια, ο αλγόριθμος εξετάζει αν τα δείγματα με μεγαλύτερη τιμή από το κατώφλι είναι διαδοχικά. Πιο συγκεκριμένα, εάν τα δείγματα που βρίσκονται στις θέσεις i και i+1 του πίνακα έχουν

διαφορά ίση με τη μονάδα θεωρούνται διαδοχικά και, συνεπώς ανήκουν στο ίδιο σύμπλεγμα QRS. Αν η διαφορά είναι μεγαλύτερη από τη μονάδα, τα δείγματα ανήκουν στο επόμενο σύμπλεγμα QRS.

```
filtered = smooth(ecg1,'sgolay');
%Mean value standard deviation, criterion y_up
mean1 = mean(filtered);
n = length(filtered);
stdev = std(filtered);
y_up = mean1 +l*stdev;

possibleQRS = find(filtered > y_up);
index = 1;

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

Ένα τελευταίο κριτήριο για την ανίχνευση των QRS είναι το κάθε πιθανό σύμπλεγμα να περιέχει έναν ελάχιστο ικανοποιητικό αριθμό δειγμάτων. Αυτό επιτυγχάνεται μέσω της σύγκρισης των δειγμάτων με ένα συντελεστή k με τιμές k=2, 6, 10, 14.

```
 if (index - temp >=k)
qrs_count = qrs_count + 1;
```

- **Ανίχνευση**

Τα συμπλέγματα QRS που ανιχνεύονται, καταχωρούνται σε ένα πίνακα. Έπειτα, συγκρίνονται με έναν πίνακα που περιέχει τα πραγματικά QRS, όπως αυτά έχουν προκύψει από τη βάση δεδομένων του MIT και έτσι εξάγονται αποτελέσματα σχετικά με τις TP, FP και FN ανιχνεύσεις, από τα οποία γίνεται και η αξιολόγηση της αποτελεσματικότητας της τεχνικής.

```
for i=1:length(A1)
 minval = abs(A1(i)-B(1:length(B)));
 test = find(minval == min(minval));
 if abs(A1(i)-B(test))<=20
 tp = tp + 1;
 C = [C B(test)];
 end
 end

 TP = [TP tp];

 fp = length(B)-length(C);
 FP =[FP fp];

 if length(A1)-length(C)>=0
 fn = length(A1)-length(C);
 FN = [FN fn];
 end
```

- **Αξιολόγηση**

Για την τεχνική αυτή έγιναν μετρήσεις στο περιβάλλον του MATLAB για την εύρεση των ανιχνεύσεων TP,FP και FN με διαφορετικές τιμές των συντελεστών k, l όπως αυτές αναφέρθηκαν παραπάνω. Οι δοκιμές έγιναν σε όλα τα σήματα και τα συγκεντρωτικά αποτελέσματα που προέκυψαν φαίνονται στον πίνακα 11.

ΤΕΧΝΙΚΗ 1								
100								
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	102	15	1	2	1	117	0,990291262	0,871794872
2	102	6	1	6	1	108	0,990291262	0,944444444
3	40	6	63	10	1	46	0,388349515	0,869565217
4	0	3	103	14	1	3	0	0
5	102	1	1	2	1.4	103	0,990291262	0,990291262
6	102	1	1	6	1.4	103	0,990291262	0,990291262

7	19	0	84	10	1.4	19	0,184466019	1
8	0	0	103	14	1.4	0	0	N/A
9	102	1	1	2	1.8	103	0,990291262	0,990291262
10	102	1	1	6	1.8	103	0,990291262	0,990291262
11	12	0	91	10	1.8	12	0,116504854	1
12	0	0	103	14	1.8	0	0	N/A
13	102	1	1	2	2.2	103	0,990291262	0,990291262
14	102	1	1	6	2.2	103	0,990291262	0,990291262
15	7	0	96	10	2.2	7	0,067961165	1
16	0	0	103	14	2.2	0	0	N/A
17	102	1	1	2	2.6	103	0,990291262	0,990291262
18	102	1	1	6	2.6	103	0,990291262	0,990291262
19	3	0	100	10	2.6	3	0,029126214	1
20	0	0	103	14	2.6	0	0	N/A
21	102	1	1	2	3.0	103	0,990291262	0,990291262
22	100	1	3	6	3.0	101	0,970873786	0,99009901
23	1	0	102	10	3.0	1	0,009708738	1
24	0	0	103	14	3.0	0	0	N/A
						101		
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	53	127	44	2	1	180	0,546391753	0,294444444
2	53	79	44	6	1	132	0,546391753	0,401515152
3	37	73	60	10	1	110	0,381443299	0,336363636
4	0	43	97	14	1	43	0	0
5	78	42	19	2	1.4	120	0,804123711	0,65
6	78	29	19	6	1.4	107	0,804123711	0,728971963
7	30	26	67	10	1.4	56	0,309278351	0,535714286
8	0	9	97	14	1.4	9	0	0
9	92	12	5	2	1.8	104	0,948453608	0,884615385
10	92	5	5	6	1.8	97	0,948453608	0,948453608
11	18	5	79	10	1.8	23	0,18556701	0,782608696

12	0	0	97	14	1.8	0	0	N/A
13	97	2	0	2	2.2	99	1	0,97979798
14	96	0	1	6	2.2	96	0,989690722	1
15	6	0	91	10	2.2	6	0,06185567	1
16	0	0	97	14	2.2	0	0	N/A
17	97	2	0	2	2.6	99	1	0,97979798
18	96	0	1	6	2.6	96	0,989690722	1
19	1	0	96	10	2.6	1	0,010309278	1
20	0	0	97	14	2.6	0	0	N/A
21	97	1	0	2	3.0	98	1	0,989795918
22	89	0	8	6	3.0	89	0,917525773	1
23	0	0	97	10	3.0	0	0	N/A
24	0	0	97	14	3.0	0	0	N/A
						102		
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	0	138	101	2	1	138	0	0
2	0	123	101	6	1	123	0	0
3	0	117	101	10	1	117	0	0
4	1	115	101	14	1	116	0,009803922	0,00862069
5	10	110	91	2	1.4	120	0,099009901	0,083333333
6	7	101	94	6	1.4	108	0,069306931	0,064814815
7	0	98	101	10	1.4	98	0	0
8	0	94	101	14	1.4	94	0	0
9	46	98	55	2	1.8	144	0,455445545	0,319444444
10	16	77	85	6	1.8	93	0,158415842	0,172043011
11	0	68	101	10	1.8	68	0	0
12	0	64	101	14	1.8	64	0	0
13	80	53	21	2	2.2	133	0,792079208	0,601503759
14	21	38	80	6	2.2	59	0,207920792	0,355932203
15	1	31	100	10	2.2	32	0,00990099	0,03125
16	0	27	101	14	2.2	27	0	0

17	95	25	6	2	2.6	120	0,940594059	0,791666667
18	8	20	93	6	2.6	28	0,079207921	0,285714286
19	1	14	100	10	2.6	15	0,00990099	0,066666667
20	0	12	101	14	2.6	12	0	0
21	97	14	4	2	3.0	111	0,96039604	0,873873874
22	1	10	100	6	3.0	11	0,00990099	0,090909091
23	1	7	100	10	3.0	8	0,00990099	0,125
24	0	4	101	14	3.0	4	0	0
103								
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	98	80	0	2	1	178	1	0,550561798
2	98	60	0	6	1	158	1	0,620253165
3	95	51	3	10	1	146	0,969387755	0,650684932
4	0	37	98	14	1	37	0	0
5	98	3	0	2	1.4	101	1	0,97029703
6	98	2	0	6	1.4	100	1	0,98
7	82	2	16	10	1.4	84	0,836734694	0,976190476
8	0	1	98	14	1.4	1	0	0
9	98	0	0	2	1.8	98	1	1
10	98	0	0	6	1.8	98	1	1
11	60	0	38	10	1.8	60	0,612244898	1
12	0	0	98	14	1.8	0	0	N/A
13	98	0	0	2	2.2	98	1	1
14	98	0	0	6	2.2	98	1	1
15	18	0	80	10	2.2	18	0,183673469	1
16	0	0	98	14	2.2	0	0	N/A
17	98	0	0	2	2.6	98	1	1
18	98	0	0	6	2.6	98	1	1
19	2	0	96	10	2.6	2	0,020408163	1
20	0	0	98	14	2.6	0	0	N/A
21	98	0	0	2	3.0	98	1	1

22	98	0	0	6	3.0	98	1	1
23	0	0	98	10	3.0	0	0	N/A
24	0	0	98	14	3.0	0	0	N/A
104								
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	68	152	35	2	1	220	0,660194175	0,309090909
2	20	73	83	6	1	93	0,194174757	0,215053763
3	4	66	99	10	1	70	0,038834951	0,057142857
4	3	57	100	14	1	60	0,029126214	0,05
5	85	54	18	2	1.4	139	0,825242718	0,611510791
6	12	27	91	6	1.4	39	0,116504854	0,307692308
7	9	25	94	10	1.4	34	0,087378641	0,264705882
8	5	21	98	14	1.4	26	0,048543689	0,192307692
9	91	32	12	2	1.8	123	0,883495146	0,739837398
10	14	14	89	6	1.8	28	0,13592233	0,5
11	14	13	89	10	1.8	27	0,13592233	0,518518519
12	7	10	96	14	1.8	17	0,067961165	0,411764706
13	93	19	10	2	2.2	112	0,902912621	0,830357143
14	17	10	86	6	2.2	27	0,165048544	0,62962963
15	16	10	87	10	2.2	26	0,155339806	0,615384615
16	8	6	95	14	2.2	14	0,077669903	0,571428571
17	95	16	8	2	2.6	111	0,922330097	0,855855856
18	20	4	83	6	2.6	24	0,194174757	0,833333333
19	12	3	91	10	2.6	15	0,116504854	0,8
20	3	2	100	14	2.6	5	0,029126214	0,6
21	76	12	27	2	3.0	88	0,737864078	0,863636364
22	22	1	81	6	3.0	23	0,213592233	0,956521739
23	8	0	95	10	3.0	8	0,077669903	1
24	0	0	103	14	3.0	0	0	N/A
105								
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα**	**Sensitivity**	**Positive**

						QRS complexes		Predictivity
1	0	118	116	2	1	118	0	0
2	0	116	116	6	1	116	0	0
3	0	116	116	10	1	116	0	0
4	0	115	116	14	1	115	0	0
5	0	115	116	2	1.4	115	0	0
6	0	115	116	6	1.4	115	0	0
7	0	115	116	10	1.4	115	0	0
8	0	112	116	14	1.4	112	0	0
9	2	113	114	2	1.8	115	0,017241379	0,017391304
10	2	113	114	6	1.8	115	0,017241379	0,017391304
11	2	113	114	10	1.8	115	0,017241379	0,017391304
12	1	89	115	14	1.8	90	0,00862069	0,011111111
13	12	103	104	2	2.2	115	0,103448276	0,104347826
14	12	103	104	6	2.2	115	0,103448276	0,104347826
15	12	103	104	10	2.2	115	0,103448276	0,104347826
16	4	25	112	14	2.2	29	0,034482759	0,137931034
17	54	61	62	2	2.6	115	0,465517241	0,469565217
18	54	61	62	6	2.6	115	0,465517241	0,469565217
19	50	60	66	10	2.6	110	0,431034483	0,454545455
20	4	3	112	14	2.6	7	0,034482759	0,571428571
21	106	9	10	2	3.0	115	0,913793103	0,92173913
22	106	9	10	6	3.0	115	0,913793103	0,92173913
23	57	7	59	10	3.0	64	0,49137931	0,890625
24	3	1	113	14	3.0	4	0,025862069	0,75
106								
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	79	70	13	2	1	149	0,858695652	0,530201342
2	79	60	13	6	1	139	0,858695652	0,568345324
3	79	56	13	10	1	135	0,858695652	0,585185185
4	0	38	92	14	1	38	0	0

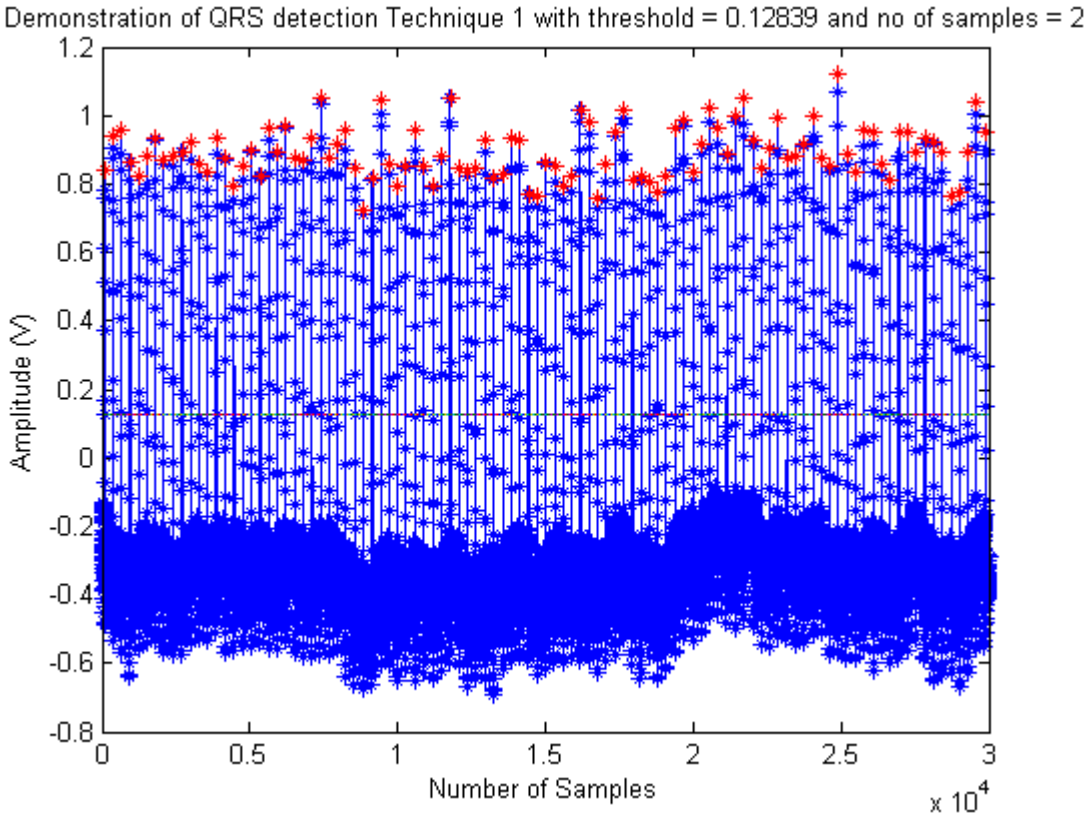
5	88	25	4	2	1.4	113	0,956521739	0,778761062
6	88	22	4	6	1.4	110	0,956521739	0,8
7	88	20	4	10	1.4	108	0,956521739	0,814814815
8	0	13	92	14	1.4	13	0	0
9	91	15	1	2	1.8	106	0,989130435	0,858490566
10	91	14	1	6	1.8	105	0,989130435	0,866666667
11	76	13	16	10	1.8	89	0,826086957	0,853932584
12	0	10	92	14	1.8	10	0	0
13	92	10	0	2	2.2	102	1	0,901960784
14	92	8	0	6	2.2	100	1	0,92
15	42	8	50	10	2.2	50	0,456521739	0,84
16	0	4	92	14	2.2	4	0	0
17	92	3	0	2	2.6	95	1	0,968421053
18	92	1	0	6	2.6	93	1	0,989247312
19	18	1	74	10	2.6	19	0,195652174	0,947368421
20	0	0	92	14	2.6	0	0	N/A
21	92	0	0	2	3.0	92	1	1
22	84	0	8	6	3.0	84	0,913043478	1
23	8	0	84	10	3.0	8	0,086956522	1
24	0	0	92	14	3.0	0	0	N/A
						107		
	TP	**FP**	**FN**	**K**	**L**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	1	189	97	2	1	190	0,010204082	0,005263158
2	1	187	97	6	1	188	0,010204082	0,005319149
3	0	184	98	10	1	184	0	0
4	0	171	98	14	1	171	0	0
5	9	155	89	2	1.4	164	0,091836735	0,054878049
6	7	147	91	6	1.4	154	0,071428571	0,045454545
7	0	135	98	10	1.4	135	0	0
8	0	72	98	14	1.4	72	0	0
9	61	51	37	2	1.8	112	0,62244898	0,544642857

10	33	49	65	6	1.8	82	0,336734694	0,402439024
11	0	27	98	10	1.8	27	0	0
12	0	97	98	14	1.8	97	0	0
13	71	7	27	2	2.2	78	0,724489796	0,91025641
14	9	6	89	6	2.2	15	0,091836735	0,6
15	0	6	98	10	2.2	6	0	0
16	0	5	98	14	2.2	5	0	0
17	9	5	89	2	2.6	14	0,091836735	0,642857143
18	0	5	98	6	2.6	5	0	0
19	0	0	98	10	2.6	0	0	N/A
20	2	4	96	14	2.6	6	0,020408163	0,333333333
21	1	4	97	2	3.0	5	0,010204082	0,2
22	0	0	98	6	3.0	0	0	N/A
23	0	0	98	10	3.0	0	0	N/A

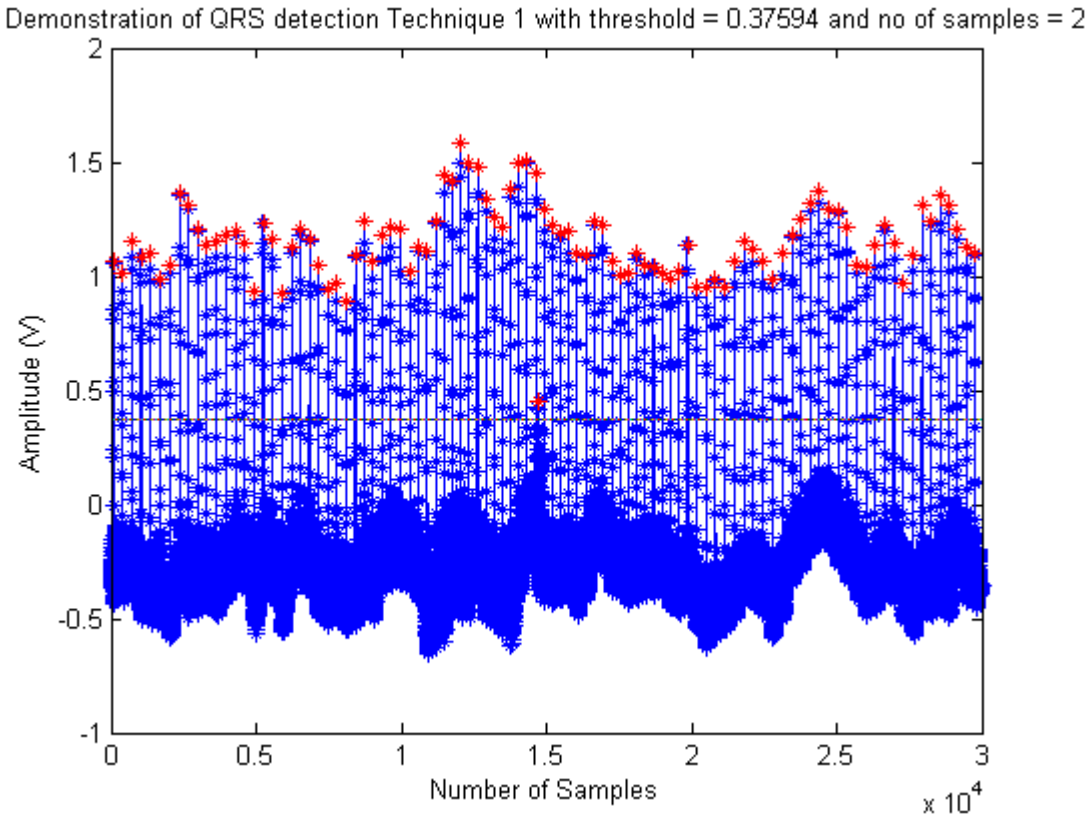
Πίνακας 11 Συγκεντρωτικά αποτελέσματα και αξιολόγηση της Τεχνικής 1 για όλα τα σήματα

Το N/A που εμφανίζεται στον πίνακα προκύπτει όταν ο παρονομαστής του κλάσματος είναι μηδέν. Στις περιπτώσεις αυτές δεν μπορούμε να εξάγουμε συμπεράσματα για την παράμετρο Positive Predictivity.

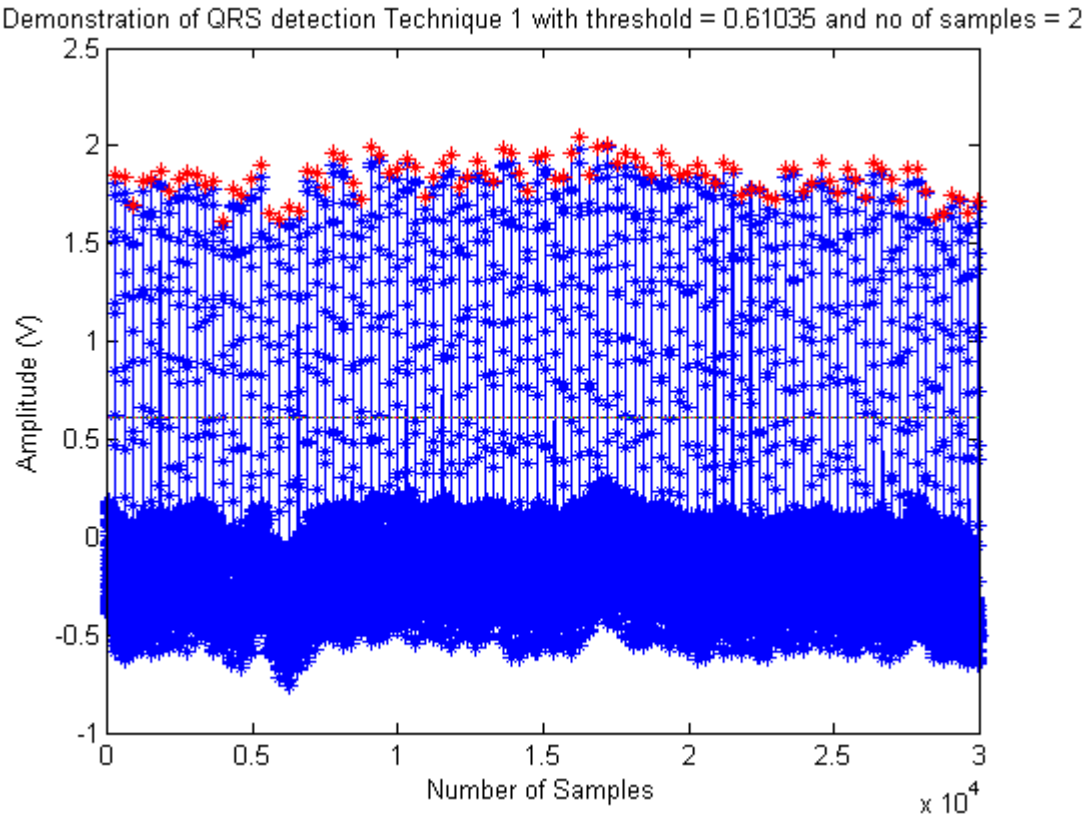
Βασιζόμενοι στις παραμέτρους Sensitivity και Positive Predictivity μπορούμε να εξάγουμε συμπεράσματα για το ποιες δοκιμές έδωσαν καλά αποτελέσματα και ποιες όχι. Πιο συγκεκριμένα, παρατηρούμε ότι για το συνδυασμό τιμών k=2, l=2.6 καθώς και για τις τιμές k=2, l=3 οι παράμετροι S και +P παίρνουν τιμές πολύ κοντά στο 1 για όλα σχεδόν τα σήματα. Ενδεικτικά, παρουσιάζονται οι γραφικές παραστάσεις για ορισμένα από τα σήματα, με αυτές τις τιμές. Στις γραφικές παραστάσεις τα κόκκινα σημεία αποτελούν τις επιβεβαιωμένες ανιχνεύσεις και το κατώφλι εμφανίζεται ως μια κόκκινη γραμμή.



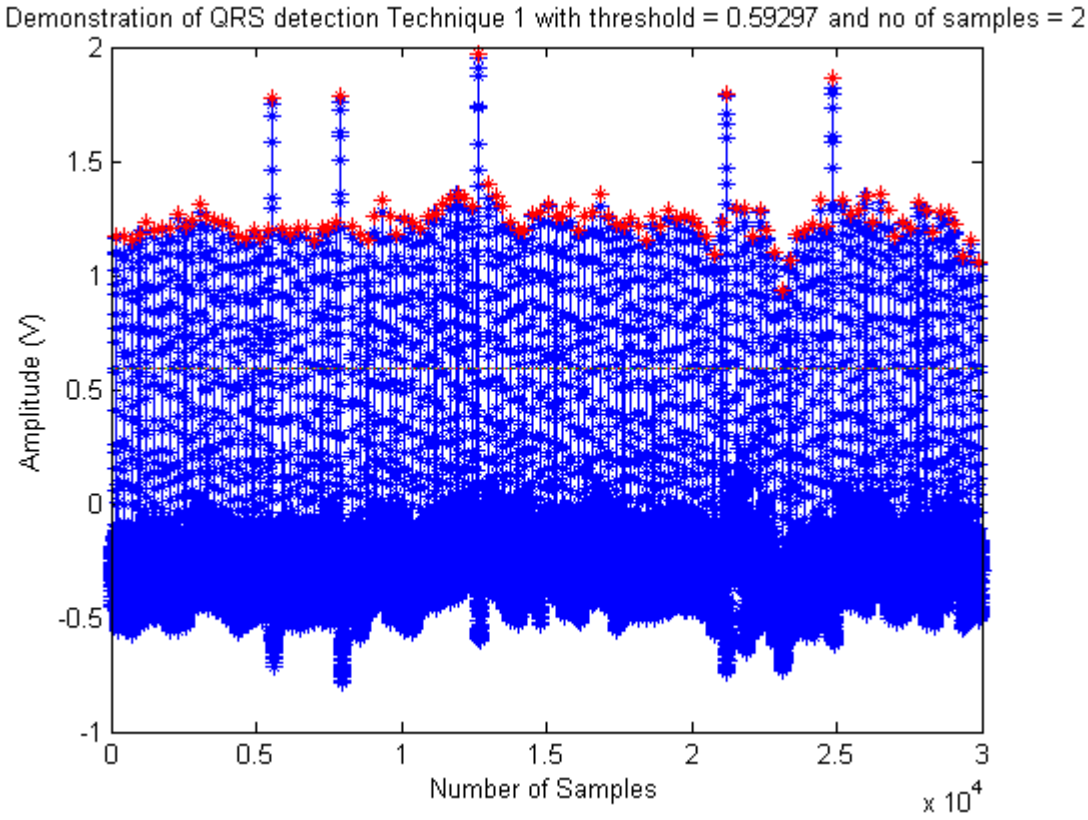
Εικόνα 63 Σήμα 100: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την μαύρη γραμμή



Εικόνα 64 Σήμα101: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

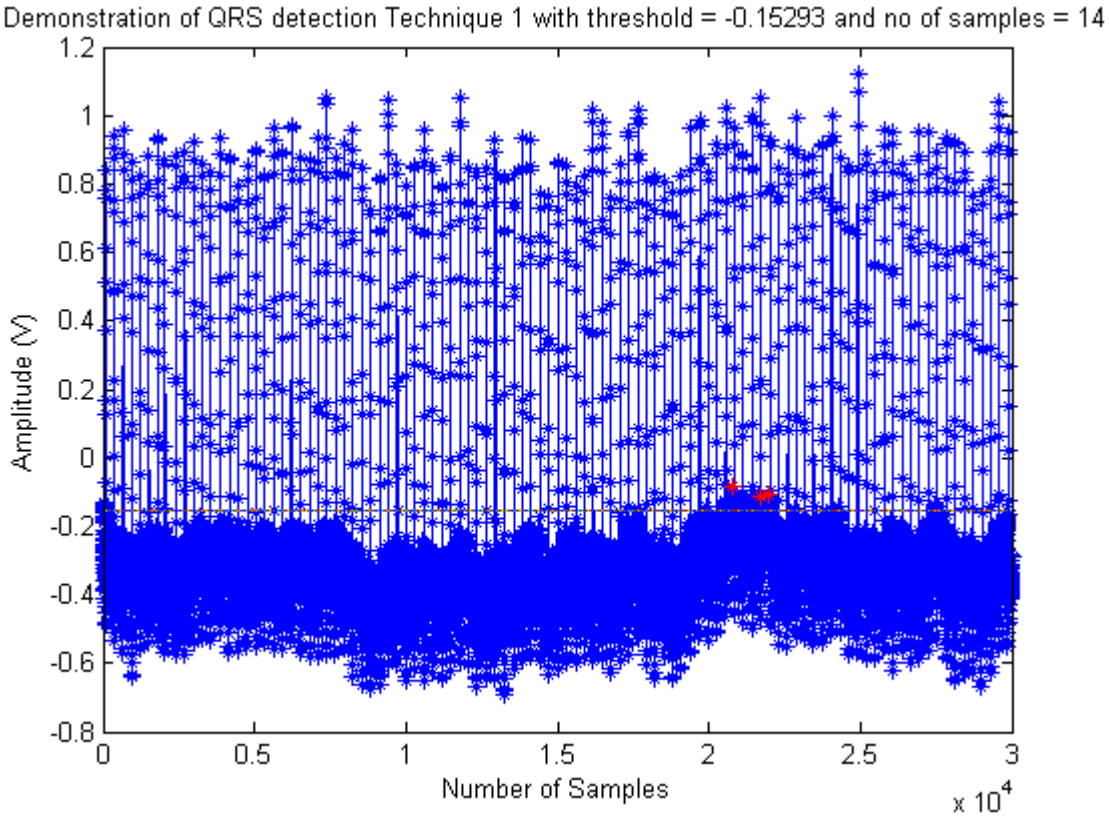


Εικόνα 65 Σήμα 103: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

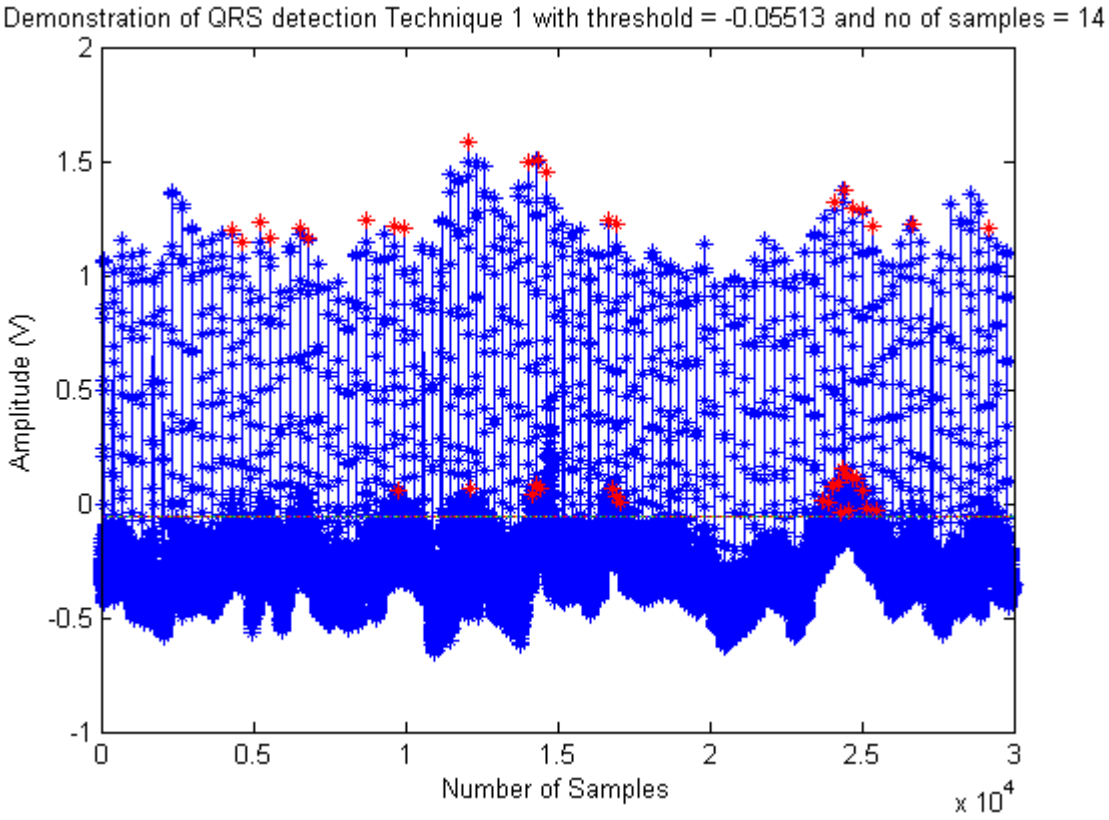


Εικόνα 66 Σήμα 105: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

Σε κάποιες άλλες δοκιμές με άλλους συνδυασμούς τιμών k, l, τα αποτελέσματα δεν ήταν τόσο καλά. Αυτό μπορεί να οφείλεται στο γεγονός ότι, λόγω μεγάλης τιμής του l, το κατώφλι έλαβε μεγάλη τιμή ενώ, ταυτόχρονα, η τιμή του k ήταν πολύ μικρή για να μπορέσει να ανιχνεύσει με επιτυχία τις κορυφές του σήματος. Κατά συνέπεια, οι παράμετροι S και +P έλαβαν χαμηλές τιμές. Ενδεικτικά, παρατίθενται γραφικές παραστάσεις για ορισμένα από τα σήματα, με αυτές τις τιμές.



Εικόνα 67 Σήμα 100: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή



Εικόνα 68 Σήμα 101: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

Η τεχνική αυτή στηρίζεται σε πολύ απλή επεξεργασία του σήματος (φίλτρο). Η ανίχνευση γίνεται συγκρίνοντας το σήμα με ένα κατώφλι. Παρατηρούμε ότι σε σήματα που δεν παρουσιάζουν εξάρσεις τιμών ή άλλες ιδιαιτερότητες, η τεχνική έχει πολύ καλή ως και άριστη απόδοση στην ορθή ανίχνευση των συμπλεγμάτων QRS. Ωστόσο, σε σήματα με ιδιομορφίες, όπως αυτά που δεν αντιστοιχούν σε υγιή ΗΚΓ και η κυματομορφή τους παρουσιάζει διακυμάνσεις, κάποια συμπλέγματα ήταν αδύνατο να ανιχνευθούν. Αυτό μπορούσε μόνο να επιτευχθεί με ακραίες τιμές των συντελεστών, κάτι που, τελικά, είχε αρνητική επίδραση στη συνολική απόδοση του αλγορίθμου.

4.3.2 Τεχνική με προσαρμοστικό φίλτρο και κατώφλι (2η τεχνική)

Τα βήματα της τεχνικής αυτής είναι τα εξής:

- **Φιλτράρισμα**

Ο κώδικας παίρνει τον πίνακα (ecg1[i]) στον οποίο είναι καταχωρημένα τα δείγματα του σήματος ΗΚΓ και για κάθε θέση i υπολογίζει τη διαφορά *ecg1(i)-ecg1(i-3)* την οποία και τοποθετεί σε ένα νέο πίνακα y_diff. Το νέο αυτό σήμα φιλτράρεται στο τελικό y_diff_filtered μέσω του αλγόριθμου

```
for i = 5:length(y_diff)
 y_diff_filtered(i-4) = y_diff(i) + 4*y_diff(i-1) +
6*y_diff(i-2) + 4*y_diff(i-3) + y_diff(i-4);
end
```

ο οποίος στην ουσία, για κάθε y_diff(i) υπολογίζει έναν γραμμικό συνδυασμό αυτού και των επόμενων διαδοχικά δειγμάτων του.

- **Κατώφλι**

Το κατώφλι υπολογίζεται δυναμικά στην τεχνική αυτή. Αυτό σημαίνει ότι η τιμή του ακολουθεί τη μορφολογία του σήματος αντί να είναι στατικό. Στην υλοποίηση αυτή, το σήμα y_diff_filtered χωρίζεται σε παράθυρα των 250 δειγμάτων. Το εύρος αυτό επιλέγεται διότι είναι ικανοποιητικό ώστε να περιέχει τουλάχιστον ένα σύμπλεγμα QRS. Για κάθε παράθυρο το δείγμα με τη μέγιστη τιμή καταχωρείται στον πίνακα test, καθώς αποτελεί πιθανή κορυφή συμπλέγματος QRS.

```
for j = 1:n
 start_index = end_index + 1;
 end_index = j*250;
 test = [test find(y_diff_filtered_abs ==
max(y_diff_filtered_abs(start_index:end_index)))];
 if (j == n)
 start_index = end_index + 1;
 end_index = length(y_diff_filtered_abs);
 test = [test find(y_diff_filtered_abs ==
max(y_diff_filtered_abs(start_index:end_index)))];
 end
end
```

Έπειτα, γίνεται υπολογισμός ενός σταθερού κατωφλίου αναφοράς, σύμφωνα με τη σχέση που χρησιμοποιήθηκε και στην προηγούμενη τεχνική.

$$stat\_thresh = mean(ecg1) + l*std(ecg1)$$

Τελικά, το κατώφλι υπολογίζεται χωριστά για κάθε παράθυρο (250) δειγμάτων ως ένας γραμμικός συνδυασμός του κατωφλίου αναφοράς και της εκάστοτε τιμής του πίνακα test.

```
f = fix(30000/length(test));
stat_thresh = mean(ecg1) + l*std(ecg1);
for z = 1:length(test)
 thr = 0.8*stat_thresh + ecg1(test(z))/10;
 thresh = [thresh thr];
…end
```

Η ανίχνευση των κορυφών γίνεται συγκρίνοντας τις τιμές των δειγμάτων του εκάστοτε παραθύρου με το αντίστοιχο κατώφλι. Τα διαδοχικά δείγματα που υπερβαίνουν την τιμή του καταχωρούνται ως μία κορυφή. Τέλος, εφόσον τα δείγματα της κάθε κορυφής ικανοποιούν το κριτήριο της υπέρβασης ενός συγκεκριμένου αριθμού (k, με μεταβλητή τιμή), η κορυφή αυτή υπολογίζεται ως ένα σύμπλεγμα QRS.

```
QRS = find(ecg1((z-1)*f+1:z*f) > thresh(z));
 possibleQRS = QRS+((z-1)*f);
 index = 1;

 while(index <= length(possibleQRS))

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

 if (index - temp >=k)
 qrs_count = qrs_count + 1;
```

- **Ανίχνευση**

Τα συμπλέγματα QRS που ανιχνεύονται, καταχωρούνται σε ένα πίνακα. Έπειτα, συγκρίνονται με έναν πίνακα που περιέχει τα πραγματικά QRS, όπως αυτά έχουν προκύψει από τη βάση δεδομένων του MIT και έτσι εξάγονται αποτελέσματα σχετικά με τις TP, FP και FN ανιχνεύσεις, από τα οποία γίνεται και η αξιολόγηση της αποτελεσματικότητας της τεχνικής.

```
for y=1:length(k)
[qrs_temp,B] = texniki2a(ecg1,k(y));
totalqrs_count = totalqrs_count + qrs_temp;
minval=[];
C=[];
tp = 0;

for i=1:length(A7)
minval = abs(A7(i)-B(1:length(B)));
test1 = find(minval == min(minval));
if isempty(test1)~=1
if abs(A7(i)-B(test1))<=20
tp = tp + 1;
C = [C B(test1)];
end
end
end

TP = [TP tp];

fp = length(B)-length(C);
FP =[FP fp];

if length(A7)-length(C)>=0
fn = length(A7)-length(C);
FN = [FN fn];
end

end
```

- **Αξιολόγηση**

Για την αξιολόγηση της τεχνικής αυτής, έγιναν αναλύσεις στο περιβάλλον του MATLAB με διάφορες τιμές των συντελεστών k, l. Τα αποτελέσματα για όλα τα σήματα, φαίνονται συγκεντρωτικά στον πίνακα 12, όπου έχουν εξαχθεί και οι τιμές των TP, FP, FN ανιχνεύσεων, καθώς και οι τιμές των Sensitivity και Positive Predictivity, που αποτελούν το βασικό κριτήριο αξιολόγησης της τεχνικής.

ΤΕΧΝΙΚΗ 2								
100								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	102	1	1	1	2	103	0,990291262	0,990291262
2	102	1	1	2	2	103	0,990291262	0,990291262
3	102	1	1	2	3	103	0,990291262	0,990291262
4	102	1	1	5	2	103	0,990291262	0,990291262
5	102	1	1	6	2	103	0,990291262	0,990291262
6	94	1	9	6	3	95	0,912621359	0,989473684
7	5	0	98	10	2	5	0,048543689	1
8	1	0	102	10	3	1	0,009708738	1
101								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	97	4	0	2	2	101	1	0,96039604
2	97	1	0	2	3	98	1	0,989795918
3	97	0	0	6	2	97	1	1
4	91	0	6	6	3	91	0,93814433	1
5	18	0	79	10	2	18	0,18556701	1
6	1	0	96	10	3	1	0,010309278	1
102								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	79	66	22	2	2	145	0,782178218	0,544827586
2	100	20	1	2	3	120	0,99009901	0,833333333
3	51	43	50	6	2	94	0,504950495	0,542553191
4	12	14	89	6	3	26	0,118811881	0,461538462
5	37	32	64	10	2	69	0,366336634	0,536231884
6	9	9	92	10	3	18	0,089108911	0,5
103								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα**	**Sensitivity**	**Positive**

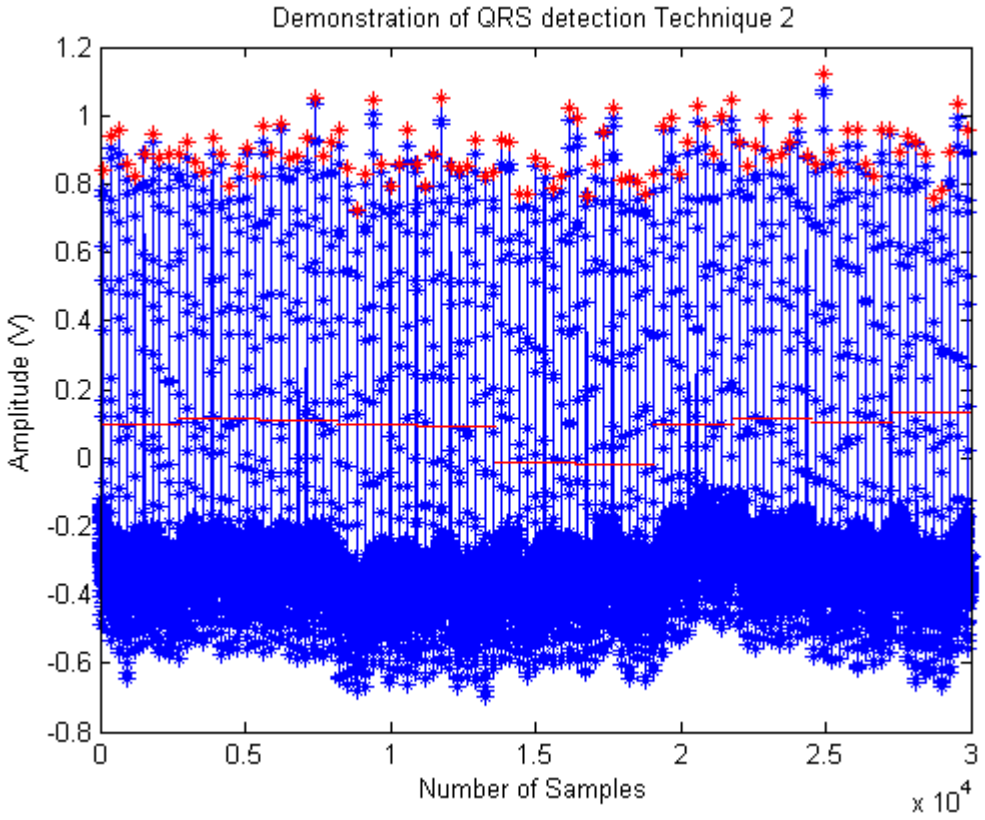
						QRS complexes		Predictivity
1	98	0	0	2	2	98	1	1
2	98	0	0	2	3	98	1	1
3	98	0	0	6	2	98	1	1
4	98	0	0	6	3	98	1	1
5	28	0	70	10	2	28	0,285714286	1
6	2	0	96	10	3	2	0,020408163	1
104								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	101	21	2	2	2	122	0,980582524	0,827868852
2	101	12	2	2	3	113	0,980582524	0,89380531
3	23	4	80	6	2	27	0,223300971	0,851851852
4	22	2	81	6	3	24	0,213592233	0,916666667
5	23	4	80	10	2	27	0,223300971	0,851851852
6	16	1	87	10	3	17	0,155339806	0,941176471
105								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	114	2	2	2	2	116	0,982758621	0,982758621
2	114	2	2	2	3	116	0,982758621	0,982758621
3	114	2	2	6	2	116	0,982758621	0,982758621
4	114	1	2	6	3	115	0,982758621	0,991304348
5	114	1	2	10	2	115	0,982758621	0,991304348
6	103	2	13	10	3	105	0,887931034	0,980952381
106								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	92	13	0	2	2	105	1	0,876190476
2	92	4	0	2	3	96	1	0,958333333
3	92	11	0	6	2	103	1	0,893203883
4	84	1	8	6	3	85	0,913043478	0,988235294

5	53	10	39	10	2	63	0,576086957	0,841269841
6	12	1	80	10	3	13	0,130434783	0,923076923
107								
	TP	**FP**	**FN**	**k**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	95	103	3	2	1	198	0,969387755	0,47979798
2	49	0	49	2	3	49	0,5	1
3	98	39	0	2	2	137	1	0,715328467
4	95	100	3	6	1	195	0,969387755	0,487179487
5	87	32	11	6	2	119	0,887755102	0,731092437
6	93	99	5	10	1	192	0,948979592	0,484375
7	47	30	51	10	2	77	0,479591837	0,61038961

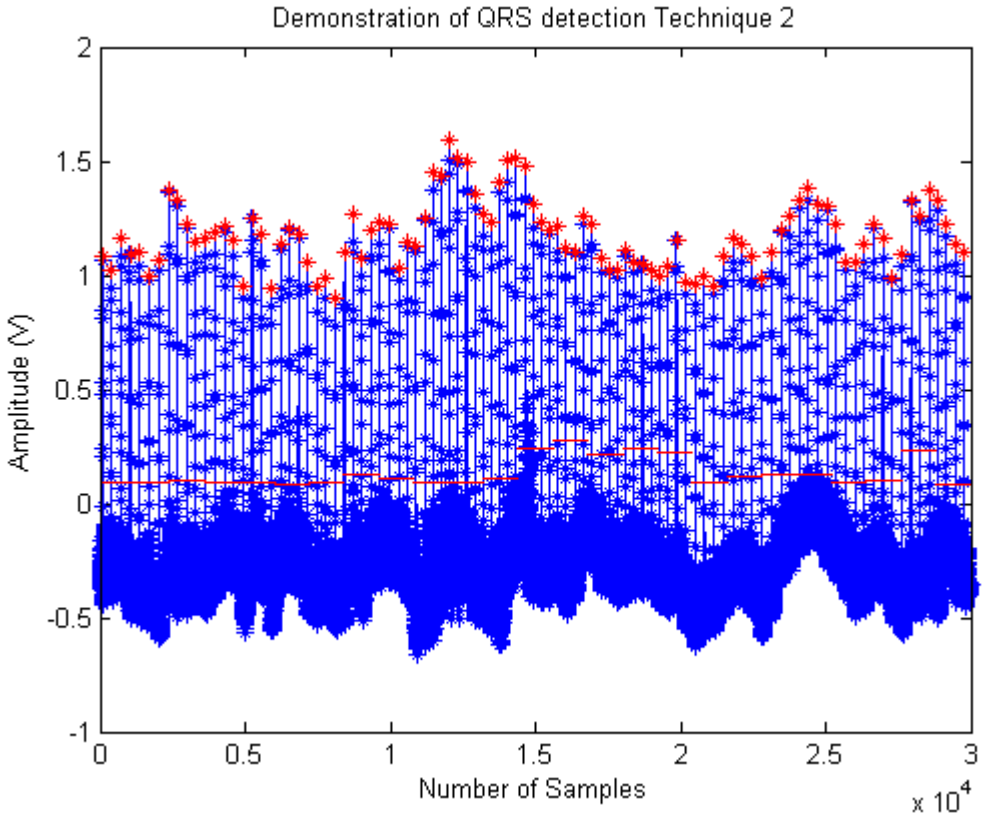
Πίνακας 12 Συγκεντρωτικά αποτελέσματα και αξιολόγηση της Τεχνικής 2 για όλα τα σήματα

Βασιζόμενοι στα αποτελέσματα των Sensitivity και Positive Predictivity διαπιστώνεται ότι δεν υπάρχουν συνδυασμοί τιμών που να δίνουν πολύ ικανοποιητικά αποτελέσματα σε όλα τα σήματα. Βέβαια ο συνδυασμός k=2, l=3, είναι σε πολύ μεγάλο ποσοστό αποδοτικός, αλλά προκύπτει ότι για κάποια σήματα οι αποκλίσεις από αυτό το συμπέρασμα είναι πολύ μεγάλες.

Αυτό οφείλεται στο γεγονός ότι η τεχνική αυτή στηρίζεται σε πιο πολύπλοκη επεξεργασία των δεδομένων, σε αντίθεση με την προηγούμενη τεχνική που συγκριτικά με την τεχνική αυτή είναι λιγότερο πολύπλοκη υπολογιστικά. Αυτό σημαίνει ότι αναλόγως με τη μορφή του σήματος, η επεξεργασία που αυτό θα υποστεί θα καταστήσει το σύμπλεγμα QRS περισσότερο ή λιγότερο ανιχνεύσιμο από την τεχνική. Πιο χαρακτηριστικά παρατηρούμε ότι για σήματα με σταθερά χαρακτηριστικά, η τεχνική δίνει πολύ καλά αποτελέσματα στην πλειοψηφία των δοκιμών, όπως φαίνεται και στις παρακάτω γραφικές παραστάσεις:

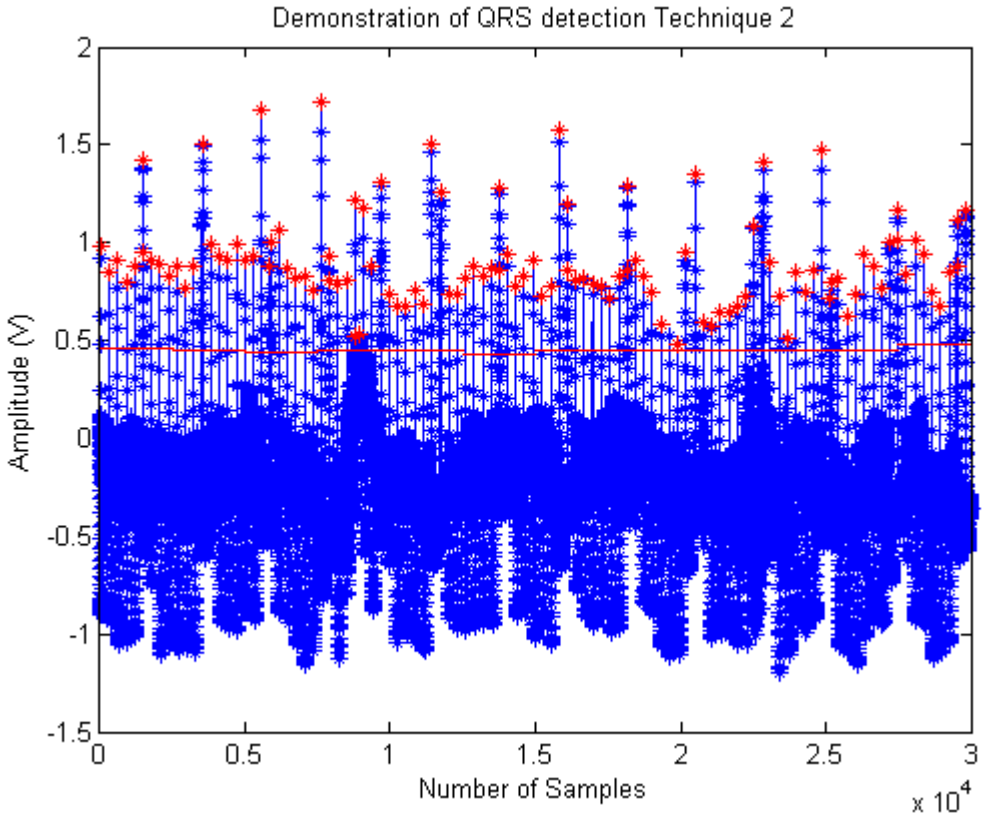


Εικόνα 69 Σήμα 100: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

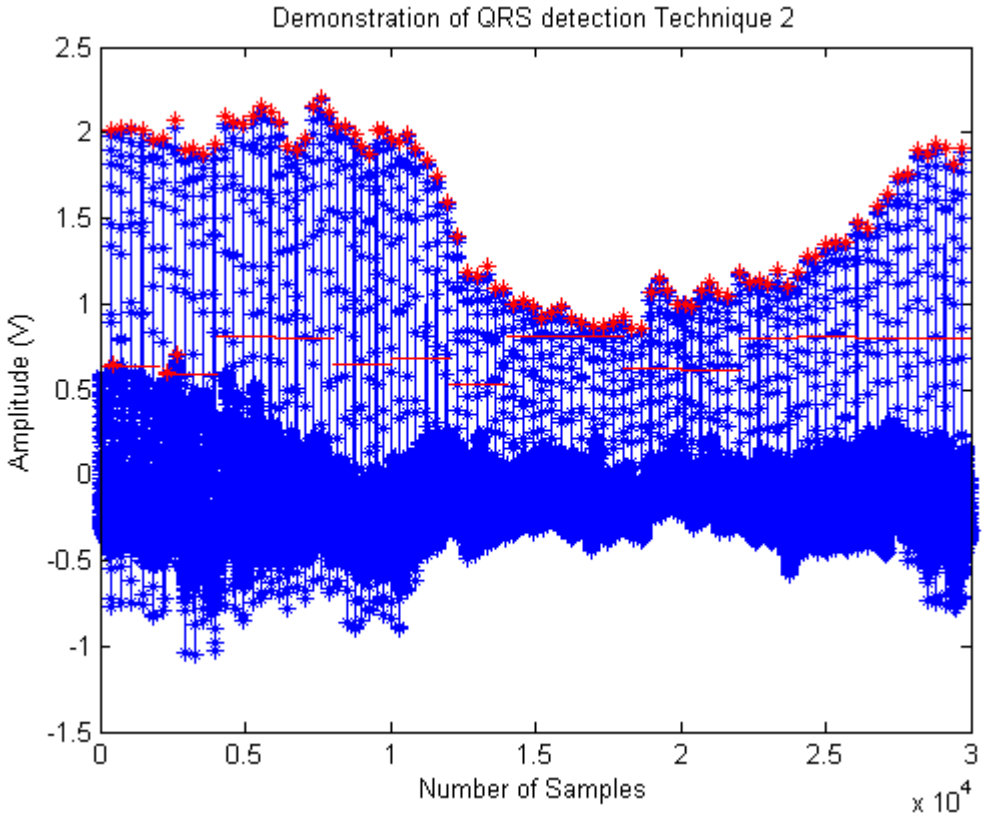


Εικόνα 70 Σήμα 101: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

Αλλά και σε σήματα, όπως τα 104, 106, 107 που παρουσίαζαν αρκετές ιδιαιτερότητες, τα αποτελέσματα ήταν πολύ καλά σε κάποιες δοκιμές:

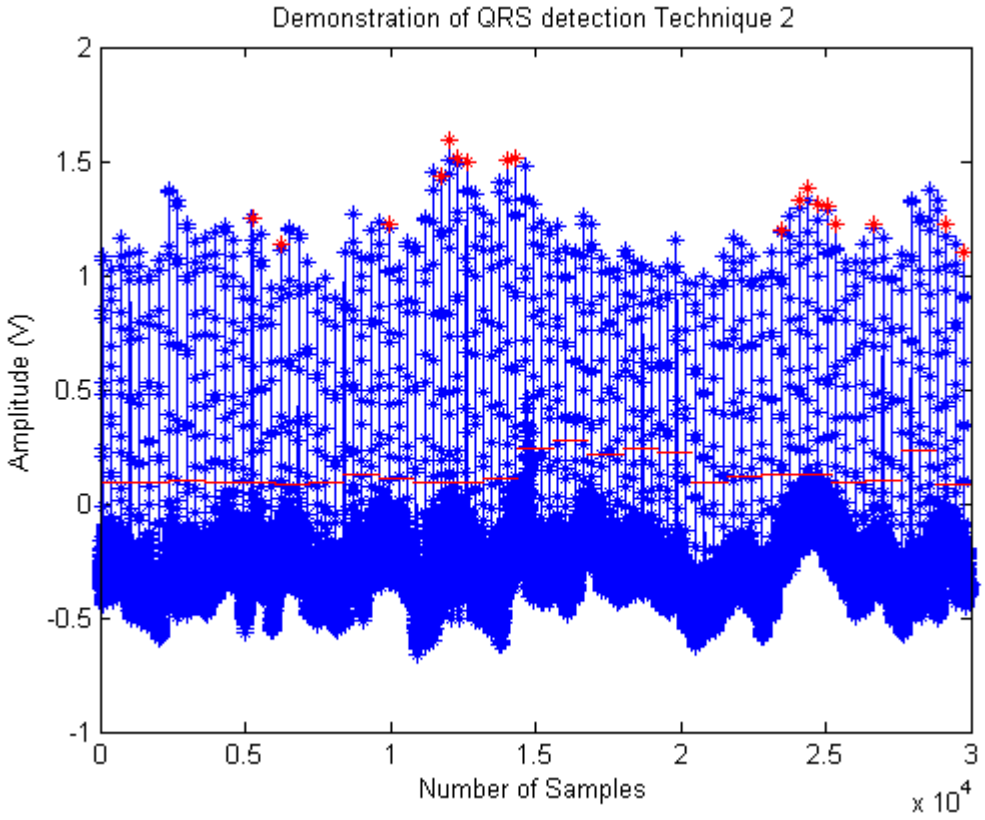


Εικόνα 71 Σήμα 104: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

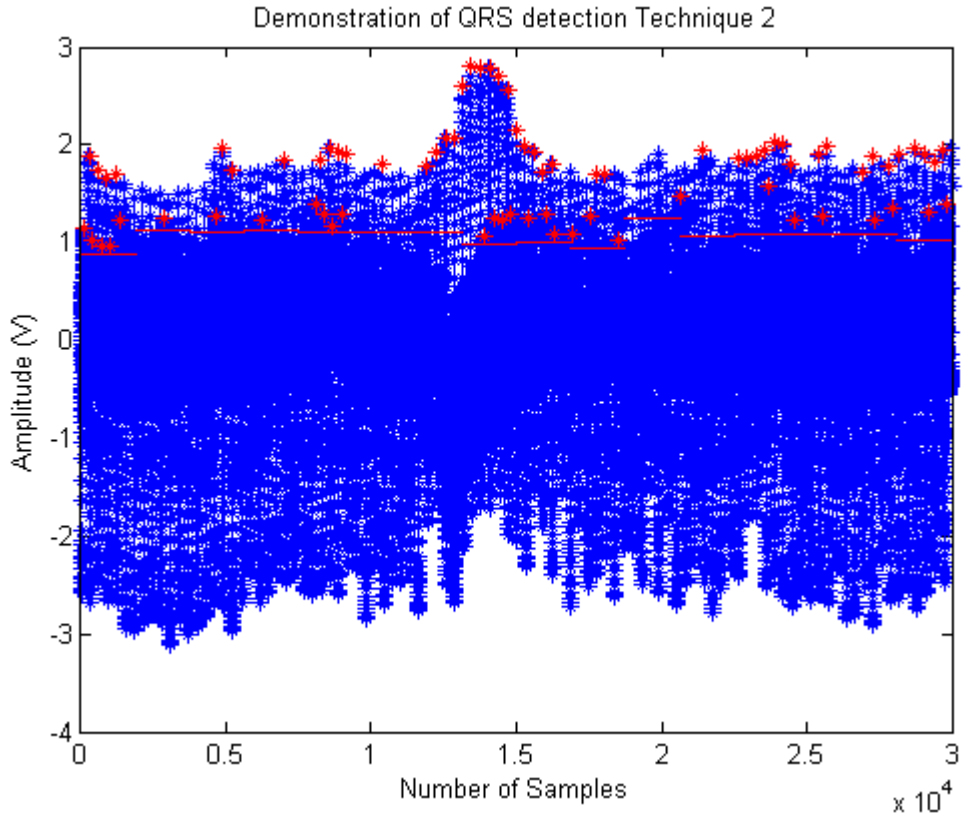


Εικόνα 72 Σήμα 106: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

Ωστόσο, κάποιες ακραίες τιμές των συντελεστών k, l δεν μπορούσαν να δώσουν ικανοποιητικά αποτελέσματα.



Εικόνα 73 Σήμα 101: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή



Εικόνα 74 Σήμα 107: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

4.3.3 Τεχνική βασισμένη σε παραγώγους με φίλτρο Savitzky-Golay και σταθερό κατώφλι (3η τεχνική)

Στην τεχνική αυτή ακολουθήσαμε τα εξής βήματα:

- **Φιλτράρισμα**

Αρχικά, το σήμα περνάει μέσα από το φίλτρο Savitzky-Golay, το οποίο αναλύθηκε παραπάνω.

```
function [peak_count,peaks] = technique3(ecg1,l)
index_array_x = [];
filtered_x= smooth(ecg1,'sgolay');
```

- **Εφαρμογή Παραγώγων**

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, έτσι ώστε να είναι πιο πιθανό να περιέχει ένα σύμπλεγμα QRS. Έπειτα, υπολογίζεται το μέγιστο δείγμα σε κάθε παράθυρο.

```
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
```

Στη συνέχεια, υπολογίζεται η παράγωγος του φιλτραρισμένου σήματος και γίνεται η υποθέση ότι σε κάθε παράθυρο υπάρχει ένα πιθανό σύμπλεγμα QRS που αντιστοιχεί σε ένα τοπικό μέγιστο, άρα στο δείγμα με τη μέγιστη τιμή μέσα στο παράθυρο και με μια σειρά από κριτήρια γίνεται προσπάθεια να επαληθευτεί.

```
diff_fltx = diff(filtered_x);

counter=0;
j=1;
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)];
```

Εάν το δείγμα αυτό με τη μέγιστη τιμή έχει και μηδενική πρώτη παράγωγο τότε αυτόματα πρόκειται για ένα πιθανό QRS. Αν η πρώτη παράγωγος του δείγματος είναι αρνητική και ταυτόχρονα του αμέσως προηγούμενου δείγματος θετική τότε πρόκειται για ακρότατο και πιθανό QRS

```
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
```

Τα πιθανά συμπλέγματα QRS τοποθετούνται σε ένα πίνακα.

- **Κατώφλι**

Στο σημείο αυτό γίνεται υπολογισμός της μέσης τιμής (mean) και της τυπικής απόκλισης (std) του φιλτραρισμένου σήματος, ώστε να βρεθεί το κατώφλι με τη σχέση: $thresh = mean + l * std$. Κάθε πιθανό QRS από αυτά που βρέθηκαν στο προηγούμενο βήμα της διαδικασίας συγκρίνεται με το κατώφλι.Στην περίπτωση που η

τιμή του είναι μεγαλύτερη από το κατώφλι θεωρείται σύμπλεγμα QRS. Αυτά τα QRS καταχωρούνται σε πίνακα.

```
thresh = mean(filtered_x) + l*std(filtered_x);

for i = 1:length(index_array_x)
 if filtered_x(index_array_x(i))>=thresh
 peaks = [peaks index_array_x(i)];
 peak_count = peak_count+1;
 end
end
```

- **Ανίχνευση**

Αυτά τα συμπλέγματα QRS συγκρίνονται με έναν πίνακα που περιέχει τα πραγματικά QRS, όπως αυτά έχουν προκύψει από τη βάση δεδομένων του MIT και έτσι εξάγονται αποτελέσματα σχετικά με τις TP, FP και FN ανιχνεύσεις, από τα οποία γίνεται και η αξιολόγηση της αποτελεσματικότητας της τεχνικής

```
for x=1:length(l)
 [qrs_temp,B] = technique3(ecg1,l(x));
 totalqrs_count = totalqrs_count + qrs_temp;
 minval=[];
 C=[];
 tp = 0;
 for z=1:length(A0)
 minval = abs(A0(z)-B(1:length(B)));
 test = find(minval == min(minval));
 if isempty(test)~=1

 if abs(A0(z)-B(test))<=5
  tp = tp + 1;
  C = [C B(test)];
 end
 end
 end

 TP = [TP tp];

 fp = length(B)-length(C);
 FP =[FP fp];

 if length(A0)-length(C)>=0
 fn = length(A0)-length(C);
 FN = [FN fn];
 end
 end
```

- **Αξιολόγηση**

Και για αυτήν την τεχνική έγιναν μετρήσεις στο περιβάλλον του MATLAB για την εύρεση των ανιχνεύσεων TP,FP και FN με διάφορες τιμές για την παράμετρο l ανάλογα με το σήμα, με κατώτατη τιμή το 1 και ανώτατη το 4.5.Οι δοκιμές έγιναν σε όλα τα σήματα και τα αποτελέσματα που προέκυψαν εμφανίζονται στον πίνακα 13.

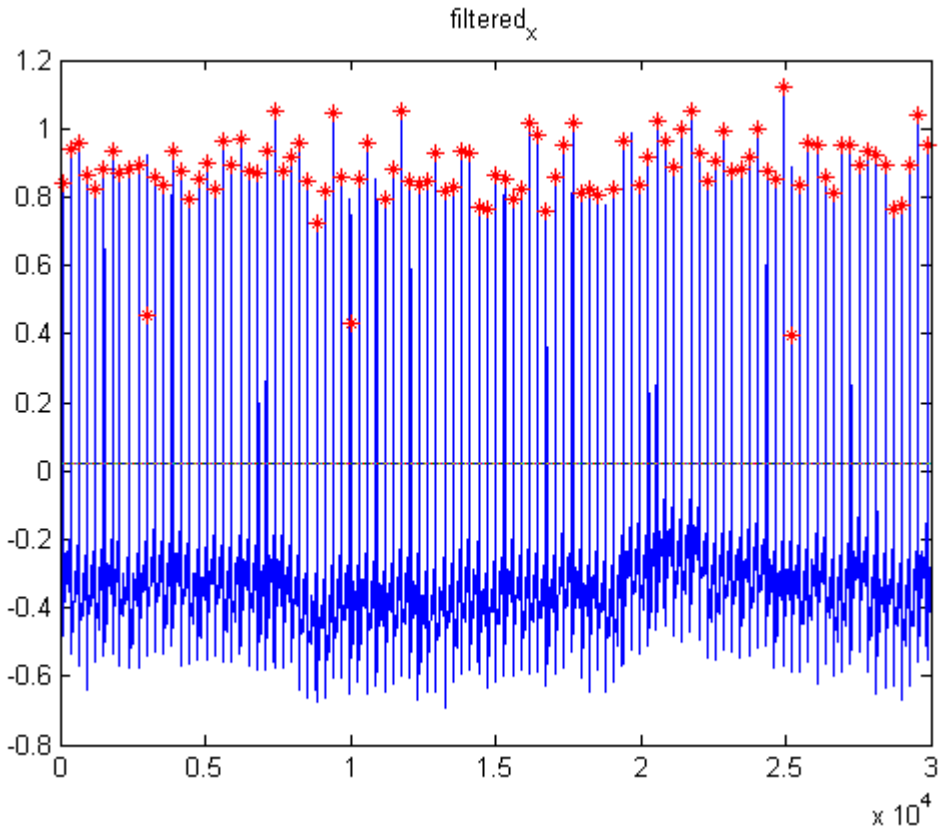
ΤΕΧΝΙΚΗ 3							
100							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	99	4	4	1	103	0.961165049	0.961165049
2	99	1	4	1.4	100	0.961165049	0.99
3	99	1	4	1.8	100	0.961165049	0.99
4	99	1	4	2	100	0.961165049	0.99
5	99	1	4	2.2	100	0.961165049	0.99
6	99	1	4	2.6	100	0.961165049	0.99
7	99	1	4	3	100	0.961165049	0.99
8	99	1	4	4	100	0.961165049	0.99
9	96	1	7	4.5	97	0.932038835	0.989690722
101							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	97	22	0	1	119	1	0.81512605
2	94	1	3	2	95	0.969072165	0.989473684
3	94	1	3	3	95	0.969072165	0.989473684
4	93	0	4	4	93	0.958762887	1
5	93	0	4	4.5	93	0.958762887	1
102							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	78	34	23	1	112	0.772277228	0.696428571
2	78	15	23	2	93	0.772277228	0.838709677
3	75	4	26	3	79	0.742574257	0.949367089

4	72	2	29	4	74	0.712871287	0.972972973
5	69	1	32	4.5	70	0.683168317	0.985714286
103							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	98	58	0	1	156	1	0.628205128
2	98	0	0	2	98	1	1
3	98	0	0	3	98	1	1
4	97	0	1	4	97	0.989795918	1
5	97	0	1	4.5	97	0.989795918	1
104							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictity**
1	87	45	16	1	132	0.844660194	0.659090909
2	86	9	17	2	95	0.834951456	0.905263158
3	80	3	23	3	83	0.776699029	0.963855422
4	25	2	78	4	27	0.242718447	0.925925926
105							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	106	10	10	1	116	0.913793103	0.913793103
2	106	3	10	2	109	0.913793103	0.972477064
3	104	2	12	3	106	0.896551724	0.981132075
4	102	2	14	4	104	0.879310345	0.980769231
5	81	2	35	4.5	83	0.698275862	0.975903614
106							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	92	37	0	1	129	1	0.713178295
2	92	9	0	2	101	1	0.910891089
3	92	0	0	3	92	1	1
4	72	0	20	4	72	0.782608696	1

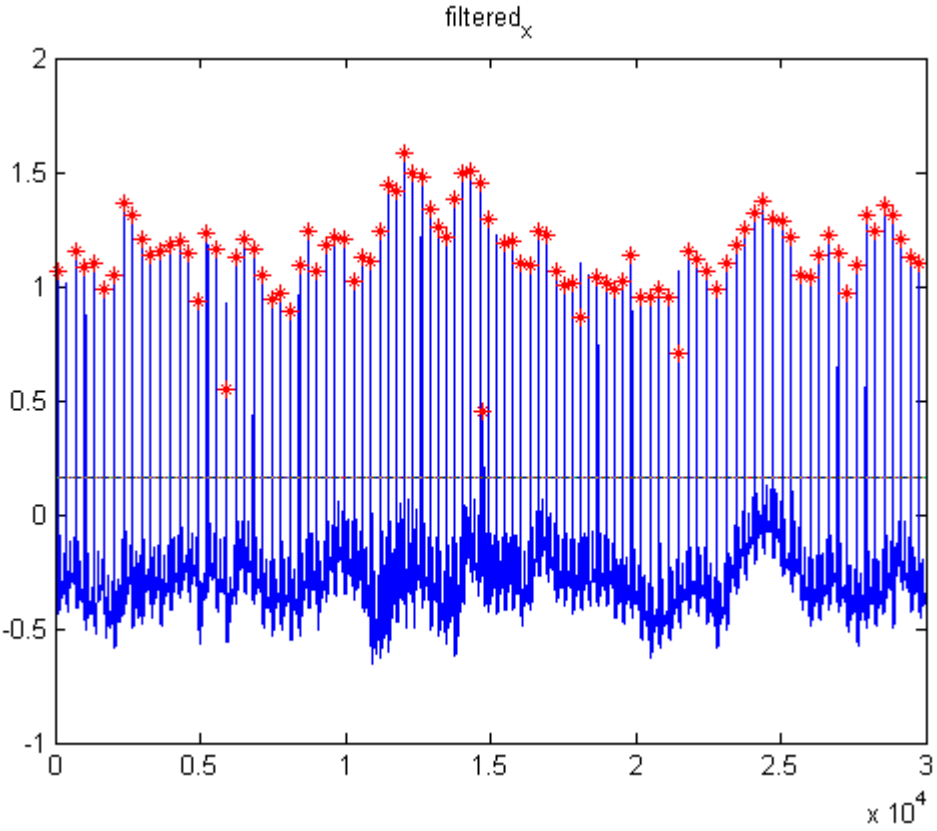
107							
	TP	**FP**	**FN**	**l**	**Επιβεβαιωμένα QRS complexes**	**Sensitivity**	**Positive Predictivity**
1	90	74	8	1	164	0.918367347	0.548780488
2	86	1	12	2	87	0.87755102	0.988505747
3	76	0	22	2.2	76	0.775510204	1
4	6	0	92	3	6	0.06122449	1

Πίνακας 13 Συγκεντρωτικά αποτελέσματα και αξιολόγηση της Τεχνικής 3 για όλα τα σήματα

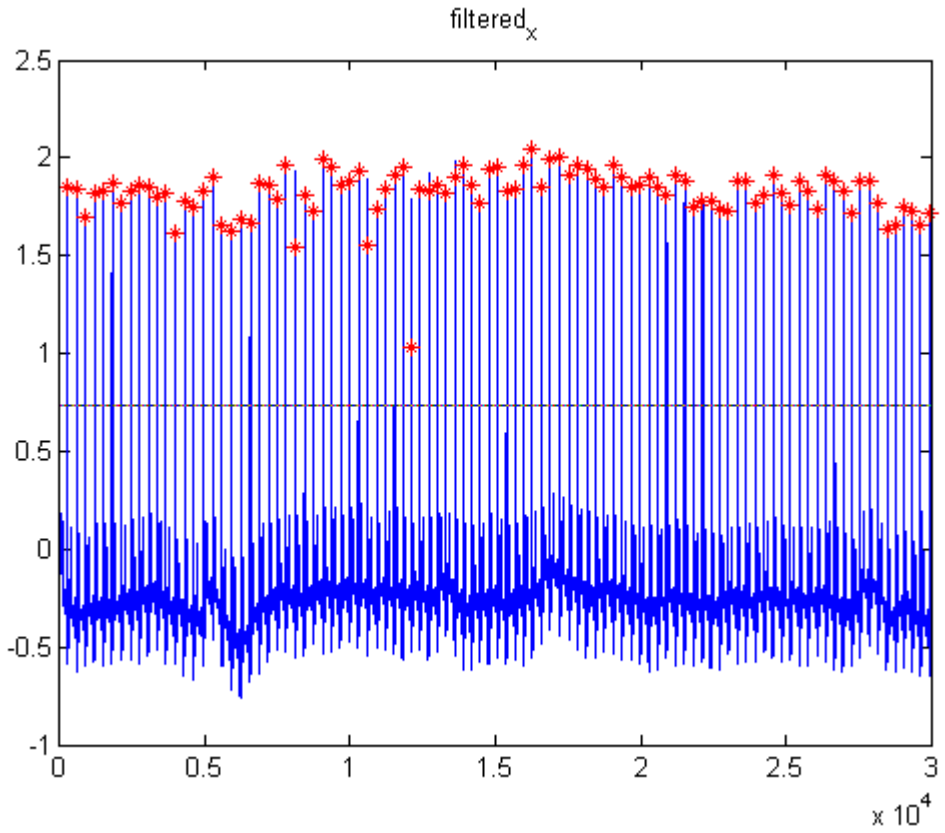
Θα βασιστούμε πάλι στις παραμέτρους S και +P, ώστε να εξάγουμε συμπέρασμα ως προς το ποιες τεχνικές έδωσαν καλά αποτελέσματα και ποιες όχι. Πιο συγκεκριμένα, για τιμές του συντελεστή l=2 και l=3, παρατηρούμε πως για όλα τα σήματα οι παράμετροι S και +P παίρνουν τιμή κοντά στο 1, κάτι που σημαίνει πως γι αυτές τις τιμές του l, και κατά συνέπεια του κατωφλιού η τεχνική καθίσταται αποδοτική. Ενδεικτικά, παρουσιάζονται οι γραφικές παραστάσεις για ορισμένα από τα σήματα, με αυτές τις τιμές:



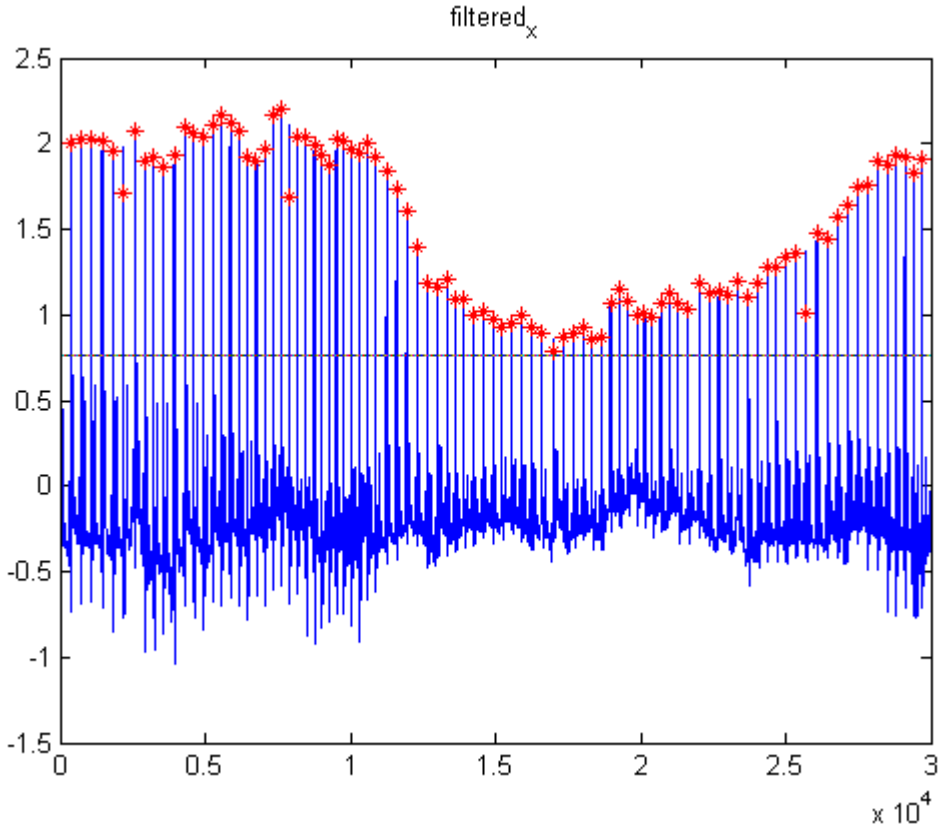
Εικόνα 75 Σήμα 100: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή



Εικόνα 76 Σήμα 101: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή

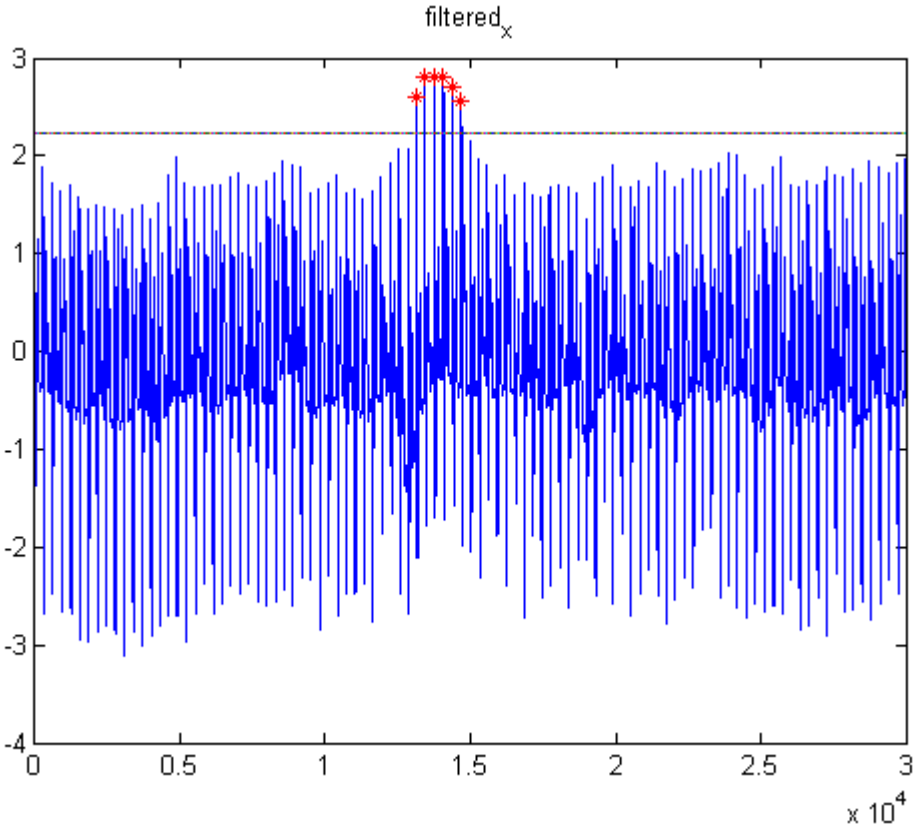


Εικόνα 77 Σήμα 103: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή



Εικόνα 78 Σήμα 106: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή

Ωστόσο, παρατηρούμε πως σε κάποια σήματα, όπως στο 107, ήταν δύσκολο να επιτευχθούν τιμές κοντά στο 1 και στη παράμετρο S και στην +P συγχρόνως. Αυτό οφείλεται στο ότι το ΗΚΓ είχε αρκετά περίπλοκη κυματομορφή που καθιστούσε δύσκολη την επιλογή του l, και κατά συνέπεια του κατωφλιού, που θα ήταν ιδανικό στην ανίχνευση του συμπλέγματος QRS σε όλο το εύρος του σήματος.



Εικόνα 79 Σήμα 107: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή

4.4 Συμπεράσματα

Μετά από εκτενείς δοκιμές και των τριών αλγορίθμων στο περιβάλλον του MATLAB και βασιζόμενοι στα στατιστικά στοιχεία που προέκυψαν, καταλήγουμε στα εξής συμπεράσματα για την κάθε τεχνική:

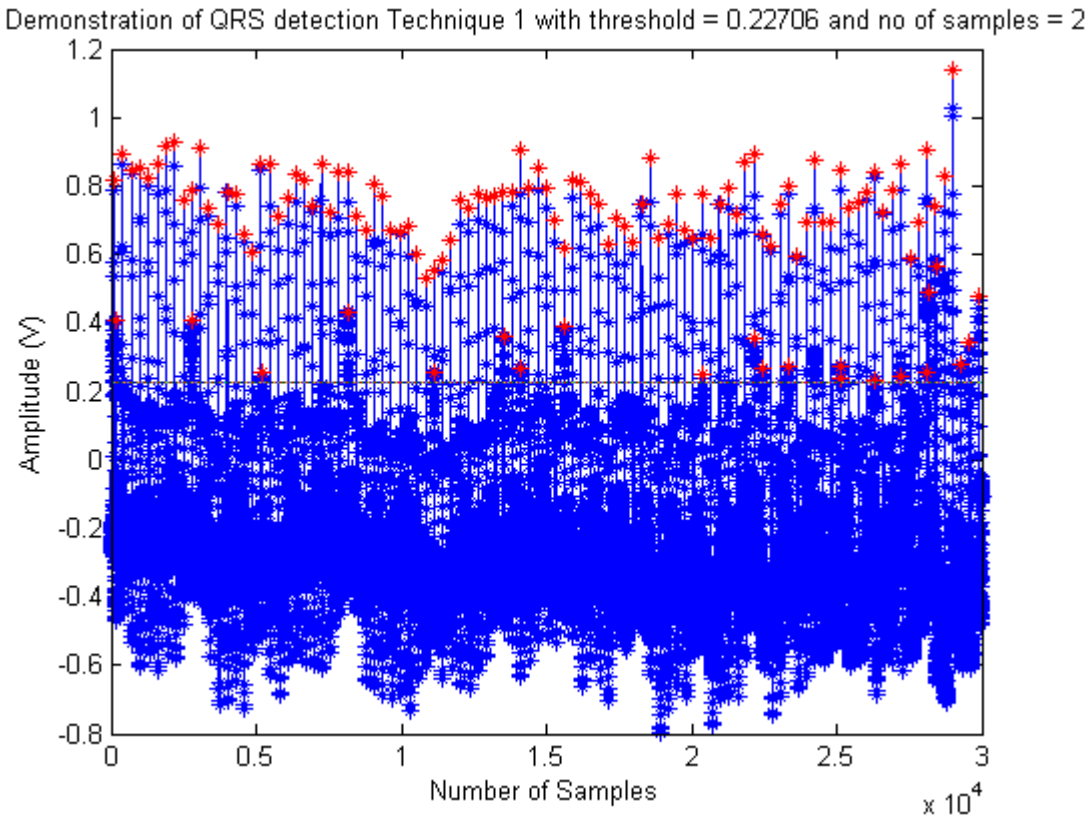
- $1^{η}$ τεχνική: ο αλγόριθμος της τεχνικής αυτής είναι προγραμματιστικά απλούστερος σε σχέση με του άλλους δύο. Ωστόσο έχει πολύ καλή απόδοση στην ανίχνευση των συμπλεγμάτων QRS. Το συμπέρασμα αυτό προκύπτει από το γεγονός ότι υπήρχαν συνδυασμοί τιμών των συντελεστών που έδωσαν πολύ καλά αποτελέσματα για όλα τα σήματα.

- $2^{η}$ τεχνική: η τεχνική αυτή βασίζεται σε έναν πιο πολύπλοκο αλγόριθμο. Όπως προκύπτει από τις δοκιμές έχει καλή απόδοση σε σήματα που παρουσιάζουν ιδιομορφίες, λόγω της προσαρμοστικότητας του κατωφλίου. Παρόλα αυτά, δεν υπάρχουν αρκετοί συνδυασμοί τιμών των συντελεστών που να έδωσαν καλά αποτελέσματα για όλα τα σήματα. Αυτό μπορεί να οφείλεται στο γεγονός ότι κατά το φιλτράρισμα τα χαρακτηριστικά των σημάτων αλλοιώνονται αρκετά.
- $3^{η}$ τεχνική: και η τεχνική αυτή παρουσιάζει περίπλοκο αλγόριθμο, καθώς η ανίχνευση των κορυφών προκύπτει με χρήση παραγώγων. Παρόλο που η απόδοσή της είναι συνολικά καλή, η πολυπλοκότητα του αλγορίθμου της την καθιστά προγραμματιστικά ασύμφορη, καθώς φέρει μεγάλο υπολογιστικό φορτίο.

Ο βασικός λόγος που τα αποτελέσματα δεν ήταν καλά σε όλες τις δοκιμές, ήταν ο συνδυασμός τιμών κατωφλίου και αριθμού δειγμάτων. Πιο συγκεκριμένα:

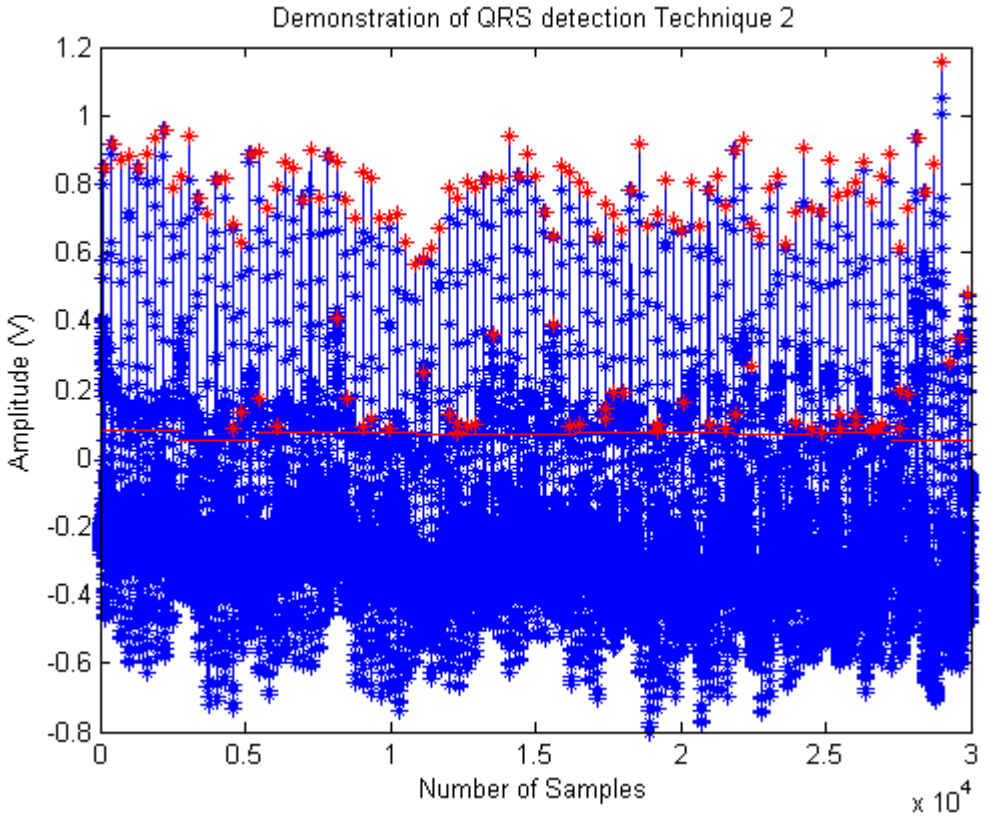
Α) Χαμηλή τιμή κατωφλίου – μικρός αριθμός δειγμάτων. Λόγω αυτού του συνδυασμού, ο αλγόριθμος ανιχνεύει κορυφές που δεν αντιστοιχούν σε πραγματικά συμπλέγματα QRS. Παρατίθενται παραδείγματα και για τις τρεις τεχνικές

1Η ΤΕΧΝΙΚΗ



Εικόνα 80 Τεχνική 1: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

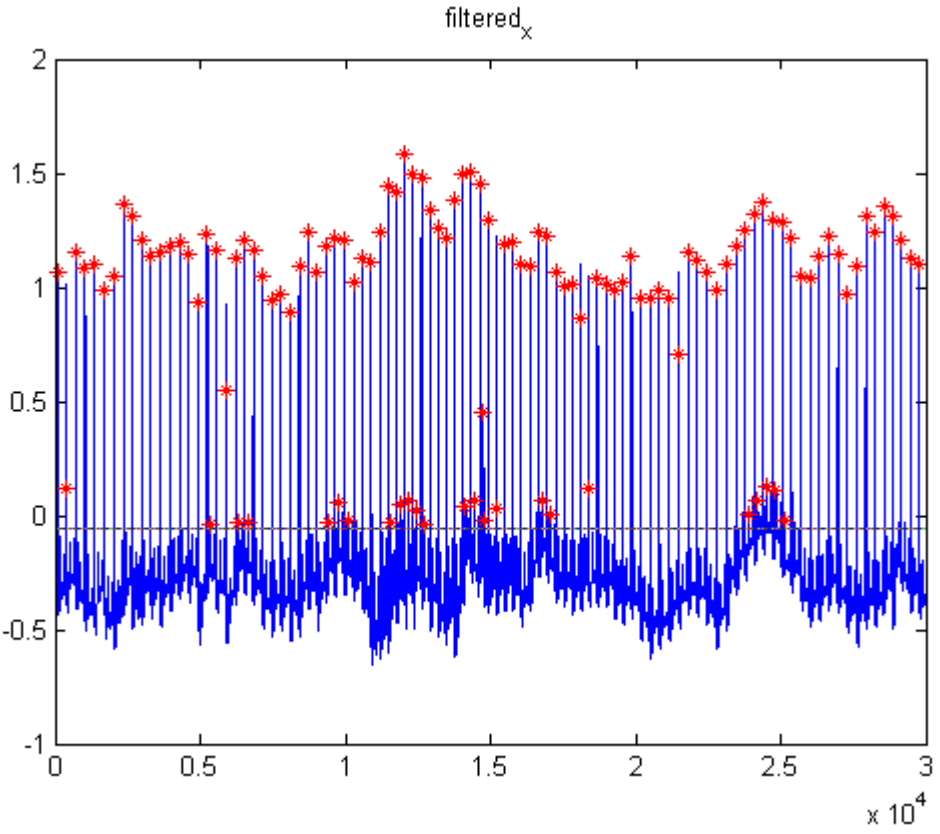
2Η ΤΕΧΝΙΚΗ



Εικόνα 81 Τεχνική 2: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

3^{Η} ΤΕΧΝΙΚΗ.

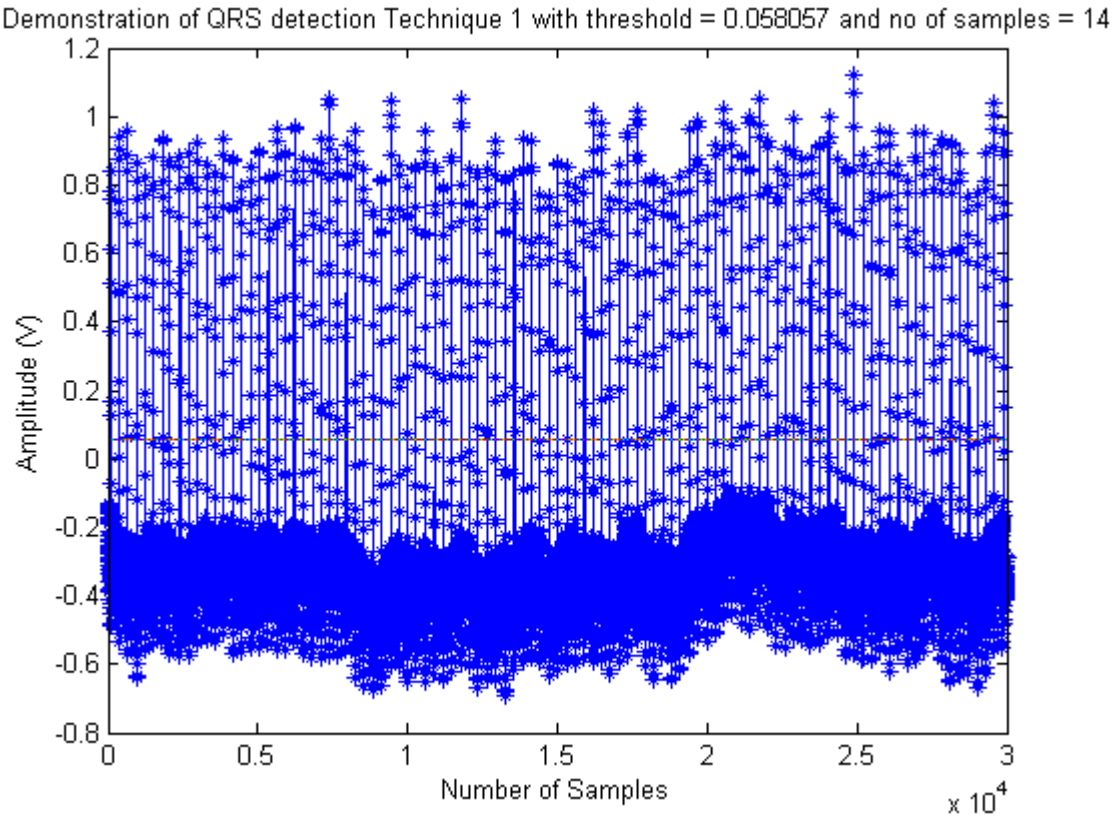
Η τεχνική αυτή ανιχνεύει τις κορυφές με βάση την αλλαγή προσήμου της παραγώγου του σήματος. Συνεπώς, μόνο η τιμή του κατωφλίου επηρεάζει τα αποτελέσματα.



Εικόνα 82 Τεχνική 3: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με τη μαύρη γραμμή

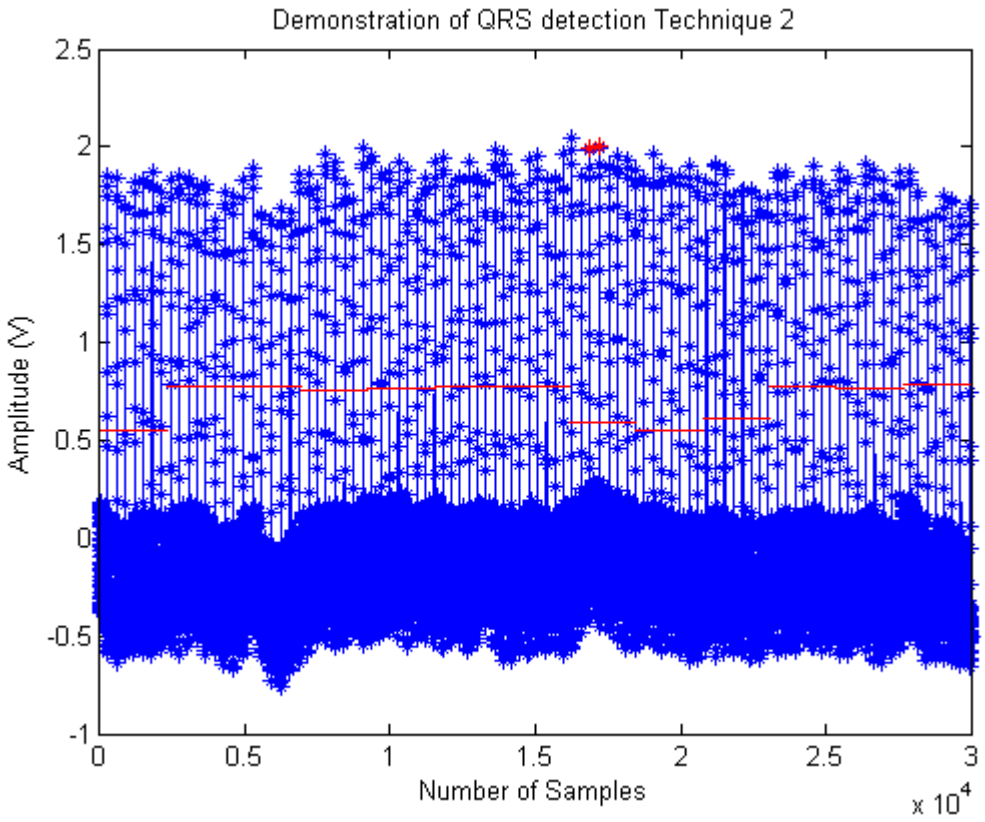
Β) Υψηλή τιμή κατωφλίου – μικρός αριθμός δειγμάτων. Στην περίπτωση αυτή, ο αλγόριθμος δεν εντοπίζει κορυφές που αντιστοιχούν σε συμπλέγματα QRS.

1Η ΤΕΧΝΙΚΗ



Εικόνα 83 Τεχνική 1: Τα δείγματα φαίνονται με μπλε αστερίσκο και το κατώφλι με τη μαύρη γραμμή

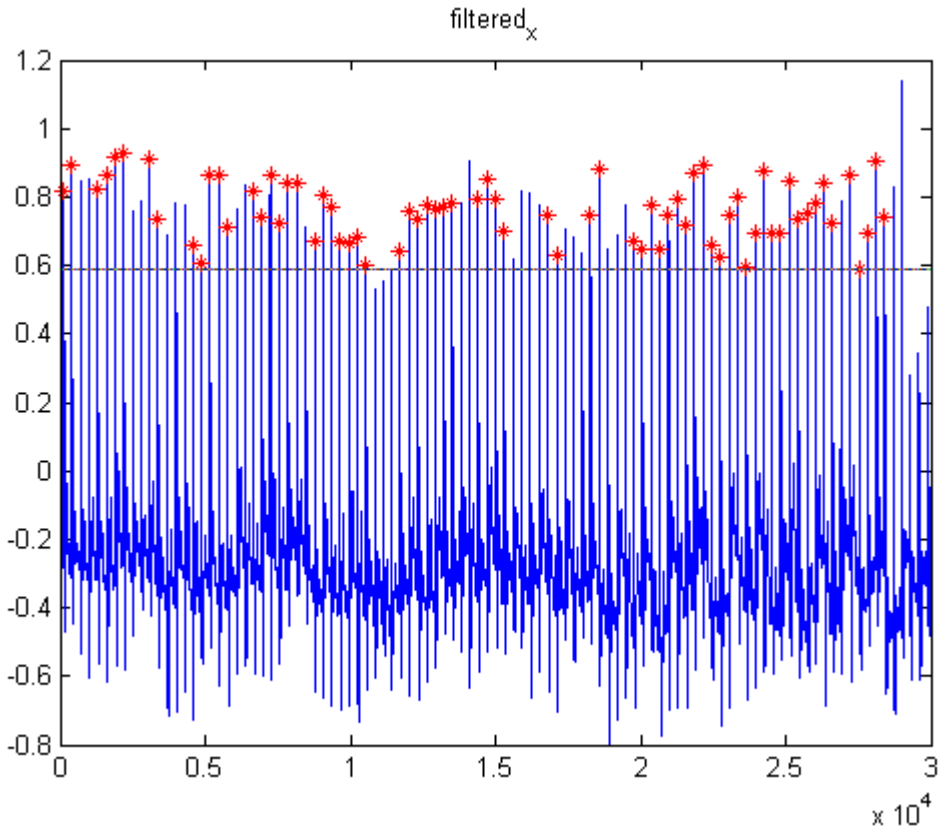
2^{H} ΤΕΧΝΙΚΗ



Εικόνα 84 Τεχνική 2: Τα δείγματα φαίνονται με μπλε αστερίσκο, τα ανιχνευμένα peaks με κόκκινο και το κατώφλι με την κόκκινη γραμμή

3Η ΤΕΧΝΙΚΗ

Και εδώ, τα αποτελέσματα εξαρτώνται μόνο από την τιμή του κατωφλίου



Εικόνα 85 Τεχνική 3: Τα ανιχνευμένα peaks φαίνονται με κόκκινο αστερίσκο και το κατώφλι με τη μαύρη γραμμή

Κεφάλαιο 5.

ΥΛΟΠΟΙΗΣΗ ΤΕΧΝΙΚΗΣ ΑΝΙΧΝΕΥΣΗΣ ΣΥΜΠΛΕΓΜΑΤΟΣ QRS ΣΕ ΕΠΙΠΕΔΟ ΔΙΚΤΥΟΥ ΑΣΥΡΜΑΤΩΝ ΑΙΣΘΗΤΗΡΩΝ.

Στο προηγούμενο κεφάλαιο μελετήθηκαν, υλοποιήθηκαν και αξιολογήθηκαν τρεις τεχνικές ανίχνευσης συμπλέγματος QRS στο περιβάλλον του MATLAB. Το κεφάλαιο αυτό πραγματεύεται την υλοποίηση τους με χρήση της πλατφόρμας tmote σε περιβάλλον TinyOS. Η πλατφόρμα αυτή έχει συγκεκριμένες προδιαγραφές, που δεν επιτρέπουν την υλοποίηση αλγορίθμου μεγάλου υπολογιστικού κόστους. Ως εκ τούτου, απορρίφθηκε η υλοποίηση των τεχνικών 2 και 3, καθώς είναι πολύπλοκες αλγοριθμικά. Η τεχνική 1 συνδυάζει αποδοτικότητα και προγραμματιστική απλότητα. Για το λόγο αυτό επιλέγεται η συγκεκριμένη τεχνική για υλοποίηση σε επίπεδο δικτύου ασύρματων αισθητήρων.

5.1 Περιγραφή του υλικού της εφαρμογής – σύνδεση και επικοινωνία των επιμέρους τμημάτων

Η εφαρμογή που υλοποιήσαμε αποτελείται από τα παρακάτω τμήματα:

5.1.1 Τμήμα συλλογής σήματος

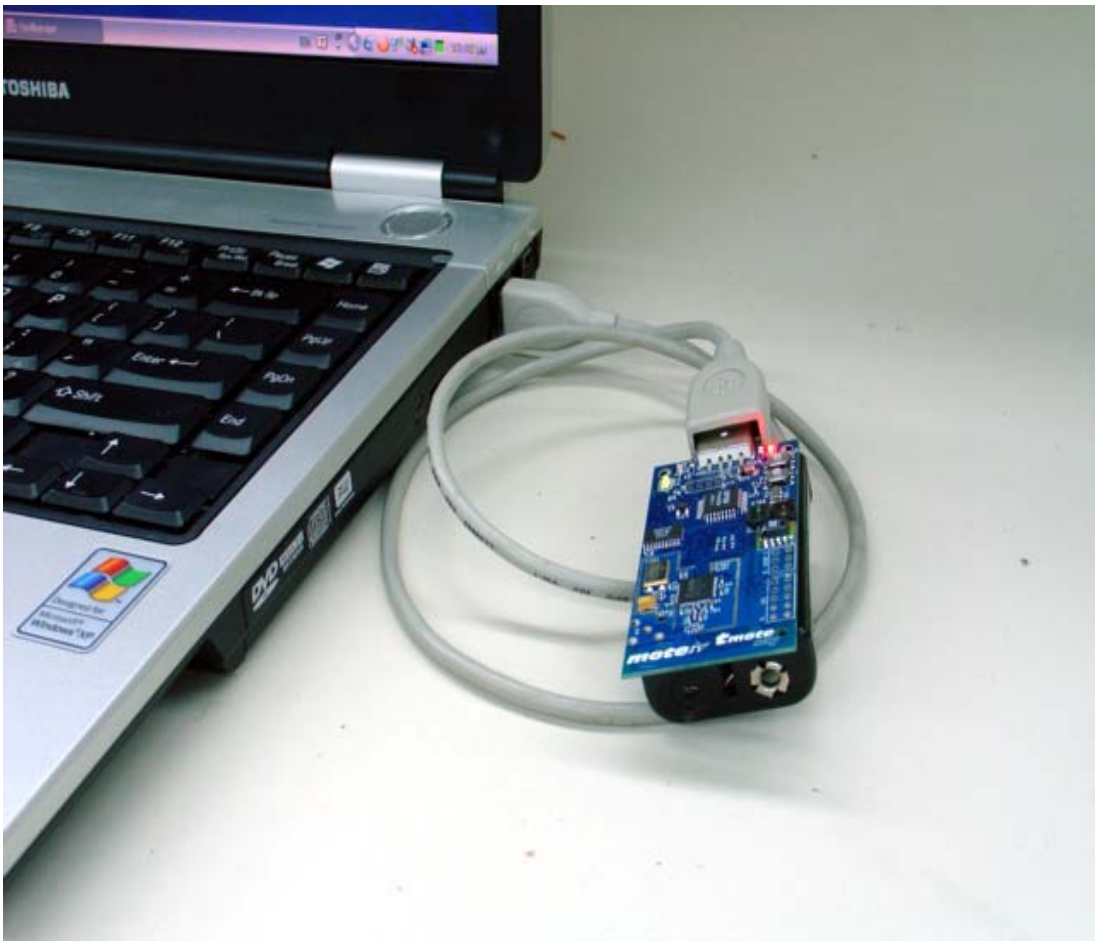
Το σήμα που μελετάμε είναι ΗΚΓ, που συλλέγεται από ηλεκτροκαρδιογράφο. Η έξοδος του ηλεκτροκαρδιογράφου συνδέεται στην είσοδο του tmote. Για να πραγματοποιήσουμε δοκιμές, χρησιμοποιήσαμε σήματα προερχόμενα από γεννήτρια. Η έξοδος της οποίας συνδέεται επίσης στην είσοδο του tmote. Τα σήματα αυτά είναι τόσο DC σήματα όσο και ημιτονοειδή διαφόρων συχνοτήτων και πλατών, όπως διαπιστώνεται με τη βοήθεια ενός παλμογράφου.

5.1.2 Τμήμα επεξεργασίας και αποστολής σήματος – μονάδα Tmote Sky

Το τμήμα αυτό αποτελείται από μια μονάδα Tmote Sky που συνδέεται με την έξοδο της γεννήτριας και αναλαμβάνει την αποστολή του σήματος ασύρματα. Τα χαρακτηριστικά της μονάδας tmote έχουν περιγραφεί αναλυτικά σε προηγούμενο κεφάλαιο. Η τροφοδότηση της μονάδας tmote γίνεται μέσω 2 αλκαλικών μπαταριών 1,5 V τύπου AA. Η μονάδα αυτή διαθέτει, όπως είδαμε, και ένα πλήθος pins εισόδου/εξόδου (16 pins σύνολο), τα οποία μπορούν να χρησιμοποιηθούν ως αναλογικές είσοδοι/έξοδοι για τη σύνδεση πολλών ακόμα συσκευών. Στην περίπτωση μας το σήμα τοποθετήθηκε στο κατάλληλο pin εισόδου του ADC (pin3 - analog input 0 – ADC0). Το σήμα αυτό δειγματοληπτείται με κατάλληλο ρυθμό, περνά ένα στάδιο επεξεργασίας, μέσω της τεχνικής που περιγράφεται παρακάτω και στη συνέχεια τα δείγματα αυτά αποστέλλονται ασύρματα, μέσα σε πακέτα δεδομένων, σε κάποιο σταθμό βάσης. Ο προγραμματισμός του tmote ώστε να εκτελεί την παραπάνω λειτουργία έγινε με την εγκατάσταση της εφαρμογής EcgSense που υλοποιήσαμε γι' αυτό το σκοπό.

5.1.3 Τμήμα λήψης σήματος

Στη συνέχεια στην εφαρμογή μας εμφανίζεται μια μονάδα Tmote Sky που είναι συνδεδεμένη με ένα φορητό υπολογιστή μέσω της θύρας USB. Η μονάδα είναι προγραμματισμένη με την εφαρμογή TOSBase, η οποία έχει ως λειτουργία να λαμβάνει τα πακέτα που στέλνονται ασύρματα από την προηγούμενη μονάδα tmote και να τα προωθεί στη σειριακή θύρα του υπολογιστή. Η απεικόνιση και επεξεργασία των σημάτων που λαμβάνουμε γίνεται στον υπολογιστή με τη βοήθεια των εργαλείων που θα περιγράψουμε στη συνέχεια. Τέλος, η τροφοδότηση της συσκευής γίνεται μέσω της θύρας USB (3V) χωρίς την ανάγκη χρήσης επιπλέον μπαταριών.



Εικόνα 86 Το τμήμα λήψης της εφαρμογής Μονάδα Tmote-Sky συνδεδεμένη με τον υπολογιστή

5.2 Περιγραφή του κώδικα της εφαρμογής

5.2.1 Μονάδα tmote συνδεδεμένη με τη γεννήτρια

Ο κώδικας που υλοποιήθηκε για το tmote που είναι συνδεδεμένο στη γεννήτρια στηρίζεται πάνω σε μια βασική εφαρμογή των δικτύων αισθητήρων. Αυτή της περιοδικής δειγματοληψίας ενός αισθητήρα και της αποστολής των τιμών σε ένα σταθμό βάσης. Το σήμα αυτό, όπως αναφέρθηκε και προηγουμένως, τοποθετείται στο κατάλληλο pin εισόδου του Tmote Sky (pin3 - analog input 0 – ADC0), δειγματοληπτείται με τον κατάλληλο ρυθμό και στη συνέχεια αποστέλλεται ασύρματα σε ένα ή περισσότερα tmotes.

Τον κύριο κώδικα της εφαρμογής μας τον ονομάσαμε EcgSence. Όπως περιγράψαμε στο δεύτερο κεφάλαιο, κάθε εφαρμογή αποτελείται από δυο στοιχεία, ένα module, όπου περιγράφεται το κυρίως σώμα της εφαρμογής, και ένα configuration όπου γίνεται η διασύνδεση των επιμέρους τμημάτων μεταξύ τους. Τα βασικά στοιχεία που χρησιμοποιήθηκαν στην εφαρμογή και περιγράφονται και στο configuration αρχείο (EcgSence.nc) είναι τα Main, TimerC για την υλοποίηση του timer, GenericComm για την υλοποίηση της διεπαφής που αναλαμβάνει την ασύρματη επικοινωνία, LedsC για την πρόσβαση στα leds, ADCC για την πρόσβαση στον AD μετατροπέα καθώς και το στοιχείο OscopeC το οποίο παρέχει τη διεπαφή Oscope της οποίας τη χρήση θα περιγράψουμε στη συνέχεια. Το Main είναι το στοιχείο που εκτελείται πρώτο σε μια εφαρμογή TinyOS. Δηλαδή η εντολή Main.StdControl.init() είναι η πρώτη εντολή που εκτελείται στο TinyOS ακολουθούμενη από την Main.StdControl.start(), όπου η StdControl είναι η διεπαφή που χρησιμοποιείται κοινώς για την αρχικοποίηση και την έναρξη εκτέλεσης των διαφόρων στοιχείων του TinyOS.

5.2.1.1 Module EcgSenseM

Το στοιχείο EcgSenseM είναι το κυρίως σώμα της εφαρμογής μας. Υλοποιεί τη διεπαφή StdControl και χρησιμοποιεί τις διεπαφές Timer, Leds, ADC, ADCControl και Oscope. Αυτό σημαίνει ότι μπορεί να καλεί κάθε εντολή που δηλώνεται σε αυτές τις διεπαφές και επίσης ότι πρέπει να υλοποιεί τα συμβάντα που σηματοδοτούνται (signaled) από αυτές. Στην τελευταία διεπαφή (Oscope) δίνουμε το στιγμιότυπο όνομα (*interface instance*) OEcgSense υλοποιώντας τη δυνατότητα που μας δίνει το TinyOS για χρήση μιας διεπαφής περισσότερες από μια φορές.

Η πορεία που ακολουθείται στον κώδικα EcgSenseM είναι η εξής: Αφού γίνουν οι κατάλληλες αρχικοποιήσεις των στοιχείων καλείται ο Timer με επαναλαμβανόμενο τρόπο (TIMER_REPEAT) και με χρονικό διάστημα που ορίζεται και αντιστοιχεί στην επιθυμητή δειγματοληψία. Ο Timer υλοποιεί μια λειτουργία που εκτελείται σε φάσεις και το τέλος της σηματοδοτείται από το συμβάν fired(), το οποίο προκύπτει ως απάντηση στο Timer. Κάθε φορά που το στοιχείο που υλοποιεί τον Timer σηματοδοτεί το συμβάν αυτό, η εφαρμογή καλεί την ADC.getdata() ώστε να γίνει δειγματοληψία στο κανάλι που έχει οριστεί.

Στη συνέχεια, καλείται το task «texniki1» στο οποίο γίνεται επεξεργασία του σήματος. Το task «texniki1» υλοποιεί την τεχνική σταθερού φίλτρου και κατωφλίου ($1^{η}$ τεχνική). Ωστόσο, ο κώδικας της τεχνικής τροποποιήθηκε κατά την υλοποίηση σε nesC σε σχέση με την υλοποίηση στο MATLAB, προκειμένου να είναι πιο αποδοτικός στο περιβάλλον του TinyOS. Όταν καλείται το task, τα δεδομένα που λαμβάνονται από τη δειγματοληψία τοποθετούνται στον πίνακα array[j]. Μόλις ο πίνακας συμπληρωθεί, υπολογίζονται η μέση τιμή (mean) και η τυπική απόκλιση (std) των στοιχείων του. Οι τιμές αυτές χρησιμοποιούνται στον καθορισμό του κατωφλίου μέσω του τύπου:

$$thres = mean + 2*std$$

Έπειτα, με τη χρήση ενός for loop, εντοπίζεται η μέγιστη τιμή του πρώτου μισού του πίνακα array[j]. Η τιμή αυτή, καθώς και οι γειτονικές της, συγκρίνονται με το κατώφλι και, εφόσον το ξεπερνούν, ανιχνεύεται ένα peak (σύμπλεγμα QRS). Η διαδικασία επαναλαμβάνεται και για το δεύτερο μισό του πίνακα. Η εκτέλεση του task σηματοδοτείται με άναμμα του κόκκινου led. Ο κώδικας της τεχνικής παρατίθεται στο Παράρτημα Β.

Κάθε φορά που εισέρχονται δεδομένα στο tmote σηματοδοτείται το συμβάν ADC.dataready(), η υλοποίηση του οποίου θέτει την εργασία putData(). Σε αυτό το σημείο πρέπει να σημειώσουμε ότι το συμβάν dataReady() εκτελείται ασύγχρονα και θέτει την εργασία (task) putData η οποία εκτελείται σύγχρονα. Η putData καλεί την εντολή OEcgSense.put() με όρισμα την τιμή που πήρε ο A/D μετατροπέας από την είσοδο.

Η εντολή put() ορίζεται στην διεπαφή Oscope (OEcgSense), και υλοποιείται στην εφαρμογή OscopeC. Ουσιαστικά η υλοποίηση της γίνεται στον κώδικα OscopeM, καθώς ο κώδικας OscopeC αποτελεί απλά το configuration της εφαρμογής Oscope μέσα στο οποίο δηλώνεται ότι παρέχεται η διεπαφή Oscope. Η εφαρμογή Oscope είναι μια ‘υπηρεσία’ που υπάρχει στις βιβλιοθήκες του TinyOS για την αποστολή ενός συνόλου δεδομένων στην μορφή εκείνη των πακέτων που μπορεί να αναγνωρίσει και να απεικονίσει γραφικά η java εφαρμογή oscilloscope (στην οποία θα αναφερθούμε στη συνέχεια). Η ‘υπηρεσία’ Oscope διαχειρίζεται κατάλληλα τη συλλογή και την ενδιάμεση αποθήκευση (buffering) των δεδομένων για την εφαρμογή μας και στη

συνέχεια, μόλις έχουν συγκεντρωθεί οι οριζόμενες συνεχόμενες τιμές μετρήσεων, τις αποστέλλει μέσα σε ένα Oscope μήνυμα. Η υπηρεσία αυτή υποστηρίζει μέχρι και δυο κανάλια 0 ή 1 (ή και περισσότερα με τροποποίηση του κώδικα), γι' αυτό και χρησιμοποιεί την έννοια της παραμετροθετημένης διεπαφής Oscope[uint8_t channel]. Στην περίπτωση της εφαρμογής μας έγινε χρήση μόνο του ενός καναλιού (channel 0). Τα πακέτα τα αποστέλλει κάνοντας χρήση της διεπαφής SendMsg η οποία και παρέχεται από το στοιχείο GenericComm. Η διαδικασία της αποστολής των μηνυμάτων μέσω του Radio σηματοδοτείται με άναμμα του πράσινου led. Ο κώδικας του OscopeM χρησιμοποιεί επίσης την τακτική για κλείδωμα της μνήμης, ώστε να μην είναι δυνατή η πρόσβαση και τροποποίησή της μέχρι την ολοκλήρωση της αποστολής του πακέτου.

Το OscopeMsg μήνυμα αποτελεί στη ουσία μια δομή για τον τύπο μηνυμάτων που στέλνουμε με την εφαρμογή μας. Η δομή αυτή ορίζεται σε ένα ξεχωριστό header αρχείο με την ονομασία Oscope.h και φαίνεται παρακάτω:

Αποτελείται όπως μπορούμε να διακρίνουμε από 4 πεδία και έχει μέγεθος 26 bytes.

```
enum
{
OSCOPE_BUFFER_SIZE = 10,
AM_OSCOPEMSG = 10,
AM_OSCOPERESETMSG = 32,
};
typedef struct OscopeMsg
{
uint16_t sourceMoteID;
uint16_t lastSampleNumber;
uint16_t channel;
uint16_t data[OSCOPE_BUFFER_SIZE];
} OscopeMsg_t;
```

Στο πεδίο data[OSCOPE_BUFFER_SIZE], το οποίο αποτελεί μια διάταξη με 10 θέσεις μνήμης των 2 bytes η καθεμιά, αποθηκεύονται τα δεδομένα που λαμβάνονται από τον ADC. Δηλαδή 10 μετρήσεις των 2 bytes η καθεμιά

Στο πεδίο sourceMoteID (2 bytes) αποθηκεύεται κάθε φορά η εσωτερική διεύθυνση της μονάδας tmote που έχει αναλάβει την αποστολή και προκύπτει από την εντολή TOS_LOCAL_ADDRESS

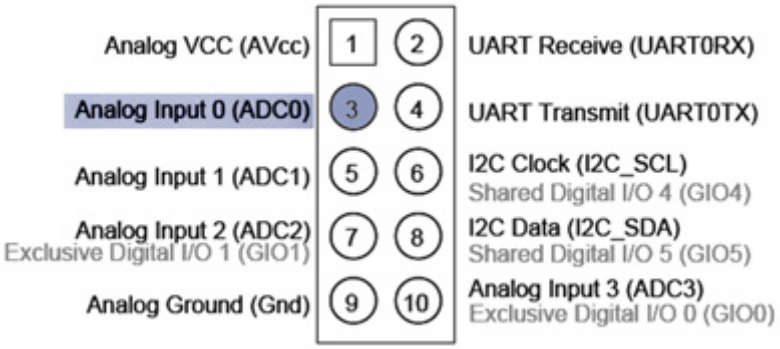
Το πεδίο lastSampleNumber (2 bytes) περιέχει τον αριθμό του πακέτου που αποστέλλεται

Τέλος, το πεδίο channel (2 bytes) αποθηκεύει κάθε φορά τον αριθμό του καναλιού που χρησιμοποιείται. Εμείς, χρησιμοποιήσαμε μόνο το κανάλι 0 στην εφαρμογή μας, οπότε αυτό το μέγεθος θα είναι πάντα σταθερό και ίσο με 0 (00 00)

Το μήνυμα Oscope (OscopeMsg) τοποθετείται στο πεδίο payload data της δομής πακέτου TOS_Msg, καθώς αυτή είναι η δομή πακέτων που αποστέλλεται μέσω του Send() στο TinyOS.

5.2.1.2 Σύνδεση εξωτερικών αισθητήρων με το tmote

Το Tmote Sky εκτός από τους εσωτερικούς αισθητήρες που διαθέτει, παρέχει τη δυνατότητα να συνδέσουμε ένα πλήθος εξωτερικών συσκευών-αισθητήρων με τον AD μετατροπέα του, μέσω των δυο συνδετήρων επέκτασης (expansion connectors) που διαθέτει, 10pin και 6pin αντίστοιχα. Στην περίπτωσή μας αυτό ήταν αναγκαίο. Συγκεκριμένα συνδέσαμε την έξοδο του τροφοδοτικού με την αναλογική είσοδο 0 ADC0 (pin3 πάνω στον 10pin connector) του AD μετατροπέα και τη γείωση του τροφοδοτικού με την αναλογική είσοδο GND.



Εικόνα 87 Ο συνδετήρας 10 pin του Tmote-Sky. Η έξοδος του τροφοδοτικού συνδέθηκε στην αναλογική είσοδο ADC0, η οποία και επισημαίνεται και η γείωση του τροφοδοτικού με την αναλογική είσοδο GND

Ο σωστός ορισμός της θύρας εισόδου όπου θα γίνει και η δειγματοληψία του σήματός μας προϋποθέτει μια σειρά από βήματα. Συγκεκριμένα:

1. Δημιουργήσαμε ένα αρχείο header με την ονομασία adc0.h μέσα στο οποίο προσδιορίσαμε τη θύρα ADC που χρησιμοποιούμε. Στο αρχείο αυτό, του οποίου ο

κώδικας παρατίθεται στο Παράρτημα Β στο τέλος της αναφοράς μας, ο προσδιορισμός αυτός γίνεται με τη βοήθεια των εντολών

```
TOS_ADC_ADC0_PORT=unique("ADCPort"),
TOSH_ACTUAL_ADC_ADC0_PORT=ASSOCIATE_ADC_CHANNEL()
```

Οι τρεις παράμετροι που περιγράφουν την εντολή ASSOCIATE_ADC_CHANNEL είναι:

• INPUT_CHANNEL_A0 όπου προσδιορίζουμε την είσοδο ADC0 που χρησιμοποιούμε.

• REFERENCE_VREFplus_AVss όπου προσδιορίζουμε τη χρήση της εσωτερικής τάσης αναφοράς.

• REFVOLT_LEVEL_2_5 όπου προσδιορίζουμε τη χρήση των 2.5V ως τιμή της εσωτερικής τάσης αναφοράς.

2. Στο σώμα της εφαρμογής μας χρησιμοποιήθηκε η διεπαφή ADC και επίσης η διεπαφή ADCControl για να εκτελεστεί το ‘binding’ της θύρας μας, με την εντολή ADCControl.bindPort(). Αυτό έγινε στη φάση της ‘αρχικοποίησης’ (‘StdControl.init()’). Από εκεί και πέρα κάθε φορά που θα καλούμε την ADC.getData(), ο AD μετατροπέας πηγαίνει και διαβάζει στη συγκεκριμένη πόρτα που του έχουμε ορίσει.

3. Το τελευταίο βήμα ήταν η διασύνδεση των διεπαφών ADC και ADCControl στο configuration της εφαρμογής μας, σύμφωνα με τις παρακάτω εντολές:

```
EcgSenseM.ADC -> ADCC.ADC[TOS_ADC_ADC0_PORT];
EcgSenseM.ADCControl -> ADCC;
```

5.2.1.3 Configuration EcgSense

Ο κώδικας EcgSense.nc αποτελεί το configuration της εφαρμογής μας και συγκεντρώνει τα επιμέρους στοιχεία μαζί, συνδέοντας διεπαφές που χρησιμοποιούνται από κάποια στοιχεία με διεπαφές που παρέχονται από άλλα στοιχεία. Ο κώδικας αυτός παρατίθεται στο Παράρτημα Β. Παρατηρούμε ότι το configuration παρέχει τη διεπαφή StdControl, της οποίας την υλοποίηση έχει αναλάβει το module EcgSenseM. Η γραμμή components προσδιορίζει το σύνολο των στοιχείων στα οποία αναφέρεται: Main, EcgSenseM, TimerC, GenericComm, LedsC, ADCC και OscopeC. Το στοιχείο Main είναι, όπως έχουμε πει, το πρώτο στοιχείο που εκτελείται σε μια εφαρμογή TinyOS.

Αρχικά, συνδέουμε τη διεπαφή StdControl που χρησιμοποιείται στη Main με αυτές των στοιχείων EcgSenseM, TimerC, GenericComm, ADCC και OscopeC, θυμίζοντας τη δυνατότητα που παρέχεται στο TinyOS να διασυνδέουμε μια διεπαφή με ένα πλήθος υλοποιήσεων αυτής (fan-out), οπότε και μια κλήση π.χ. της StdControl.start() μπορεί να είναι συνδεδεμένη με ένα αυθαίρετο πλήθος υλοποιήσεων της εντολής αυτής. Στη συνέχεια διασυνδέουμε τη διεπαφή OEcgSense που χρησιμοποιεί ο EcgSenseM με το στοιχείο OscopeC που την παρέχει. Η σύνδεση αυτή γίνεται μέσω μιας παραμετροθετημένης διεπαφής Oscope[0] και αυτό γιατί η διεπαφή Oscope μπορεί να υλοποιηθεί για περισσότερα από ένα κανάλια, ενώ εμείς που χρησιμοποιούμε το κανάλι 0 θέλουμε να συνδέσουμε την εφαρμογή μας μόνο με τη διεπαφή που αντιστοιχεί στο κανάλι μας.

Στη διασύνδεση με τον Timer, ο οποίος υλοποιείται από το στοιχείο TimerC, χρησιμοποιούμε επίσης παραμετροθετημένη διεπαφή. Αυτό γιατί έχουμε την ανάγκη ύπαρξης περισσότερων του ενός Timer, καθένας από τους οποίους δρα ανεξάρτητα και ικανοποιεί τις απαιτήσεις του κάθε στοιχείου. Παράλληλα, χρησιμοποιούμε τον όρο unique("Timer") σαν όρισμα της διεπαφής έτσι ώστε να έχουμε κάθε φορά ένα διαφορετικό αναγνωριστικό id από το σύνολο των δυνατών αναγνωριστικών που μπορεί να παρέχει ο Timer και που δεν χρησιμοποιούνται από άλλο στοιχείο.

Στο configuration γίνεται επίσης και η διασύνδεση της διεπαφής Leds με το στοιχείο LedsC που την παρέχει. Η διεπαφή αυτή χρησιμοποιείται από την εφαρμογή μας για τον έλεγχο των Leds. Τέλος γίνεται η διασύνδεση των διεπαφών ADC, ADCControl

που περιγράψαμε και παραπάνω μέσω χρήσης της παραμετροθετημένης διεπαφής ADC[TOS_ADC_ADC0_PORT], την οποία παρέχει ο ADCC, και όρισμα το μοναδικό αναγνωριστικό που αντιστοιχεί στην είσοδο ADC0 που ορίσαμε στο αρχείο adc0.h.

Μπορούμε να πούμε γενικά ότι το βέλος '→' στον κώδικα του configuration EcgSense δείχνει πάντα από το χρήστη μιας διεπαφής στο στοιχείο που παρέχει τη διεπαφή αυτή. Ο κώδικας των στοιχείων που περιγράψαμε παραπάνω παρατίθεται στο Παράρτημα Β, στο τέλος της εργασίας.

5.2.2 Μονάδα tmote συνδεδεμένη με τον υπολογιστή

Η μονάδα tmote που είναι συνδεδεμένη με τον υπολογιστή λαμβάνει τα δεδομένα που στέλνουμε και τα προωθεί σε αυτόν για απεικόνιση, αποθήκευση και επεξεργασία. Για αυτή τη λειτουργία χρησιμοποιήσαμε την εφαρμογή TOSBase που υπάρχει στις βιβλιοθήκες προγραμμάτων του TinyOS. Η εφαρμογή TOSBase δρα κατά κάποιο τρόπο ως γέφυρα μεταξύ της ασύρματης και σειριακής σύνδεσης. Τα μηνύματα που διακινούνται από τη σειριακή προς την ασύρματη σύνδεση παίρνουν την ετικέτα group ID που πήρε η εφαρμογή TOSBase κατά τη μεταγλώττιση της, ενώ τα μηνύματα που φτάνουν ασύρματα 'φιλτράρονται' επίσης μέσα από το ίδιο group ID πριν προωθηθούν στη σειριακή θύρα.

Η εφαρμογή TOSBase ενσωματώνει ουρές δεδομένων και προς τις δυο κατευθύνσεις, εγγυώμενη ότι εάν ένα πακέτο εισέλθει στην ουρά τότε θα εξέλθει στο τέλος σίγουρα προς την άλλη διεπαφή. Αυτές οι ουρές επιτρέπουν στην εφαρμογή να χειρίζεται με μεγαλύτερη αξιοπιστία τυχόν αιχμές δεδομένων. Τα Leds είναι προγραμματισμένα να αναβοσβήνουν στις ακόλουθες περιπτώσεις:

Κόκκινο led: Το μήνυμα προωθείται από τη σειριακή σύνδεση στην ασύρματη.

Πράσινο led: Το μήνυμα λαμβάνεται ασύρματα και προωθείται στη σειριακή σύνδεση (κάτι το οποίο συμβαίνει στην περίπτωση μας).

Κίτρινο led: ένα ή περισσότερα πακέτα απορρίφθηκαν λόγω υπερχείλισης της ουράς σε μια από τις δυο κατευθύνσεις.

Η εφαρμογή TOSBase μας επιτρέπει, με τη βοήθεια των εργαλείων που θα περιγράψουμε και στη συνέχεια, να απεικονίζουμε εύκολα τα δεδομένα μας, να τα

αποθηκεύουμε και επίσης να τα προωθούμε σε άλλες εφαρμογές προς περαιτέρω επεξεργασία.

5.3 Μεταγλώττιση προγραμμάτων και εγκατάσταση τους

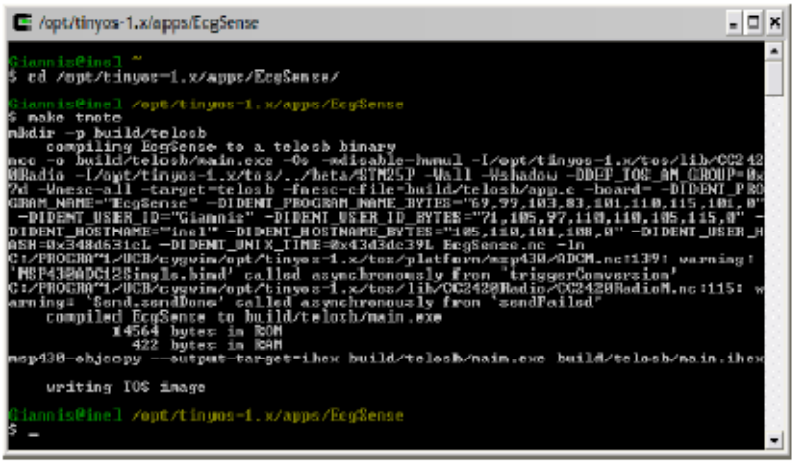
Για τη δημιουργία του περιβάλλοντος προγραμματισμού και ανάπτυξης εφαρμογών του Tmote ήταν απαραίτητη η εγκατάσταση των εργαλείων και προγραμμάτων.

Η εγκατάσταση αυτή περιλαμβάνει τα cygwin, Java, τον MSPGCC MSP430 μεταγλωττιστή, τους οδηγούς USB και άλλα προγράμματα υποστήριξης καθώς και εφαρμογές γραμμένες σε κώδικα NesC.

Ο cygwin αποτελεί ένα περιβάλλον εξομοίωσης Linux για Windows που παρέχει το βασικό περιβάλλον ανάπτυξης για το TinyOS και το Tmote Sky. Η java χρησιμοποιείται για τα εργαλεία που διασυνδέουν το Tmote Sky με τον Η/Υ. Ο MSPGCC είναι ο μεταγλωττιστής (compiler) που χρησιμοποιείται από το TinyOS για την πλατφόρμα του μικροελεγκτή TI MSP430.

Ο cygwin χρησιμοποιείται για τη μεταγλώττιση και τον προγραμματισμό εφαρμογών για το Tmote Sky. Απαραίτητη προϋπόθεση για να προγραμματίσουμε μια μονάδα tmote με την εφαρμογή που επιθυμούμε είναι η σύνδεση του με τη θύρα USB του υπολογιστή. Έτσι, στην περίπτωση μας, αφού συνδέσαμε το αντίστοιχο tmote, η μεταγλώττιση του EcgSence έγινε εύκολα πηγαίνοντας, μέσα από το παράθυρο εντολών του Cygwin, στο φάκελο που περιέχει την εφαρμογή μας και εκτελώντας την εντολή:

```
‘make tmote’
```

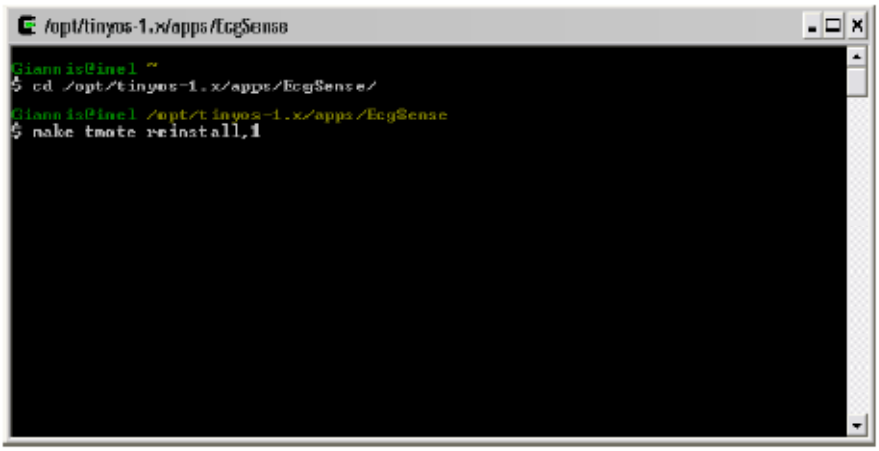


Εικόνα 88 Παράθυρο εντολών του cygwin - Μεταγλώττιση του κώδικα της εφαρμογής EcgSense

Στη συνέχεια, για να προγραμματίσουμε το tmote με την συγκεκριμένη εφαρμογή πληκτρολογήσαμε την εντολή

```
‘make tmote reinstall, 1’
```

Στην παραπάνω εντολή το ‘reinstall,1’ σημαίνει ότι προγραμματίζουμε τη μονάδα μας με το ήδη μεταγλωττισμένο δυαδικό αρχείο και θέτουμε τη διεύθυνση δικτύου σε 1.



Εικόνα 89 Παράθυρο εντολών του cygwin - Προγραμματισμός του Tmote με τον κώδικα της εφαρμογής EcgSense

5.4 Απεικόνιση δεδομένων στον υπολογιστή

Η απεικόνιση των δεδομένων στον υπολογιστή έγινε με τη χρήση συγκεκριμένων εργαλείων σε Java που μας παρέχει το TinyOS. Τα εργαλεία αυτά μας βοηθούν τόσο στο να απεικονίσουμε τα πακέτα στη μορφή 'raw data' με την οποία καταφθάνουν στο tmote, όσο και να παραστήσουμε γραφικά τις τιμές των μετρήσεων που πήραμε από τον ηλεκτροκαρδιογράφο.

5.4.1 Το εργαλείο Listen – Μορφή πακέτων raw data

Το εργαλείο Listen χρησιμοποιείται για να απεικονίσουμε στη οθόνη τα πακέτα raw data όπως αυτά καταφθάνουν στη σειριακή θύρα του υπολογιστή μέσω της εφαρμογής TOSBase. Με την πληκτρολόγηση της εντολής:

```
export MOTECOM=serial@serialport:tmote
```

όπου serialport είναι η σειριακή θύρα στην οποία έχουμε συνδέσει το Tmote Sky στον υπολογιστή, καθοδηγούμε το εργαλείο Listen ώστε να ακούει το συγκεκριμένο tmote. Η θύρα αυτή μπορεί εύκολα να βρεθεί εκτελώντας την εντολή 'motelist'.

Εκτελώντας στη συνέχεια την εντολή

```
'java net.TinyOS.tools.Listen'
```

παρατηρούμε στην οθόνη μια συνεχόμενη ροή δεδομένων που αντιστοιχούν στα πακέτα 'raw data' που λαμβάνουμε:

```
% java net.tinyos.tools.Listen
serial@COM11:57600: resynchronising
1A 01 08 B2 FF FF FF FF 0A 7D 03 00 F4 76 00 00 47 08 0F 08 7B 08 BE
08 CA 08 EA 07 52 09 C7 08 2B 08 6F 08
1A 01 08 B3 FF FF FF FF 0A 7D 03 00 FE 76 00 00 5F 08 BE 08 AE 08 93
08 66 09 72 08 D7 08 5B 09 32 08 5E 08
1A 01 08 B4 FF FF FF FF 0A 7D 03 00 08 77 00 00 E6 08 77 08 9E 08 D6
08 23 08 97 08 77 08 3A 09 1A 08 F3 07
```

Εικόνα 90 Ροή δεδομένων raw data

Επειδή η συχνότητα δειγματοληψίας που έχουμε ορίσει στην εφαρμογή μας οδηγεί σε μια συνεχόμενη ροή δεδομένων που είναι δύσκολο να παρατηρηθούν και να μελετηθούν ταυτόχρονα στην οθόνη, τα αποθηκεύουμε σε ένα αρχείο.*txt*.

Κάθε πακέτο που φθάνει από το tmote αποτελείται από ένα σύνολο πεδίων. Κάποια από αυτά αντιστοιχούν στα αντίστοιχα πεδία που ορίζονται στη δομή του πακέτου TOS_Msg, ενώ στο πεδίο 'data payload' του μηνύματος περιέχεται το μήνυμα Oscope που ορίσαμε παραπάνω και που η δομή του περιγράφεται στο header αρχείο Oscope.h. Πρέπει εδώ να αναφέρουμε ξανά ότι τα δεδομένα μέσα στα πακέτα παριστάνονται σε μορφή little-endian, οπότε και μια τιμή '6F 08' (2 bytes) παριστάνει μια μέτρηση με το περισσότερο σημαντικό στοιχείο 0x08 και το λιγότερο σημαντικό ψηφίο 0x6F, το οποίο τελικά μας δίνει την τιμή 0x086F ή 2159 σε δεκαδική μορφή.

Η μορφή του πακέτου TOS_Msg που στέλνεται ασύρματα αποτελείται από τα επόμενα πεδία:

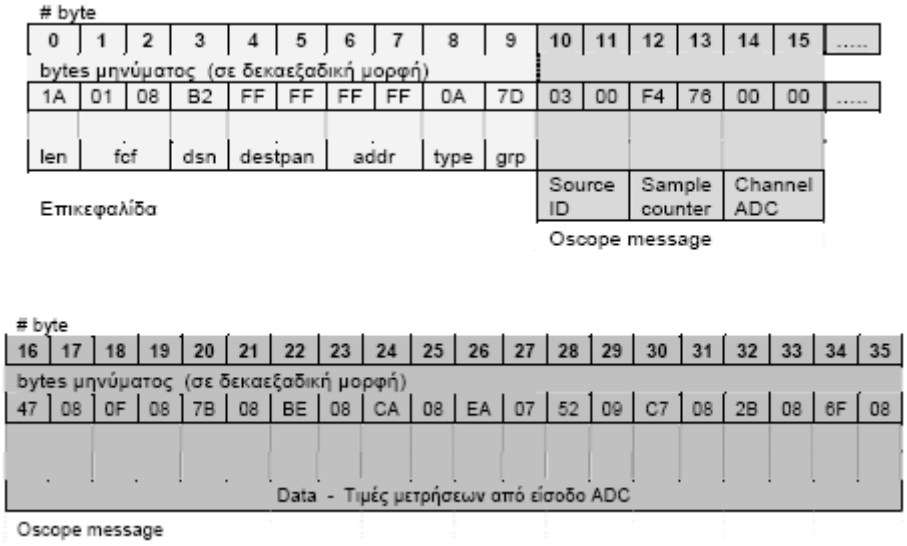
- length (1byte): μήκος μηνύματος, χωρίς να συμπεριλαμβάνεται η επικεφαλίδα
- fcf -frame control field- (2 bytes): πεδίο ελέγχου πλαισίου IEEE 802.15.4 (δεσμευμένο)
- dsn – data sequence number (1 byte): αριθμός ακολουθίας δεδομένων (δεσμευμένο- IEEE 802.15.4)
- destpan – destination personal area network id (2 bytes): αναγνωριστικό id δικτύου προορισμού (δεσμευμένο - IEEE 802.15.4)
- addr (2 bytes): διεύθυνση προορισμού (TinyOS)
- type (1 byte): αναγνωριστικός αριθμός AM για τον τύπο μηνύματος TinyOS
- group id (1 byte): αναγνωριστικός αριθμός για τον γκρουπ των κόμβων που συμμετέχουν στο δίκτυο
- payload data (μέχρι 29bytes): πεδίο χρήσιμων δεδομένων μηνύματος.
    - source mote ID (2 bytes): διεύθυνση μονάδας που στέλνει το μήνυμα.
    - sample counter (2 bytes): αριθμός δείγματος που αποστέλλεται.
    - ADC channel (2 bytes): αριθμός καναλιού που χρησιμοποιείται.
    - ADC data readings (10 μετρήσεις των 2 bytes η καθεμιά).

Έστω ότι έχουμε το πακέτο:

```
1A 01 08 B2 FF FF FF FF 0A 7D 03 00 F4 76 00 00 47 08 0F 08 7B 08 BE
08 CA 08 EA 07 52 09 C7 08 2B 08 6F 08
```

Εικόνα 91 Πακέτο δεδομένων raw data

Αν θελήσουμε να αντιστοιχίσουμε τα δεδομένα του παραπάνω πακέτου στα αντίστοιχα πεδία, τότε προκύπτουν τα εξής:



Εικόνα 92 Δομή πακέτου TOS_Msg για το Tmote-Sky

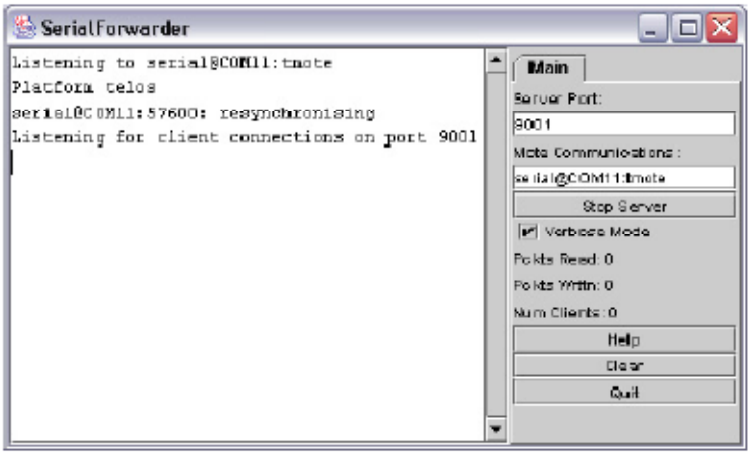
5.4.2 Η εφαρμογή SerialForwarder

Με το εργαλείο Listen δεν μπορούμε να απεικονίσουμε άμεσα και εύκολα τα δεδομένα που έρχονται μέσω της σειριακής θύρας του υπολογιστή. Για το λόγο αυτό, υπάρχουν μερικά άλλα εργαλεία που μπορούν να βοηθήσουν προς την κατεύθυνση αυτή. 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 εργαλείο SerialForwarder τρέχει εισάγοντας την παρακάτω εντολή στο παράθυρο εντολών του Cygwin, έχοντας βέβαια αρχικά συνδέσει μια μονάδα tmote στη θύρα USB του υπολογιστή:

```
java net.TinyOS.sf.SerialForwarder -comm
serial@COMx:tmote
```

όπου x στο COMx εισάγουμε τον αριθμό της θύρας με την οποία έχει αναγνωριστεί η μονάδα μας μετά τη σύνδεση με τον υπολογιστή. Ο αριθμός αυτός διαπιστώνεται εύκολα τρέχοντας την εντολή 'motelist' η οποία επιστρέφει τις ενεργές θύρες COM στις οποίες είναι συνδεδεμένες μονάδες tmote με τον Η/Υ.

Εκτελώντας αυτή την εντολή στην περίπτωσή μας (COM11) ανοίγει το παράθυρο του SerialForwarder:



Εικόνα 93 Το παράθυρο του SerialForwarder

Το όρισμα –comm θέτει τον SerialForwarder να επικοινωνήσει με τη θύρα COM11. Προσδιορίζει επίσης την προέλευση των πακέτων που προωθούνται μέσω του εργαλείου αυτού, χρησιμοποιώντας την ίδια σύνταξη όπως και η μεταβλητή MOTECOM που είδαμε προηγουμένως. Σε αντίθεση όμως με τα περισσότερα προγράμματα, ο SerialForwarder δεν κάνει χρήση της μεταβλητής MOTECOM αλλά χρησιμοποιεί το όρισμα –comm για την πηγή προέλευσης του πακέτου.

Το όρισμα 'tmote' θέτει τον SerialForwarder να επικοινωνήσει με την ταχύτητα που υποστηρίζει η πλατφόρμα tmote, δηλαδή στην περίπτωσή μας τον ρυθμό 57600 baud.

Ο SerialForwarder δεν απεικονίζει το ίδιο το πακέτο, αλλά ανανεώνει τον μετρητή πακέτων στην δεξιά γωνία του παραθύρου. Καθώς τρέχει, ακούει για συνδέσεις πελάτη δικτύου σε μία δοσμένη θύρα (port) – 9001 είναι η προκαθορισμένη θύρα –

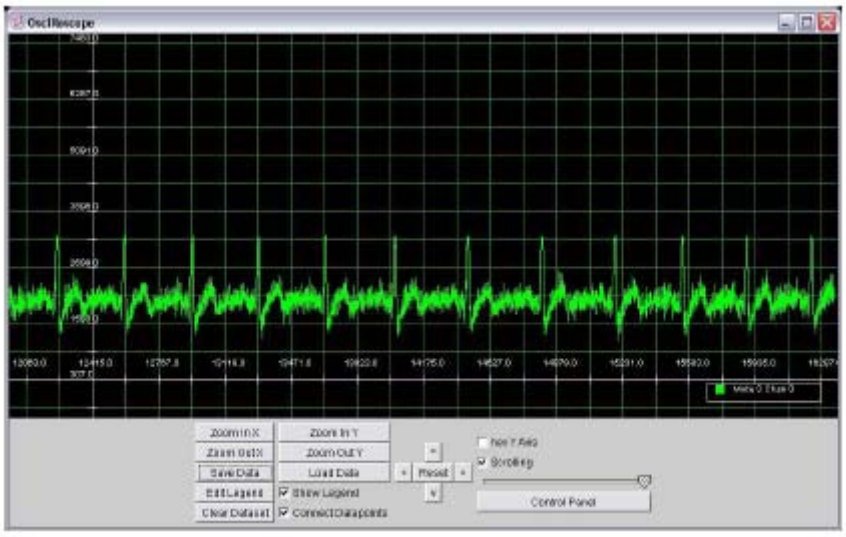
και απλά προωθεί πακέτα TinyOS από τη σειριακή θύρα στη σύνδεση πελάτη και αντίστροφα. Εδώ αξίζει να αναφέρουμε ότι, στον SerialForwarder, μπορούν να συνδεθούν ταυτόχρονα περισσότερες της μια εφαρμογές και όλες θα λαμβάνουν ένα αντίγραφο του μηνύματος από το δίκτυο αισθητήρων. Τέλος, στην περίπτωση που έχουμε ανοίξει τον SerialForwarder, το εργαλείο Listen μπορεί επίσης να εκτελεστεί άμεσα χωρίς την ανάγκη να θέσουμε τη μεταβλητή MOTECOM.

5.4.3 Το εργαλείο γραφικής απεικόνισης Oscilloscope

Για την γραφική απεικόνιση των δεδομένων μας χρησιμοποιήσαμε το Java εργαλείο Oscilloscope. Το εργαλείο αυτό δίνει επίσης τη δυνατότητα για αποθήκευση των μετρήσεων, έτσι ώστε να είναι δυνατή στη συνέχεια η περαιτέρω επεξεργασία τους. Αφήνοντας τον SerialForwarder ανοικτό να τρέχει, πληκτρολογούμε την παρακάτω εντολή:

```
java net.TinyOS.oscope.oscilloscope
```

Η εντολή αυτή ανοίγει το παράθυρο γραφικής απεικόνισης του Oscilloscope. Το εργαλείο αυτό συνδέεται με τον serialForwarder μέσω του δικτύου, ανακτά τα πακέτα δεδομένων που του προωθεί, αναλύει τις τιμές των μετρήσεων που στάλθηκαν ασύρματα μέσω του Tmote Sky και τις απεικονίζει γραφικά. Έχοντας συνδεδεμένα όλα τα επιμέρους τμήματα της εφαρμογής μας και εκτελώντας την παραπάνω εντολή βλέπουμε το επόμενο παράθυρο:



Εικόνα 94 Παράθυρο γραφικής απεικόνισης του ληφθέντος ΗΚΓ με το εργαλείο Oscilloscope

Ο οριζόντιος άξονας x του γραφήματος παριστάνει τον αριθμό του μετρητή πακέτων και ο κάθετος άξονας y παριστάνει την τιμή μέτρησης του ADC. Πολλές φορές όταν δεν είναι άμεσα ορατή η κυματομορφή, είναι απαραίτητη η ρύθμιση του zoom με πάτημα των κουμπιών ‘Zoom In’ ή ‘Zoom Out’ για τον οριζόντιο ή τον κατακόρυφο άξονα. Επίσης όταν ο αριθμός των δειγμάτων του μετρητή είναι εκτός του πεδίου απεικόνισης, η ενεργοποίηση της επιλογής ‘Scrolling’ μας οδηγεί απευθείας στις πιο πρόσφατες εικονιζόμενες τιμές. Τέλος, αποεπιλέγοντας το πεδίο ‘Connect Data points’ μπορούμε να δούμε τα αντίστοιχα σημεία (τιμές) που αποτελούν τα δείγματα του ADC που στάλθηκαν ασύρματα και τα οποία σχηματίζουν το καρδιογράφημα.

Τα δεδομένα μπορούν ταυτόχρονα να αποθηκευτούν σε αρχείο.txt πατώντας το πλήκτρο ‘Save Data’ και επιλέγοντας το φάκελο αποθήκευσης.

5.5 Παρατηρήσεις.

Ο σκοπός της διπλωματικής αυτής ήταν η εύρεση ενός αλγορίθμου ανίχνευσης συμπλέγματος QRS που θα μπορούσε να εφαρμοστεί με επιτυχία σε επίπεδο δικτύου ασύρματων αισθητήρων. Θεωρητικά, η τεχνική που επιλέχθηκε στηρίζεται σε απλούς υπολογισμούς και θεωρήθηκε λειτουργική για υλοποίηση σε επίπεδο tmote. Πρακτικά, το γεγονός ότι η πλατφόρμα έχει συγκεκριμένα χαρακτηριστικά δυσχέραινε τη λειτουργικότητα της. Οι λόγοι για τους οποίους συμβαίνει αυτό παρουσιάζονται εκτενώς στη συνέχεια:

Α) Το υπολογιστικό φορτίο της τεχνικής, αν και μικρότερο των άλλων δύο που δεν επελέγησαν, αποδείχθηκε μεγάλο για την περιορισμένη δυνατότητα επεξεργασίας της πλατφόρμας. Στον αλγόριθμο του task της τεχνικής αυτής εμφανίζονται τρία for loops, ανάγκη πρόσβασης σε μεταβλητές που αλλάζουν έξω από αυτό και χρήση πινάκων. Στον απλό αλγόριθμο της τεχνικής αυτής τα στοιχεία αυτά προσδίδουν το μεγαλύτερο προγραμματιστικό φορτίο. Προκειμένου να ξεπεραστεί το πρόβλημα αυτό, δοκιμάστηκαν πιο απλοποιημένες μορφές του αλγορίθμου.

Μια προσπάθεια απλοποίησης του αλγορίθμου περιλάμβανε τη μείωση του μεγέθους του πίνακα array[j] έτσι ώστε να είναι πιο σύντομη η εκτέλεση των for loops. Ακόμη, για να μειωθεί ο αριθμός των for loops, η επιλογή της κορυφής έγινε απλά με εύρεση της μέγιστης τιμής του πίνακα. Ο αλγόριθμος αυτός εκτελέστηκε με επιτυχία για κάποιες τιμές δειγματοληψίας, γύρω στα 100Hz. Το γεγονός, όμως, ότι ο αλγόριθμος δεν ανταποκρίνεται στην τεχνική ανίχνευσης συμπλέγματος QRS καθιστά τα αποτελέσματα μη ικανοποιητικά. Αυτό συμβαίνει για δύο λόγους. Πρώτον, οι τιμές του πίνακα δεν συγκρίνονται με κατώφλι, οπότε δε γνωρίζουμε κατά πόσο ξεπερνούν μία ελάχιστη τιμή. Δεύτερον, το μέγεθος του πίνακα είναι πολύ μικρό, συνεπώς δεν μπορούμε να ξέρουμε αν η μέγιστη τιμή του αντιστοιχεί σε σύμπλεγμα QRS. Ο κώδικας αυτός (ECGSenseaplo) παρατίθεται στο Παράρτημα Β.

Μια άλλη απόπειρα απλούστευσης του κώδικα περιλάμβανε την απλοποίηση των πράξεων υπολογισμού του κατωφλίου. Πιο συγκεκριμένα, το κατώφλι προέκυπτε από τον τύπο:

$$thresh=mean+2*x,$$

όπου mean η μέση τιμή των στοιχείων του πίνακα και x=mean/(αριθμός στοιχείων του πίνακα). Στη συνέχεια, κάθε τιμή του πίνακα συγκρινόταν με το κατώφλι αυτό και, εφόσον το ξεπερνούσε, ανιχνευόταν μία κορυφή. Η απλοποίηση αυτή έχει αρκετές διαφορές από τον αρχικό αλγόριθμο ανίχνευσης και δεν εντοπίζει πραγματικά συμπλέγματα QRS. Παρόλα αυτά, δοκιμάστηκε στην προσπάθειά μας να διαπιστώσουμε σε ποια σημεία ο κώδικας εμφανίζει τη μεγαλύτερη καθυστέρηση κατά την εκτέλεσή του. Ο κώδικας (ECGSense_working) παρατίθεται στο Παράρτημα Β.

Β) Ο καθορισμός της συχνότητας δειγματοληψίας ήταν ένα σημείο που απασχόλησε αρκετά κατά τη διάρκεια υλοποίησης της εφαρμογής. Ο καθορισμός αυτός γίνεται έμμεσα μέσω της χρήσης του Timer. Κάθε φορά που ο Timer σηματοδοτεί το fired() συμβάν τότε καλείται η getData() ώστε να πάρει το δείγμα από τη θύρα που έχουμε ορίσει. Επομένως, ο καθορισμός της τιμής της παραμέτρου x στην εντολή που καλεί τον Timer:

Timer.start(TIMER_REPEAT, x);

(όπου x το χρονικό διάστημα σε ms μεταξύ των διαδοχικών fired συμβάντων), θέτει ουσιαστικά και το ρυθμό δειγματοληψίας του σήματος.

Επειδή γνωρίζουμε ότι το καρδιακό σήμα καλύπτει ένα εύρος συχνοτήτων 0- 100Hz περίπου, μια συχνότητα δειγματοληψίας τουλάχιστον διπλάσια του εύρους αυτού θα κάλυπτε, σύμφωνα με το θεώρημα της δειγματοληψίας, τις ανάγκες μας. Πρακτικά, η τιμή της συχνότητας αυτής καθορίζεται αρκετά μεγαλύτερη της διπλάσιας, ώστε να έχουμε περισσότερα δείγματα ανά περίοδο και να μπορούμε να απεικονίσουμε πιο αξιόπιστα το σήμα μας.

Στην εφαρμογή μας ορίσαμε τη συχνότητα δειγματοληψίας στα 512 Hz, θέτοντας την τιμή της παραμέτρου x στα 2ms. Να σημειώσουμε εδώ ότι η τιμή αυτή της παραμέτρου ‘x’ αναφέρεται σε binary ms για τα οποία ισχύει η σχέση 1 binary ms =1/1024 ms. Για το λόγο αυτό, μια τιμή 2ms για την παράμετρο ‘x’ αντιστοιχεί τελικά σε 512 Hz συχνότητα δειγματοληψίας.

Η μεγαλύτερη απόκλιση στο καρδιογράφημα εντοπίζεται περίπου μέσα στην περιοχή των 20ms και συμβαίνει στο σύμπλεγμα QRS. Είναι σημαντικό να καταγράψουμε το

QRS σύμπλεγμα στο σύνολο του ώστε να μπορούμε να βγάλουμε χρήσιμα συμπεράσματα για την κυματομορφή του ηλεκτροκαρδιογραφήματος. Με μια συχνότητα δειγματοληψίας 512Hz ή περίοδο περίπου 2ms μπορούμε να καταγράφουμε περίπου δέκα (10) δείγματα από το σύμπλεγμα QRS βεβαιώνοντας έτσι ότι το QRS είναι πλήρως ψηφιοποιημένο.

Κατά την εκτέλεση των δοκιμών με αυτή τη τιμή δειγματοληψίας παρατηρήσαμε ότι το κόκκινο led (που αντιστοιχεί στην εκτέλεση του task της τεχνικής) άναψε μόνο μία φορά, πράγμα που σημαίνει ότι το task δεν εκτελούνταν σωστά. Μία εξήγηση για το φαινόμενο αυτό αποτελεί το γεγονός ότι σε τέτοιους ρυθμούς δειγματοληψίας τα δεδομένα φτάνουν πολύ γρήγορα στο tmote, με αποτέλεσμα ο buffer του να μην προλαβαίνει να αδειάσει έγκαιρα ώστε να εκτελεστεί ξανά το task.

Προκειμένου να ξεπεράσουμε το εμπόδιο αυτό, δοκιμάσαμε μικρότερες συχνότητες δειγματοληψίας. Οι τιμές των συχνοτήτων αυτών κινούνταν σε εύρος από 10Hz έως 256Hz. Παρατηρήσαμε ότι μόνο σε μικρές τιμές συχνοτήτων, δηλαδή γύρω στα 10Hz, τα δύο leds αναβόσβηναν με τον ίδιο ρυθμό. Παρόλο που αυτό σήμαινε ότι ο αλγόριθμος δούλευε κανονικά, η συχνότητα αυτή δειγματοληψίας δεν είναι ικανοποιητική για σήματα με μεγάλο φασματικό περιεχόμενο, όπως το ΗΚΓ που μελετάμε. Στις απλοποιημένες μορφές του αλγορίθμου που προαναφέρθηκαν, παρατηρήσαμε ότι σε συχνότητες τάξης 100Hz ο αλγόριθμος, αν και δεν μπορούσε να εντοπίσει τη μορφολογία του σήματος, κατάφερνε να ανιχνεύσει το ρυθμό της κυματομορφής του (envelope).

Πρακτικά, αν είχαμε ακόμα υψηλότερη συχνότητα δειγματοληψίας θα εξασφαλίζαμε περισσότερα σημεία και καλύτερη αναπαράσταση του σήματος. Η αμέσως επόμενη επιλογή που μας δίνει ο Timer είναι 1ms παράμετρο ‘x’ που αντιστοιχεί σε περίπου 1KHz δειγματοληψία. Οι δοκιμές που κάναμε όμως με αυτή την τιμή δεν έφερε σωστά αποτελέσματα.

Γ)Το βασικό χαρακτηριστικό του λειτουργικού συστήματος TinyOS είναι ότι έχει ένα προγραμματιστικό μοντέλο προορισμένο για εφαρμογές «οδηγούμενες από συμβάντα» (event-driven). Αυτό σημαίνει ότι ο κώδικας του προγράμματος δεν εκτελείται σειριακά. Τα tasks είναι λειτουργίες των οποίων η εκτέλεση μετατίθεται χρονικά. Τα events συμβαίνουν ασύγχρονα. Όταν εμφανίζεται ένα event, η εκτέλεση

άλλων λειτουργιών, όπως και των tasks, διακόπτεται. Αυτό ήταν και το κύριο πρόβλημα της περίπτωσης που μελετάμε. Λόγω του ότι το task είναι αργό, όταν εμφανίζονται events για δειγματοληψία, σταματάει η εκτέλεση του. Μία προσπάθεια επίλυσης του προβλήματος αυτού είναι η εφαρμογή της ατμητότητας (atomicity), η οποία εξασφαλίζει την εκτέλεση του task χωρίς διακοπές. Το μειονέκτημα της χρήσης του atomic section είναι ότι, όσο εκτελείται το task, ο ρυθμός με τον οποίο ο buffer γεμίζει με δεδομένα από τη δειγματοληψία υπερβαίνει το ρυθμό με τον οποίο αδειάζει, με αποτέλεσμα κάποια δεδομένα να χάνονται. Αυτό δεν εμποδίζει τη δειγματοληψία και την απεικόνιση του σήματος, αλλά δυσχεραίνει την εφαρμογή της τεχνικής σε όλο τον όγκο των δεδομένων.

Κεφάλαιο 6.

ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΕΚΤΑΣΕΙΣ

6.1 Γενικά συμπεράσματα

Στα πλαίσια αυτής της εργασίας έγινε μια μελέτη στους αλγορίθμους ανίχνευσης συμπλέγματος QRS σε ΗΚΓ και υλοποίηση σε δίκτυο ασύρματων αισθητήρων. Μέσα από αυτή τη μελέτη, χρησιμοποιήθηκε μια εφαρμογή μετάδοσης σήματος προς ένα σταθμό βάσης, με τη χρήση τέτοιων ασύρματων μονάδων δικτύων αισθητήρων. Η εφαρμογή αυτή περιλάμβανε τη δειγματοληψία του σήματος, καθώς επίσης την επεξεργασία του σε επίπεδο δικτύων ασυρμάτων αισθητήρων (on board processing) και τέλος τη μετάδοση των αποτελεσμάτων της επεξεργασίας και απεικόνιση τους 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). Η μελέτη που έγινε δηλαδή αφορούσε αρχικά την επεξεργασία του σήματος μέσω των αλγορίθμων ανίχνευσης συμπλέγματος QRS σε περιβάλλον MATLAB. Το φιλτράρισμα του σήματος, η επιλογή του αριθμού δειγμάτων και ο προσδιορισμός του κατωφλίου με το οποίο συγκρινόταν το κάθε σήμα ήταν μόνο κάποια από τα σημεία που απασχόλησαν στους αλγορίθμους αυτούς. Με αυτόν τον τρόπο προέκυψαν τρεις τεχνικές επεξεργασίας σήματος ΗΚΓ με διαφορετικά χαρακτηριστικά η καθεμιά. Στη συνέχεια, η μελέτη αυτή αφορούσε την παρατήρηση όλων των σταδίων που ακολούθησε το σήμα αυτό μέχρι να απεικονιστεί στον υπολογιστή, ύστερα από την επεξεργασία on board και μετάδοσή του μέσα από το ασύρματο δίκτυο αισθητήρων.

Για το λόγο αυτό, αρχικά επικεντρωθήκαμε στα βασικά χαρακτηριστικά των ασύρματων δικτύων αισθητήρων, όσο και των βασικών μονάδων-κόμβων από τις οποίες αποτελούνται. Τα βασικά πλεονεκτήματα των μονάδων αυτών 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 αποτελούν βασικές παράμετροι στο σχεδιασμό τέτοιων δικτύων είναι ο χρόνος ζωής, δεδομένου ότι πολλά από τα δίκτυα αυτά τροφοδοτούνται από συσσωρευτές (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.

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 χρησιμοποιήθηκε. Η μονάδα tmote υλοποιεί 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 εφαρμογής.

Η λειτουργία του Tmote Sky βασίζεται στο λειτουργικό σύστημα TinyOS, σχεδιασμένο ειδικά για event-driven ενσωματωμένα συστήματα. Όλη η οργανωτική δομή του TinyOS, οι βιβλιοθήκες και οι εφαρμογές είναι γραμμένες, όπως είπαμε, στη γλώσσα προγραμματισμού NesC. Στην εργασία έγινε αναλυτική περιγραφή των σημαντικότερων ιδιοτήτων τόσο του TinyOS όσο και της NesC, καθώς στη συνέχεια χρειάστηκε να προγραμματιστεί η πλατφόρμα Tmote Sky με αυτή τη γλώσσα ώστε να υλοποιηθεί η εφαρμογή.

Η εφαρμογή απαιτούσε την δειγματοληψία του σήματος-ΗΚΓ από μια κατάλληλα ορισμένη θύρα εισόδου του A/D μετατροπέα του Tmote Sky, την επεξεργασία του σε επίπεδο κόμβου ασύρματου δικτύου (on board processing) και την αποστολή των αποτελεσμάτων της επεξεργασίας, που σε αυτή την περίπτωση ήταν ένα άλλο tmote συνδεδεμένο με έναν φορητό υπολογιστή. Σημαντικά στοιχεία κατά την υλοποίηση του κώδικα της εφαρμογής για τον προγραμματισμό της μονάδας αυτής ήταν η επιλογή του κατάλληλου ρυθμού δειγματοληψίας του σήματος από την είσοδο του ADC αλλά και ο τρόπος επεξεργασίας τους. Έτσι πραγματοποιήθηκε μια μελέτη του

αλγορίθμου επεξεργασίας του σήματος σε επίπεδο tmote καθώς και της συμπεριφοράς του με διάφορες τιμές συχνότητας δειγματοληψίας.

Για την εκτέλεση των μετρήσεων, έγιναν πολλές προσπάθειες με αλλαγές στον κώδικα της αποδοτικότερης τεχνικής που επιλέχτηκε με τα κριτήρια που αναφέρθηκαν παραπάνω, καθώς και στις τιμές δειγματοληψίας, για να ξεπεραστούν οι δυσκολίες που προέκυπταν. Η πρώτη δυσκολία, που παρουσιάστηκε, προέκυψε όταν στην τιμή συχνότητας δειγματοληψίας ίση με 512 Hz, κατά την οποία είναι δυνατή η καταγραφή δέκα (10) δειγμάτων από το σύμπλεγμα QRS, παρατηρήθηκε πως το κόκκινο led (που αντιστοιχεί στην εκτέλεση του task της τεχνικής) άναψε μόνο μία φορά. Αυτό ερμηνεύεται ως μη σωστή εκτέλεση του task, πράγμα που οφείλεται στο γεγονός ότι σε τέτοιους ρυθμούς δειγματοληψίας τα δεδομένα φτάνουν πολύ γρήγορα στο tmote, με αποτέλεσμα ο buffer του να μην προλαβαίνει να αδειάσει έγκαιρα ώστε να εκτελεστεί ξανά το task. Προκειμένου να ξεπεράσουμε το εμπόδιο αυτό, δοκιμάσαμε μικρότερες συχνότητες δειγματοληψίας. Οι τιμές των συχνοτήτων αυτών κινούνταν σε εύρος από 10Hz έως 256Hz. Παρατηρήσαμε ότι μόνο σε μικρές τιμές συχνοτήτων, δηλαδή γύρω στα 10Hz, τα δύο leds αναβόσβηναν με τον ίδιο ρυθμό. Παρόλο που αυτό σήμαινε ότι ο αλγόριθμος δούλευε κανονικά, η συχνότητα αυτή δειγματοληψίας δεν είναι ικανοποιητική για σήματα με μεγάλο φασματικό περιεχόμενο, όπως το ΗΚΓ που μελετάμε.

Εκτός από την περιγραφή του κώδικα της εφαρμογής μας για το τμήμα επεξεργασίας και αποστολής του σήματος μας, παρουσιάστηκε και το τμήμα λήψης του. Όπως αναφέρθηκε, το σήμα αποστελλόταν ασύρματα σε ένα σταθμό βάσης, δηλαδή στη δεύτερη μονάδα tmote, συνδεδεμένη με τον υπολογιστή μέσω της θύρας USB. Περιγράψαμε την 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 απαιτούνταν για την απεικόνιση του σήματος (Listen, Oscilloscope, SerialForwarder) και του τρόπου που αυτά χειρίζονται τα λαμβανόμενα πακέτα.

Όσον αφορά το σήμα, έγινε αναλυτική περιγραφή των επιμέρους σταδίων από τα οποία πέρασε, έτσι ώστε από την πρωτογενή του μορφή να φτάσει στην έξοδο του κυκλώματος με τα κατάλληλα χαρακτηριστικά και στη συνέχεια να είναι δυνατό να

εισέλθει στην πλατφόρμα Tmote Sky.Το σήμα που λαμβάναμε τελικά στον υπολογιστή, έφτανε σε μορφή πακέτων.

6.2 Προεκτάσεις

Το πρώτο θέμα με το οποίο ασχοληθήκαμε ήταν η επιλογή των τεχνικών ανίχνευσης συμπλέγματος QRS. Μελετήθηκαν οι αλγόριθμοι πολλών τέτοιων τεχνικών, αλλά βάσει της αποδοτικότητας και της μικρότερης δυνατής πολυπλοκότητας, επελέγησαν κάποιες από αυτές, που εξυπηρετούσαν το σκοπό της εργασίας μας. Μελλοντικά, ωστόσο, προτείνεται η μελέτη και άλλων αλγορίθμων που θα μπορούσαν να αποδειχθούν εξίσου αποδοτικοί στην ανίχνευση συμπλέγματος QRS σε ΗΚΓ. Γνωστός για τη λειτουργικότητά του στον εντοπισμό συμπλέγματος QRS αποτελεί ο αλγόριθμος των Pan-Tompkins. Τα βήματα του αλγορίθμου αυτού είναι, συνοπτικά, τα εξής:

- Φιλτράρισμα του σήματος με ζωνοπερατό φίλτρο(συνδυασμός υψιπερατού και βαθυπερατού φίλτρου) με σκοπό τη μείωση του θορύβου του σήματος

- Διαφόριση (differentiation) του φιλτραρισμένου σήματος, η οποία δίνει πληροφορίες για το slope του συμπλέγματος QRS

- Τετραγωνισμός (squaring), που συντελεί στη μείωση των FP ανιχνεύσεων

- Ολοκλήρωση με χρήση κινούμενου παραθύρου (moving-window integration), που παράγει ένα σήμα το οποίο δίνει πληροφορίες τόσο για το slope όσο και για το πλάτος του συμπλέγματος QRS

- Διπλό προσαρμοστικό κατώφλι (dual threshold) που κάνει την τελική ανίχνευση και μειώνει τις FN ανιχνεύσεις.

Η εφαρμογή θα μπορούσε να επεκταθεί στη μελέτη ΗΚΓ που εμφανίζουν συγκεκριμένη παθολογία. Η μελέτη στην εργασία μας έγινε τόσο για υγιή όσο και για παθολογικά ΗΚΓ, χωρίς όμως να δοθεί εμφαση στα συγκεκριμένα χαρακτηριστικά των παθήσεων αυτών. Μια πρόταση που θα μπορούσε να γίνει είναι η ανάπτυξη αλγορίθμων που θα προγματεύονται ΗΚΓ με μία συγκεκριμένη πάθηση και η ανίχνευση συμπλέγματος QRS θα βασίζεται στα χαρακτηριστικά της. Μια συχνά εμφανιζόμενη πάθηση αποτελεί η κοιλιακή ταχυκαρδία, η οποία χαρακτηρίζεται από

απότομες αλλαγές στο ρυθμό αλλά και στο πλάτος του ΗΚΓ, πράγμα που καθιστά ενδιαφέρουσα τη μελέτη ανίχνευσης συμπλέγματος QRS σε τέτοια ΗΚΓ.

Όσον αφορά την αξιολόγηση των αλγορίθμων ανίχνευσης QRS, στην παρούσα εργασία βασιστήκαμε στους εξής συντελεστές:

- Ευαισθησία (sensitivity): Se= $\frac{TP}{TP+FN}$
- Θετική προβλεπτικότητα (positive predictivity): +P=$\frac{TP}{TP+FP}$

Όπου

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

Μια επέκταση της μελέτης αξιολόγησης των αλγορίθμων θα μπορούσε να πραγματοποιηθεί με εύρεση επιπλέον ανιχνεύσεων. Αυτές θα μπορούσαν να είναι οι εξής:

- **SP** – Shifted positive error: Αφορά την περίπτωση που μία FP ανίχνευση ακολουθείται αμέσως από μία FN ανίχνευση
- **SN** – Shifted negative error: Αφορά την περίπτωση που μία FN ανίχνευση ακολουθείται αμέσως από μία FP ανίχνευση

Με χρήση και των πέντε αυτών συντελεστών θα μπορούσε να υπολογίζονται η ευαισθησία και η θετική προβλεπτικότητα με τους εξής τύπους:

- Ευαισθησία (sensitivity): Se=$\frac{TP}{TP+FN+SN}$
- Θετική προβλεπτικότητα (positive predictivity): +P=$\frac{TP}{TP+FP+SP}$

Με τον τρόπο αυτό η αξιολόγηση των τεχνικών θα μπορούσε ίσως να καταστεί πιο αξιόπιστη.

Η αξιολόγηση των τεχνικών στη συγκεκριμένη εργασία βασίστηκε στη μελέτη σημάτων ΗΚΓ από τη MIT-BIH βάση δεδομένων. Τα σήματα αυτά προτιμήθηκαν, διότι προέρχονται από μια έγκριτη και προτυποποιημένη βάση δεδομένων, δηλαδή τα δεδομένα της χρησιμοποιούνται ως πρότυπα σε πολλές μελέτες. Ωστόσο, επέκταση των δοκιμών θα μπορούσαν να περιλαμβάνουν και πειραματικά σήματα.

Ως προς την υλοποίηση του αλγόριθμου σε γλώσσα NesC, ο βασικός άξονας κατεύθυνσης ήταν η όσο το δυνατόν μικρότερη πολυπλοκότητα του κώδικα, ώστε να ανταποκρίνεται στις απαιτήσεις του λειτουργικού συστήματος TinyOS. Στη συγκεκριμένη εργασία βασιστήκαμε στο χρόνο εκτέλεσης του κώδικα, καθώς και στη μνήμη που αυτός δέσμευε ώστε να βγάλουμε συμπεράσματα για την πολυπλοκότητά του. Οι παράμετροι αυτές θα μπορούσαν μελλοντικά να συντελέσουν στη θέσπιση κριτηρίων για την πολυπλοκότητα των αλγορίθμων και αυτό θα μπορούσε να βοηθήσει στη βελτιστοποίηση τους.

Το βασικότερο, ίσως, αντικείμενο μελλοντικής μελέτης είναι η βελτιστοποίηση του ίδιου του κώδικα που υλοποιεί την τεχνική ανίχνευσης συμπλέγματος QRS σε επίπεδο tmote. Ένα πρώτο βήμα θα μπορούσε να είναι η χρήση του TinyOS-2 στη θέση του TinyOS-1 που επιλέχτηκε στην εργασία αυτή, λόγω του ότι υπήρχαν τμήματα κώδικα δοκιμασμένα από άλλους προγραμματιστές με εγγυημένη λειτουργικότητα. Το πλεονέκτημά του είναι ότι παρέχει τη δυνατότητα αυτοματοποίησης ορισμένων διαδικασιών, γεγονός που το καθιστά πιο αποδοτικό. Μία άλλη ιδέα είναι η αντικατάσταση των tasks από μικρότερα, που θα εκτελούν επιμέρους διεργασίες του αρχικού. Πιο συγκεκριμένα, στην εργασία αυτή, το task της τεχνικής ανίχνευσης QRS εκτελεί πολλές λειτουργίες:

- Εκτέλεση for loop για συλλογή δεδομένων και καταχώρησή τους σε πίνακα
- Προσδιορισμός κατωφλίου μέσω πολύπλοκων πράξεων (που έχουν περιγραφεί αναλυτικά στο κεφάλαιο 5)
- Εκτέλεση for loop για εντοπισμό της μέγιστης τιμής του πίνακα
- Σύγκριση τόσο της τιμής αυτής όσο και των γειτονικών με το κατώφλι

Μία ιδέα βελτιστοποίησης περιλαμβάνει την τμηματοποίηση του task αυτού σε μικρότερα, καθένα από τα οποία θα αναλαμβάνει μία διεργασία. Με τον τρόπο αυτό είναι πιθανό να επιτευχθεί μείωση του απαιτούμενου χρόνου εκτέλεσης και της πολυπλοκότητας. Η κύρια δυσκολία στην προσπάθεια αυτή είναι η υλοποίηση του κώδικα με τρόπο τέτοιο ώστε αυτά τα επιμέρους tasks να καλούν το ένα το άλλο συγχρονισμένα σε ένα ασύγχρονο περιβάλλον, όπως είναι το TinyOS.

Ένα ακόμη βήμα που προτείνεται με βάση τον κώδικα που έχει ήδη υλοποιηθεί σε επίπεδο ασύρματων αισθητήρων είναι η προσθήκη και συλλογή δεδομένων και άλλων αισθητήρων στη πλατφόρμα Tmote Sky, εκτός του ηλεκτροκαρδιογράφου. Αυτοί οι αισθητήρες θα μπορούσαν να είναι είτε επιταχυνσιόμετρα (accelerometers) για την καταγραφή της αναπνοής ενός ατόμου μέσω των κινήσεων του θώρακα ή γενικότερα της κίνησής του μέσα στο χώρο ή ακόμα και αισθητήρες για την καταγραφή του κορεσμού οξυγόνου στο αίμα. Τα βιοσήματα αυτά δεν είναι κύρια σήματα αλλά δίνουν μία πιο ολοκληρωμένη εικόνα των συνθηκών υπό τις οποίες συλλέγεται το βασικό βιοσήμα, που στην περίπτωσή μας είναι το ΗΚΓ. Ο συνδυασμός του βασικού σήματος (ΗΚΓ) και των άλλων σημάτων (context sensing) μας βοηθούν στην εξαγωγή συμπερασμάτων για την κατάσταση της υγείας του ασθενούς.

Μια άλλη επέκταση της επεξεργασίας σε επίπεδο ασύρματων αισθητήρων για τη μελέτη της κατάστασης της υγείας κάθε ασθενούς συνίσταται στη συλλογή μεγάλου αριθμού σημάτων ΗΚΓ του ασθενούς αυτού. Η συλλογή αυτή θα μπορούσε να αποτελέσει μία βάση αναφοράς (pattern) για το συγκεκριμένο ασθενή, με την οποία θα συγκρίνεται κάθε ΗΚΓ που θα λαμβάνεται σε άλλες περιστάσεις. Με τον τρόπο αυτό, θα μπορούσε να γίνει γνωμάτευση ή και πρόβλεψη της κατάστασης του ασθενούς, βασισμένη στα ιδιαίτερα χαρακτηριστικά του βιοσήματός του. Η υλοποίηση, ωστόσο, της επεξεργασίας αυτής σε επίπεδο tmote παρουσιάζει ιδιαίτερες δυσκολίες καθώς έχει μεγάλες απαιτήσεις υπολογιστικής ισχύος.

Μία πολύ ενδιαφέρουσα προσέγγιση αποτελεί η επέκταση της υλοποίησης των αλγορίθμων ανίχνευσης συμπλέγματος QRS από επίπεδο simulation (προγραμματισμός του tmote) σε επίπεδο hardware. Θα μπορούσε να μελετηθεί η κατασκευή ενός ολοκληρωμένου κυκλώματος (chip) που θα εκτελεί μία ή περισσότερες επιμέρους διεργασίες. Το ολοκληρωμένο αυτό κύκλωμα θα πρέπει να έχει συγκεκριμένα χαρακτηριστικά ώστε να μπορεί να συνδέεται στο ειδικό pin

(expansion connector) του tmote. Με τον τρόπο αυτό, επιτυγχάνεται γρηγορότερη επεξεργασία και δεν απαιτείται η πραγματοποίηση πολλών loops στο TinyOS.

Τέλος, μια σημαντική δυνατότητα που παρέχει το Tmote Sky και μπορεί να εξεταστεί, είναι η ενεργοποίηση, με κατάλληλη υλοποίηση κώδικα, του DMA. Ο ελεγκτής DMA, όπως αναφέραμε και στο Κεφάλαιο 2, 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 μονάδας του Μικροελεγκτή (MCU). Με τη χρήση αυτής της δυνατότητας μπορούμε θεωρητικά να βελτιώσουμε τις επιδόσεις και το ρυθμό δειγματοληψίας, αλλά θα πρέπει να εξεταστεί και το θέμα της δυνατότητας ασύρματης αποστολής πακέτων δεδομένων σε μεγάλους ρυθμούς. Μάλιστα αυτό είναι και ένα σημείο που πρέπει να διερευνηθεί περαιτέρω καθώς απασχόλησε αρκετά. Αλλιώς μια αύξηση του ρυθμού δειγματοληψίας παραμένει ουσιαστικά χωρίς αντίκρισμα, στην περίπτωση εκείνη που θέλουμε να έχουμε και μετάδοση δεδομένων με παρόμοιο ρυθμό. 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 απασχολεί αρκετά, αλλά σε περιπτώσεις που είναι η φορητότητα και η μεγάλη διάρκεια αυτονομίας είναι αναγκαίες τότε πρέπει να ληφθεί υπόψη.

6.3 Επίλογος

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

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 αυτοοργάνωσης 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.

ΠΑΡΑΡΤΗΜΑ Α: ΚΩΔΙΚΕΣ ΣΕ ΓΛΩΣΣΑ MATLAB

Α.1 Κώδικας readMIT για τη μελέτη και οπτικοποίηση των σημάτων της βάσης MIT-BIH σε MATLAB

```
% This programm reads ECG data which are saved in format
212.
% (e.g., 100.dat from MIT-BIH-DB, cu01.dat from CU-DB,...)
% The data are displayed in a figure together with the
annotations.
% The annotations are saved in the vector ANNOT, the
corresponding
% times (in seconds) are saved in the vector ATRTIME.
% The annotations are saved as numbers, the meaning of the
numbers can
% be found in the codetable "ecgcodes.h" available at
www.physionet.org.
%
% ANNOT only contains the most important information,
which is displayed
% with the program rdann (available on www.physionet.org)
in the 3rd row.
% The 4th to 6th row are not saved in ANNOT.
%
%
%      created on Feb. 27, 2003 by
%      Robert Tratnig (Vorarlberg University of Applied
Sciences)
%      (email: rtratnig@gmx.at),
%
%      algorithm is based on a program written by
%      Klaus Rheinberger (University of Innsbruck)
```

```
%      (email: klaus.rheinberger@uibk.ac.at)
%
% -------------------------------------------------------
-----------------
clc; clear all;

%------ SPECIFY DATA -----------------------------------
-----------------
PATH= 'C:\Documents and
Settings\alex\Desktop\stella_giota\MIT database'; % path,
where data are saved
HEADERFILE= '107.hea';      % header-file in text format
ATRFILE= '107.atr';         % attributes-file in binary
format
DATAFILE='107.dat';         % data-file
SAMPLES2READ=30000;         % number of samples to be read
                            % in case of more than one
signal:
                            % 2*SAMPLES2READ samples are
read

%------ LOAD HEADER DATA -------------------------------
-----------------
fprintf(1,'\\n$> WORKING ON %s ...\n', HEADERFILE);
signalh= fullfile(PATH, HEADERFILE);
fid1=fopen(signalh,'r');
z= fgetl(fid1);
A= sscanf(z, '%*s %d %d %d',[1,3]);
nosig= A(1);  % number of signals
sfreq=A(2);   % sample rate of data
clear A;
for k=1:nosig
    z= fgetl(fid1);
    A= sscanf(z, '%*s %d %d %d %d %d',[1,5]);
```

```
    dformat(k)= A(1);           % format; here only 212 is
allowed
    gain(k)= A(2);              % number of integers per
mV
    bitres(k)= A(3);            % bitresolution
    zerovalue(k)= A(4);         % integer value of ECG
zero point
    firstvalue(k)= A(5);        % first integer value of
signal (to test for errors)
end;
fclose(fid1);
clear A;

%------ LOAD BINARY DATA ----------------------------------
-----------------
if dformat~= [212,212], error('this script does not apply
binary formats different to 212.'); end;
signald= fullfile(PATH, DATAFILE);            % data in
format 212
fid2=fopen(signald,'r');
A= fread(fid2, [3, SAMPLES2READ], 'uint8')';  % matrix
with 3 rows, each 8 bits long, = 2*12bit
fclose(fid2);
M2H= bitshift(A(:,2), -4);
M1H= bitand(A(:,2), 15);
PRL=bitshift(bitand(A(:,2),8),9);     % sign-bit
PRR=bitshift(bitand(A(:,2),128),5);   % sign-bit
M( : , 1)= bitshift(M1H,8)+ A(:,1)-PRL;
M( : , 2)= bitshift(M2H,8)+ A(:,3)-PRR;
if M(1,:) ~= firstvalue, error('inconsistency in the first
bit values'); end;
switch nosig
case 2
    M( : , 1)= (M( : , 1)- zerovalue(1))/gain(1);
```

```
    M( : , 2)= (M( : , 2)- zerovalue(2))/gain(2);
    TIME=(0:(SAMPLES2READ-1))/sfreq;
case 1
    M( : , 1)= (M( : , 1)- zerovalue(1));
    M( : , 2)= (M( : , 2)- zerovalue(1));
    M=M';
    M(1)=[];
    sM=size(M);
    sM=sM(2)+1;
    M(sM)=0;
    M=M';
    M=M/gain(1);
    TIME=(0:2*(SAMPLES2READ)-1)/sfreq;
otherwise  % this case did not appear up to now!
    % here M has to be sorted!!!
    disp('Sorting algorithm for more than 2 signals not
programmed yet!');
end;
clear A M1H M2H PRR PRL;
fprintf(1,'\\n$> LOADING DATA FINISHED \n');

%------ LOAD ATTRIBUTES DATA ------------------------------
-----------------
atrd= fullfile(PATH, ATRFILE);      % attribute file with
annotation data
fid3=fopen(atrd,'r');
A= fread(fid3, [2, inf], 'uint8')';
fclose(fid3);
ATRTIME=[];
ANNOT=[];
sa=size(A);
saa=sa(1);
i=1;
while i<=saa
```

```
    annoth=bitshift(A(i,2),-2);
    if annoth==59
        ANNOT=[ANNOT;bitshift(A(i+3,2),-2)];
        ATRTIME=[ATRTIME;A(i+2,1)+bitshift(A(i+2,2),8)+...

bitshift(A(i+1,1),16)+bitshift(A(i+1,2),24)];
        i=i+3;
    elseif annoth==60
        % nothing to do!
    elseif annoth==61
        % nothing to do!
    elseif annoth==62
        % nothing to do!
    elseif annoth==63
        hilfe=bitshift(bitand(A(i,2),3),8)+A(i,1);
        hilfe=hilfe+mod(hilfe,2);
        i=i+hilfe/2;
    else

ATRTIME=[ATRTIME;bitshift(bitand(A(i,2),3),8)+A(i,1)];
        ANNOT=[ANNOT;bitshift(A(i,2),-2)];
   end;
   i=i+1;
end;
ANNOT(length(ANNOT))=[];       % last line = EOF (=0)
ATRTIME(length(ATRTIME))=[];   % last line = EOF
clear A;
ATRTIME= (cumsum(ATRTIME))/sfreq;
ind= find(ATRTIME <= TIME(end));
ATRTIMED= ATRTIME(ind);
ANNOT=round(ANNOT);
ANNOTD= ANNOT(ind);
```

```
%------ DISPLAY DATA ------------------------------------
-----------------
figure(1); clf, box on, hold on
plot(TIME, M(:,1),'r');
if nosig==2
    plot(TIME, M(:,2),'b');
end;
for k=1:length(ATRTIMED)
    text(ATRTIMED(k),0,num2str(ANNOTD(k)));
end;
xlim([TIME(1), TIME(end)]);
xlabel('Time / s'); ylabel('Voltage / mV');
string=['ECG signal ',DATAFILE];
title(string);
fprintf(1,'\\n$> DISPLAYING DATA FINISHED \n');

% ------------------------------------------------------
-----------------
fprintf(1,'\\n$> ALL FINISHED \n');
```

A.2 Κώδικες σε MATLAB για την υλοποίηση αλγορίθμων ανίχνευσης συμπλέγματος QRS

A.2.1 Τεχνική 1: Τεχνική με φίλτρο Savitzky-Golay και σταθερό κατώφλι

A.2.1.1 Κώδικας υλοποίησης της τεχνικής (φίλτρο και κατώφλι)

```
function [qrs_count,peak_time_values] =
technique1(ecg1,l,k)
%Initialization values
qrs_count = 0;
index_max = 0;
```

```
peak_values = [];
peak_time_values = [];
period_peak = [];
%filtered_x = moving_average(60,X);
filtered = smooth(ecg1,'sgolay');

%Mean value standard deviation, criterion y_up
mean1 = mean(filtered);
n = length(filtered);
%stdev = sqrt(sum((decvalue_x-mean1).^2/n));
stdev = std(filtered);
y_up = mean1 +l*stdev;
possibleQRS = find(filtered > y_up);
index = 1;

while(index <= length(possibleQRS))


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

    %QRS complex, get the maximum value in the complex.
Inside each RESP
    %complex find the max voltage value at ECG and the
index of this value in order to append them to the
appropriate matrix
    if (index - temp >=k)
        qrs_count = qrs_count + 1;
```

```
        temp_max = -8;
        for j=temp:index-1
            temp1 = filtered(possibleQRS(j));
            if (temp1 >= temp_max)
                %temp_max = temp1;
                index_max = j;
            end
        end

        peak_values = [peak_values
max(filtered(possibleQRS(temp:index-1)))];
        peak_time_values = [peak_time_values
possibleQRS(index_max)];
    end
end
qrs_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 QRS complexes = ');disp(qrs_count);
%period_peak

figure();
plot(filtered,'-*');
hold on;
plot(peak_time_values,peak_values,'r*');
```

```
hold on;
plot(1:length(filtered),y_up);
xlabel('Number of Samples');
ylabel('Amplitude (V)');
title(['Demonstration of QRS detection Technique 1 with
threshold = ',num2str(y_up),' and no of samples =
',int2str(k)]);
```

Α.2.1.2 Κώδικας για την εξαγωγή των στατιστικών και των τελικών ανιχνεύσεων

```
totalqrs_count = 0;
qrs_temp = 0;
table=[];
ecg1=y_3000;

l=[1:0.4:3];
k=[2:4:14];
TP = [];
tp = 0;
FP = [];
fp = 0;
FN = [];
fn = 0;

for x=1:length(l)
    for y=1:length(k)
        [qrs_temp,B] = technique1(ecg1,l(x),k(y));
        totalqrs_count = totalqrs_count + qrs_temp;
        minval=[];
        C=[];
        tp = 0;

        for i=1:length(A1)
```

```
            minval = abs(A1(i)-B(1:length(B)));
            test = find(minval == min(minval));
            if abs(A1(i)-B(test))<=20
                tp = tp + 1;
                C = [C B(test)];
            end
        end

        TP = [TP tp];
        fp = length(B)-length(C);
        FP =[FP fp];
        if length(A1)-length(C)>=0
            fn = length(A1)-length(C);
            FN = [FN fn];
        end

    end
end
```

A.2.2. Τεχνική 2: Τεχνική με προσαρμοστικό φίλτρο και κατώφλι

A.2.2.1 Κώδικας υλοποίησης της τεχνικής (φίλτρο και κατώφλι)

```
function [qrs_count,peak_time_values_t] =
texniki2a(ecg1,k)

peak_values_t = [];
peak_time_values_t = [];
qrs_count = 0;
index_max = 0;
peak_values = [];
peak_time_values = [];
period_peak = [];

%% Differentiator
```

```
y_diff = [];
y_diff_filtered = [];
thresh = [];

for i = 4:length(ecg1)
    y_diff(i-3) = ecg1(i) - ecg1(i-3);
end

%% LPF of y_diff
for i = 5:length(y_diff)
    y_diff_filtered(i-4) = y_diff(i) + 4*y_diff(i-1) +
6*y_diff(i-2) + 4*y_diff(i-3) + y_diff(i-4);
end

%% Absolute value of the filtered
y_diff_filtered_abs = abs(y_diff_filtered);

%% Adaptive Threshold calculation
start_index = 0;
end_index = 0;
test = [];
thresh = [];
thr = 0;

l = fix((length(ecg1)/10));
n = fix(length(y_diff_filtered_abs)/(length(ecg1)/10));

for j = 1:n
    start_index = end_index + 1;
    end_index = j*250;
    test = [test find(y_diff_filtered_abs ==
max(y_diff_filtered_abs(start_index:end_index)))];
    if (j == n)
        start_index = end_index + 1;
```

```
        end_index = length(y_diff_filtered_abs);
        test = [test find(y_diff_filtered_abs ==
max(y_diff_filtered_abs(start_index:end_index)))];
    end
end

f = fix(30000/length(test));
stat_thresh = mean(ecg1) + 1*std(ecg1);
%thr = stat_thresh;
for z = 1:length(test)
    thr = 0.8*stat_thresh + ecg1(test(z))/10;
    thresh = [thresh thr];

    QRS = find(ecg1((z-1)*f+1:z*f) > thresh(z));
    possibleQRS = QRS+((z-1)*f);
    index = 1;

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

    %QRS complex, get the maximum value in the complex.
    %complex find the max voltage value at ECG and the
index of this value in order to append them to the
appropriate matrix
        if (index - temp >=k)
            qrs_count = qrs_count + 1;
        peak_values = [];
```

```
        peak_time_values = [];

        for j=temp:index-1
            temp1 = ecg1(possibleQRS(j));
            if (temp1 >= -8)
                index_max = j;
            end
            peak_values = max(ecg1(possibleQRS(temp:index-
1)));
            peak_time_values = possibleQRS(index_max);
        end

            peak_values_t = [peak_values_t peak_values];
            peak_time_values_t = [peak_time_values_t
peak_time_values];
        end
    end

end

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

% mean_period = mean(period_peak)
 disp('Number of QRS complexes = ');disp(qrs_count);
%period_peak

figure;
plot(ecg1,'-*');
hold on;
```

```
 plot(peak_time_values_t,peak_values_t,'r*');
hold on;
for i = 1:length(test)
    plot((i-1)*f+1:i*f,thresh(i),'r');
end
xlabel('Number of Samples');
ylabel('Amplitude (V)');
title(['Demonstration of QRS detection Technique 2']);
```

A.2.2.2 Κώδικας για την εξαγωγή των στατιστικών και των τελικών ανιχνεύσεων

```
totalqrs_count = 0;
qrs_temp = 0;
table=[];

ecg1=M(:,1);

l=[2 3];
k=[1 2 5 6 10];
TP = [];
tp = 0;
FP = [];
fp = 0;
FN = [];
fn = 0;

for x=1:length(l)
    for y=1:length(k)
        [qrs_temp,B] = texniki2a(ecg1,l(x),k(y));
        totalqrs_count = totalqrs_count + qrs_temp;
        minval=[];
        C=[];
        tp = 0;
```

```
        for i=1:length(A7)
            minval = abs(A7(i)-B(1:length(B)));
            test1 = find(minval == min(minval));
          if isempty(test1)~=1
            if abs(A7(i)-B(test1))<=20
                tp = tp + 1;
                C = [C B(test1)];
            end
          end
        end

        TP = [TP tp];
        fp = length(B)-length(C);
        FP =[FP fp];
        if length(A7)-length(C)>=0
            fn = length(A7)-length(C);
            FN = [FN fn];
        end

    end
```

A.2.3 Τεχνική 3: Τεχνική βασισμένη σε παραγώγους με φίλτρο Savitzky-Golay και σταθερό κατώφλι

A.2.3.1 Κώδικας υλοποίησης της τεχνικής (φίλτρο και κατώφλι)

```
function [peak_count,peaks] = technique4(ecg1,l)
```

```
index_array_x  = [];
filtered_x= smooth(ecg1,'sgolay');
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

diff_fltx = diff(filtered_x);
counter=0;
j=1;
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)];
            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)];
                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
            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
peaks = [];
```

```
peak_count=0;
thresh = mean(filtered_x) + l*std(filtered_x);

for i = 1:length(index_array_x)
    if filtered_x(index_array_x(i))>=thresh
        peaks = [peaks index_array_x(i)];
        peak_count = peak_count+1;
    end
end
disp('Number of QRS complexes =');disp(peak_count);
disp('-----------------------------------------------------
-------------------------------------------------------
');
plot(filtered_x);title('filtered_x');hold
on;plot(peaks,filtered_x(peaks),'r*');hold
on;plot(1:length(filtered_x),thresh);
```

A.2.3.2 Κώδικας για την εξαγωγή των στατιστικών και των τελικών ανιχνεύσεων

```
totalqrs_count = 0;
qrs_temp = 0;
table=[];
ecg1=M(:,1);
 l=[1 2 2.2 3 4 4.5];
TP = [];
tp = 0;
FP = [];
fp = 0;
FN = [];
fn = 0;

 for x=1:length(l)
```

```
        [qrs_temp,B] = technique4(ecg1,l(x));
        totalqrs_count = totalqrs_count + qrs_temp;
        minval=[];
        C=[];
        tp = 0;
        for z=1:length(A0)
            minval = abs(A0(z)-B(1:length(B)));
            test = find(minval == min(minval));
            if isempty(test)~=1
                if abs(A0(z)-B(test))<=5
                    tp = tp + 1;
                    C = [C B(test)];
                end

            end
        end

        TP = [TP tp];
        fp = length(B)-length(C);
        FP =[FP fp];

        if length(A0)-length(C)>=0
            fn = length(A0)-length(C);
            FN = [FN fn];
        end
    end
```

ΠΑΡΑΡΤΗΜΑ Β: ΚΩΔΙΚΕΣ ΣΕ ΓΛΩΣΣΑ NesC

Β.1 Κώδικας της εφαρμογής ECGSense σε γλώσσα nesC

Β.1.1 Αρχείο ECGSense.nc

```
includes adc0;
includes Oscope;

configuration EcgSense
{
   provides {
      interface StdControl;
   }
}

implementation
{
   components Main, EcgSenseM, TimerC, GenericComm, LedsC,
ADCC,OscopeC;

  // Main.StdControl -> EcgSense;
   Main.StdControl -> EcgSenseM.StdControl;
   Main.StdControl -> OscopeC.StdControl;
   Main.StdControl -> GenericComm.Control;
   Main.StdControl -> ADCC;
   Main.StdControl -> TimerC;

   StdControl = EcgSenseM;

   EcgSenseM.OEcgSense -> OscopeC.Oscope[0];
   EcgSenseM.Timer -> TimerC.Timer[unique("Timer")];
   EcgSenseM.SendMsg -> GenericComm.SendMsg[AM_INTMSG];
```

```
   EcgSenseM.Leds -> LedsC;
   EcgSenseM.ADC -> ADCC.ADC[TOS_ADC_ADC0_PORT];
   EcgSenseM.ADCControl -> ADCC;
}
```

B.1.2 Αρχείο ECGSenseM.nc

```
includes IntMsg;
//includes stdio.h;
//#include math.h;


module EcgSenseM
{
      provides {
          interface StdControl;
     }
     uses {
           interface Timer;
           interface SendMsg;
           interface Leds;
           interface ADC;
           interface ADCControl;
           interface Oscope as OEcgSense;
     }
}
implementation
{

  TOS_Msg m_msg;
  struct IntMsg *body;
  uint8_t m_int;
  bool m_sending;
  uint8_t  i;
```

```
  uint8_t array[1024];

/*
float mean;
  float std;
  float thres;
  float max;
*/
  command result_t StdControl.init()    {

      i = 0;
      atomic m_int = 0;
      m_sending = FALSE;
      call Leds.init();
      call Leds.redOn();
      call ADCControl.init();
     call ADCControl.bindPort( TOS_ADC_ADC0_PORT,
TOSH_ACTUAL_ADC_ADC0_PORT );
     return SUCCESS;
  }

  command result_t StdControl.start()
  {
    call Timer.start( TIMER_REPEAT, 2);
    //call ADC.getData();
    return SUCCESS;
  }

  command result_t StdControl.stop()
  {
    return SUCCESS;
  }

  task void texniki1()
```

```
   {
      uint16_t  sum=0;
      uint16_t  tot=0;
      uint16_t  mean = 0;
      float std = 0;
      float thres = 0;
      //uint8_t m_int = 0;
      uint16_t max = 0;
      uint16_t max1 = 0;
      uint16_t j=0;
      uint16_t k=0;

     for (j=1; j<=1024; j++)
        sum = sum + array[j];
     mean = sum/1024;
     for (j=1; j<=1024; j++)
        tot = tot + (array[j]*array[j] - mean);
     std = sqrt(tot/1024);
     thres = mean + 2*std;



     max = array[1];
     for (j=1; j<=512; j++) {
            if (max<array[j]){
                 max = array[j];
                 k=i;
            }
     }

atomic {
if (max>thres && array[k-1]>thres && array[k-2]>thres &&
array[k+1]>thres && array[k+2]>thres) {
        m_int=m_int + 1;
}
```

```
}


  max1= array[513];
     for (j=513; j<=1024; j++){
           if (max1<array[j]){
                 max1= array[j];
                k=i;
            }
     }

atomic {
if (max1>thres && array[k-1]>thres && array[k-2]>thres &&
array[k+1]>thres && array[k+2]>thres) {
    m_int = m_int + 1;
}
}

//return m_int;



/*
#if (max>thres&& array[k]>thres && array[i-2]>thres &&
array[i+1]>thres && array[i+2]>thres)
 #           m_int = m_int + 1
  #  end;
   # if (array[1]>thres && array[2]>thres && array[3]>
thres)
        m_int = m_int +1;
     if (array[1000]>thres && array[999]>thres &&
array[998]>thres)
        m_int = m_int + 1;
```

```
*/


   }



  event result_t Timer.fired()
  {
     uint16_t n=0;
     atomic { i = i + 1; }
     if( m_sending == FALSE )
     {
               body = ( struct IntMsg *)m_msg.data;
               atomic  { n = m_int; }
                    body->val = n;
               body->src = TOS_LOCAL_ADDRESS;
                    if( call SendMsg.send( TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg ) == SUCCESS )
               {
                    m_sending = TRUE;
                    //call SendMsg.send(TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg );
                    call Leds.yellowToggle();

               }
       }


       call ADC.getData();
       call Leds.yellowToggle();
       return SUCCESS;
  }
```

```
  task void putData()
  {
     call OEcgSense.put(m_int);
     //call Leds.yellowOn();
  }

  async event result_t ADC.dataReady( uint16_t data )
  {
    atomic array[i] = data;
    post putData();
    post texniki1();
    return SUCCESS;
  }


  event result_t SendMsg.sendDone( TOS_MsgPtr msg,
result_t success )
  {
      m_sending = FALSE;
      return SUCCESS;
  }



}
```

Β.1.3 Αρχείο header adc0.h

```
enum
{
TOS_ADC_ADC0_PORT = unique("ADCPort"),
```

```
TOSH_ACTUAL_ADC_ADC0_PORT = ASSOCIATE_ADC_CHANNEL(
INPUT_CHANNEL_A0,
REFERENCE_VREFplus_AVss,
REFVOLT_LEVEL_2_5
),
};
```

B.1.4 Αρχείο Oscope.h

```
//$Id: Oscope.h,v 1.1 2004/06/12 15:52:36 cssharp Exp $
/* "Copyright (c) 2000-2003 The Regents of the University
of California.
* All rights reserved.
*
* Permission to use, copy, modify, and distribute this
software and its
* documentation for any purpose, without fee, and without
written agreement
* is hereby granted, provided that the above copyright
notice, the following
* two paragraphs and the author appear in all copies of
this software.
*
* IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE
TO ANY PARTY FOR
* DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES ARISING OUT
* OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN
IF THE UNIVERSITY
* OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

```
*
* THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
* INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY
* AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE
PROVIDED HEREUNDER IS
* ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA
HAS NO OBLIGATION TO
* PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
*/
// @author Nelson Lee
// @author Cory Sharp <cssharp@eecs.berkeley.edu>
#ifndef _H_Oscope_h
#define _H_Oscope_h
#ifndef OSCOPE_MAX_CHANNELS
#define OSCOPE_MAX_CHANNELS 2
#endif
enum
{
OSCOPE_BUFFER_SIZE = 10,
AM_OSCOPEMSG = 10,
AM_OSCOPERESETMSG = 32,
};
typedef struct OscopeMsg
{
uint16_t sourceMoteID;
uint16_t lastSampleNumber;
uint16_t channel;
uint16_t data[OSCOPE_BUFFER_SIZE];
} OscopeMsg_t;
typedef struct OscopeResetMsg
{
```

```
/* Empty payload! */
} OscopeResetMsg_t;
#endif//_H_Oscope_h
```

B.1.5 Αρχείο OscopeC.nc

```
//$Id: OscopeC.nc,v 1.1 2004/06/12 15:52:36 cssharp Exp $

/* "Copyright (c) 2000-2003 The Regents of the University
of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this
software and its
 * documentation for any purpose, without fee, and without
written agreement
 * is hereby granted, provided that the above copyright
notice, the following
 * two paragraphs and the author appear in all copies of
this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE
LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN
IF THE UNIVERSITY
 * OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
 *
```

```
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE
PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA
HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
 */

// @author Cory Sharp <cssharp@eecs.berkeley.edu>

includes Oscope;

configuration OscopeC
{
  provides interface StdControl;
  provides interface Oscope[uint8_t channel];
}
implementation
{
  components OscopeM
         , GenericComm as Comm
         ;

  StdControl = OscopeM;
  Oscope = OscopeM;

  OscopeM.ResetCounterMsg ->
Comm.ReceiveMsg[AM_OSCOPERESETMSG];
  OscopeM.DataMsg -> Comm.SendMsg[AM_OSCOPEMSG];
}
```

B.1.6 Αρχείο OscopeM.nc

```
//$Id: OscopeM.nc,v 1.2 2004/12/02 23:21:07 cssharp Exp $

/* "Copyright (c) 2000-2003 The Regents of the University
of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this
software and its
 * documentation for any purpose, without fee, and without
written agreement
 * is hereby granted, provided that the above copyright
notice, the following
 * two paragraphs and the author appear in all copies of
this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE
LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN
IF THE UNIVERSITY
 * OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE
PROVIDED HEREUNDER IS
```

```
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA
HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
 */

// @author Cory Sharp <cssharp@eecs.berkeley.edu>

includes Oscope;

module OscopeM
{
  provides interface StdControl;
  provides interface Oscope[uint8_t channel];
  uses interface SendMsg as DataMsg;
  uses interface ReceiveMsg as ResetCounterMsg;
}
implementation
{
  enum
  {
    MAX_CHANNELS = OSCOPE_MAX_CHANNELS,
    BUFFER_SIZE = OSCOPE_BUFFER_SIZE,
  };

  typedef struct
  {
    uint8_t index;
    bool send_data_ready;
    uint16_t lastCount;
    uint16_t count;
    uint16_t data[BUFFER_SIZE];
    uint16_t send_data[BUFFER_SIZE];
  } OscopeData_t;
```

```
  TOS_Msg m_msg;
  int m_is_sending;
  int m_send_next;

  OscopeData_t m_data[ MAX_CHANNELS ];

  // reset invoked on any one channel resets all channels.
The oscilloscope
  // visualization is really only useful when all channels
are synchronized.

  command void Oscope.reset[uint8_t channel]()
  {
    OscopeData_t* ch = m_data+0;
    OscopeData_t* chend = m_data+MAX_CHANNELS;

    atomic
    {
      for( ; ch != chend; ch++ )
      {
     ch->index = 0;
     ch->send_data_ready = FALSE;
     ch->count = 0;
      }
    }

    m_send_next = 0;
    m_is_sending = FALSE;
  }

  command result_t StdControl.init()
  {
    call Oscope.reset[MAX_CHANNELS]();
```

```
    return SUCCESS;
  }

  command result_t StdControl.start()
  {
    return SUCCESS;
  }

  command result_t StdControl.stop()
  {
    return SUCCESS;
  }

  // sendHelper and sendHelperTask post tasks to retry
sending a packet until
  // DataMsg.send returns SUCCESS.  It's guarded by
m_is_sending so that a call
  // to Oscope.reset() also flushes any pending
sendHelperTask.

  void sendHelper();
  task void sendHelperTask()
  {
    sendHelper();
  }

  void sendHelper()
  {
    if( m_is_sending == TRUE )
    {
      if( call DataMsg.send( TOS_BCAST_ADDR,
sizeof(OscopeMsg_t), &m_msg ) == FAIL )
      {
     if( post sendHelperTask() == FALSE )
```

```
       m_is_sending = FALSE; //the task didn't post,
nothing to do but drop the msg
      }
    }
  }

  int channel_inc( int ch, int inc )
  {
    ch += inc;
    if( ch >= MAX_CHANNELS )
      ch -= MAX_CHANNELS;
    return ch;
  }

  // send tries to be a little fair by transmitting
pending channels round
  // robin instead of possibly letting one channel starve
another.  Also, as
  // soon as a pending channel is found, its data is
stuffed into the local
  // TOS_Msg.  The channel is then allowed to aggregate
new, additional data,
  // possibly before the message is sent.

  task void send()
  {
    if( m_is_sending == FALSE )
    {
      int i;
      OscopeData_t* chbegin = m_data + 0;
      OscopeData_t* chend = m_data + MAX_CHANNELS;
      OscopeData_t* ch = m_data + m_send_next;

      for( i=0; i<MAX_CHANNELS; i++ )
```

```
      {
     if( ch->send_data_ready == TRUE )
     {
       OscopeMsg_t* body = (OscopeMsg_t*)m_msg.data;
       int n = channel_inc( i, m_send_next );

       body->sourceMoteID = TOS_LOCAL_ADDRESS;
       body->lastSampleNumber = ch->lastCount;
       body->channel = n;
       memcpy( body->data, ch->send_data,
sizeof(uint16_t)*BUFFER_SIZE );

       ch->send_data_ready = FALSE;

       m_is_sending = TRUE;
       m_send_next = channel_inc( n, 1 );

       sendHelper();
       return; // a channel has been queued for send, get
out of here
     }

     if( ++ch == chend )
       ch = chbegin;
      }
    }
  }

  // When a message is done sending, mark the local
TOS_Msg as free to use.
  // Post send again here in case another channel is
pending to send, as well.
```

```
  event result_t DataMsg.sendDone( TOS_MsgPtr msg,
result_t success )
  {
    m_is_sending = FALSE;
    post send();
    return SUCCESS;
  }

  // put increments the reading count regardless if the
value is ultimately
  // dropped or not -- this is appropriate.  Because of
this, we have to save
  // the count as lastCount so that Oscilloscope can
properly visualze in time
  // the aggregated data versus the dropped data.

  task void prepare_send_data()
  {
    OscopeData_t* ch = m_data + 0;
    OscopeData_t* chend = m_data + MAX_CHANNELS;

    for( ; ch != chend; ch++ )
    {
      atomic
      {
     if( ch->index >= BUFFER_SIZE )
     {
       ch->lastCount = ch->count;
       memcpy( ch->send_data, ch->data,
sizeof(uint16_t)*BUFFER_SIZE );
       ch->send_data_ready = TRUE;
       ch->index = 0;
       post send();
     }
```

```
      }
    }
  }

  async command result_t Oscope.put[uint8_t channel](
uint16_t value )
  {
    result_t rv = FAIL;
    atomic
    {
      if( channel < MAX_CHANNELS )
      {
     OscopeData_t* ch = &m_data[channel];
     ch->count++;
     if( ch->index < BUFFER_SIZE )
     {
       ch->data[ ch->index++ ] = value;
       if( ch->index >= BUFFER_SIZE )
         post prepare_send_data();
       rv = SUCCESS;
     }
      }
    }
    return rv;
  }

  event TOS_MsgPtr ResetCounterMsg.receive( TOS_MsgPtr msg
)
  {
    call Oscope.reset[MAX_CHANNELS]();
    return msg;
  }
}
```

B.2 Κώδικας TOSBaseM

```
// $Id: TOSBaseM.nc,v 1.9 2005/07/14 17:48:56 gtolle Exp $

/*                                                  tab:4
 * "Copyright (c) 2000-2003 The Regents of the University
of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this
software and its
 * documentation for any purpose, without fee, and without
written agreement is
 * hereby granted, provided that the above copyright
notice, the following
 * two paragraphs and the author appear in all copies of
this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE
LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN
IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY
WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE
PROVIDED HEREUNDER IS
```

```
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA
HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS."
 *
 * Copyright (c) 2002-2003 Intel Corporation
 * All rights reserved.
 *
 * This file is distributed under the terms in the
attached INTEL-LICENSE
 * file. If you do not find these files, copies can be
found by writing to
 * Intel Research Berkeley, 2150 Shattuck Avenue, Suite
1300, Berkeley, CA,
 * 94704.  Attention:  Intel License Inquiry.
 */

/*
 * @author Phil Buonadonna
 * @author Gilman Tolle
 * Revision:   $Id: TOSBaseM.nc,v 1.9 2005/07/14 17:48:56
gtolle Exp $
 */

/*
 * TOSBaseM bridges packets between a serial channel and
the radio.
 * Messages moving from serial to radio will be tagged
with the group
 * ID compiled into the TOSBase, and messages moving from
radio to
 * serial will be filtered by that same group id.
 */
```

```
#ifndef TOSBASE_BLINK_ON_DROP
#define TOSBASE_BLINK_ON_DROP
#endif

module TOSBaseM {
  provides interface StdControl;
  uses {
    interface StdControl as UARTControl;
    interface BareSendMsg as UARTSend;
    interface ReceiveMsg as UARTReceive;
    interface TokenReceiveMsg as UARTTokenReceive;

    interface StdControl as RadioControl;
    interface BareSendMsg as RadioSend;
    interface ReceiveMsg as RadioReceive;

    interface Leds;
  }
}

implementation
{
  enum {
    UART_QUEUE_LEN = 12,
    RADIO_QUEUE_LEN = 12,
  };

  TOS_Msg    uartQueueBufs[UART_QUEUE_LEN];
  uint8_t    uartIn, uartOut;
  bool       uartBusy, uartCount;

  TOS_Msg    radioQueueBufs[RADIO_QUEUE_LEN];
  uint8_t    radioIn, radioOut;
  bool       radioBusy, radioCount;
```

```
  task void UARTSendTask();
  task void RadioSendTask();

  void failBlink();
  void dropBlink();
  void processUartPacket(TOS_MsgPtr Msg, bool wantsAck,
uint8_t Token);

  command result_t StdControl.init() {
    result_t ok1, ok2, ok3;

    uartIn = uartOut = uartCount = 0;
    uartBusy = FALSE;

    radioIn = radioOut = radioCount = 0;
    radioBusy = FALSE;

    ok1 = call UARTControl.init();
    ok2 = call RadioControl.init();
    ok3 = call Leds.init();

    dbg(DBG_BOOT, "TOSBase initialized\n");

    return rcombine3(ok1, ok2, ok3);
  }

  command result_t StdControl.start() {
    result_t ok1, ok2;

    ok1 = call UARTControl.start();
    ok2 = call RadioControl.start();

    return rcombine(ok1, ok2);
```

```
  }

  command result_t StdControl.stop() {
    result_t ok1, ok2;

    ok1 = call UARTControl.stop();
    ok2 = call RadioControl.stop();

    return rcombine(ok1, ok2);
  }

  event TOS_MsgPtr RadioReceive.receive(TOS_MsgPtr Msg) {

    dbg(DBG_USR1, "TOSBase received radio packet.\n");

    if ((!Msg->crc) || (Msg->group != TOS_AM_GROUP))
      return Msg;

    if (uartCount < UART_QUEUE_LEN) {

      memcpy(&uartQueueBufs[uartIn], Msg,
sizeof(TOS_Msg));
      uartCount++;

      if( ++uartIn >= UART_QUEUE_LEN ) uartIn = 0;

      if (!uartBusy) {
     if (post UARTSendTask()) {
       uartBusy = TRUE;
     }
      }
    } else {
      dropBlink();
    }
```

```
    return Msg;
  }

  task void UARTSendTask() {
    dbg (DBG_USR1, "TOSBase forwarding Radio packet to
UART\n");

    if (uartCount == 0) {

      uartBusy = FALSE;

    } else {

      if (call UARTSend.send(&uartQueueBufs[uartOut]) ==
SUCCESS) {
     call Leds.greenToggle();
      } else {
     failBlink();
     post UARTSendTask();
      }
    }
  }

  event result_t UARTSend.sendDone(TOS_MsgPtr msg,
result_t success) {

    if (!success) {
      failBlink();
    } else {
      uartCount--;
      if( ++uartOut >= UART_QUEUE_LEN ) uartOut = 0;
    }
```

```
    post UARTSendTask();

    return SUCCESS;
  }

  event TOS_MsgPtr UARTReceive.receive(TOS_MsgPtr Msg) {
    processUartPacket(Msg, FALSE, 0);
    return Msg;
  }

  event TOS_MsgPtr UARTTokenReceive.receive(TOS_MsgPtr
Msg, uint8_t Token) {
    processUartPacket(Msg, TRUE, Token);
    return Msg;
  }

  void processUartPacket(TOS_MsgPtr Msg, bool wantsAck,
uint8_t Token) {
    bool reflectToken = FALSE;

    dbg(DBG_USR1, "TOSBase received UART token
packet.\n");

    if (radioCount < RADIO_QUEUE_LEN) {
      reflectToken = TRUE;

      memcpy(&radioQueueBufs[radioIn], Msg,
sizeof(TOS_Msg));

      radioCount++;

      if( ++radioIn >= RADIO_QUEUE_LEN ) radioIn = 0;

      if (!radioBusy) {
```

```
     if (post RadioSendTask()) {
       radioBusy = TRUE;
     }
      }
    } else {
      dropBlink();
    }

    if (wantsAck && reflectToken) {
      call UARTTokenReceive.ReflectToken(Token);
    }
  }

  task void RadioSendTask() {

    dbg(DBG_USR1, "TOSBase forwarding UART packet to
Radio\n");

    if (radioCount == 0) {

      radioBusy = FALSE;

    } else {

      radioQueueBufs[radioOut].group = TOS_AM_GROUP;

      if (call RadioSend.send(&radioQueueBufs[radioOut])
== SUCCESS) {
     call Leds.redToggle();
      } else {
     failBlink();
     post RadioSendTask();
      }
    }
```

```
  }

  event result_t RadioSend.sendDone(TOS_MsgPtr msg,
result_t success) {

    if (!success) {
      failBlink();
    } else {
      radioCount--;
      if( ++radioOut >= RADIO_QUEUE_LEN ) radioOut = 0;
    }

    post RadioSendTask();
    return SUCCESS;
  }

  void dropBlink() {
#ifdef TOSBASE_BLINK_ON_DROP
    call Leds.yellowToggle();
#endif
  }

  void failBlink() {
#ifdef TOSBASE_BLINK_ON_FAIL
    call Leds.yellowToggle();
#endif
  }
}
```

Β.3 Κώδικας ECGSenseaplo

```
includes IntMsg;

//includes stdio.h;
#include <math.h>;


module EcgSenseM
{
      provides {
          interface StdControl;
     }
     uses {
           interface Timer;
           interface SendMsg;
           interface Leds;
           interface ADC;
           interface ADCControl;
           interface Oscope as OEcgSense;
     }
}
implementation
{

  TOS_Msg m_msg;
  struct IntMsg *body;
  uint8_t m_int;
  bool m_sending;
  uint8_t  i;
  uint8_t array[1024];

/*
float mean;
```

```
  float std;
  float thres;
  float max;
*/
  command result_t StdControl.init()    {

      i = 0;
      atomic m_int = 0;
      m_sending = FALSE;
      call Leds.init();
     // call Leds.redOn();
      call ADCControl.init();
     call ADCControl.bindPort( TOS_ADC_ADC0_PORT,
TOSH_ACTUAL_ADC_ADC0_PORT );
     return SUCCESS;
  }

  command result_t StdControl.start()
  {
    call Timer.start( TIMER_REPEAT, 100);
    //call ADC.getData();
    return SUCCESS;
  }

  command result_t StdControl.stop()
  {
    return SUCCESS;
  }



  task void texniki1()
   {
```

```
call Leds.redOn();
atomic {

uint16_t  sum=0;
uint16_t  tot=0;
uint16_t  mean = 0;
float std = 0;
float thres = 0;
//uint8_t m_int = 0;
uint16_t max = 0;
uint16_t max1 = 0;
uint16_t j=0;
uint16_t k=0;

for (j=1; j<=10; j++) {
   sum = sum + array[j];
}

mean = sum/10;

for (j=1; j<=10; j++) {
  tot = tot + (array[j]*array[j] - mean);
}

std = (tot/10);
thres = mean + 2*std;
max = array[1];

for (j=1; j<=5; j++)  {
    if (max<array[j]) {
        max = array[j];
        k=i;
    }
}
```

```
      max1= array[5];

      for (j=5; j<=10; j++) {
          if (max1<array[j]) {
             max1= array[j];
             k=i;
          }
      }

    }
     call Leds.redToggle();
  }

  event result_t Timer.fired()
  {
     uint16_t n=0;
     atomic {
           i = i + 1;
        if( m_sending == FALSE )
        {
          post texniki1();
          //call Leds.yellowOn();
             body = ( struct IntMsg *)m_msg.data;
               n = m_int;
               body->val = n;
               body->src = TOS_LOCAL_ADDRESS;
               if( call SendMsg.send( TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg ) == SUCCESS )
               {
             m_sending = TRUE;
                //call SendMsg.send(TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg );
                  call Leds.greenOn();
```

```
          call Leds.redToggle();
               }
           }
        }
        call ADC.getData();
        //call Leds.yellowOn();
        return SUCCESS;
  }

  task void putData()
  {
     call OEcgSense.put(m_int);
     //call Leds.yellowOn();
  }
  async event result_t ADC.dataReady( uint16_t data )
  {
    //atomic array[i] = data;
    post putData();
    //post texniki1();
    //call Leds.greenOn();
    //post texniki1();
    //call Leds.yellowToggle();
    return SUCCESS;
  }

  event result_t SendMsg.sendDone( TOS_MsgPtr msg,
result_t success )
  {
      m_sending = FALSE;
      call Leds.greenToggle();
      return SUCCESS;
  }
}
```

B.4 Κώδικας ECGSense_working

```
includes IntMsg;
//includes stdio.h;
#include <math.h>;


module EcgSenseM
{
      provides {
          interface StdControl;
     }
     uses {
           interface Timer;
           interface SendMsg;
           interface Leds;
           interface ADC;
           interface ADCControl;
           interface Oscope as OEcgSense;
     }
}
implementation
{

  TOS_Msg m_msg;
  struct IntMsg *body;
  uint8_t m_int;
  bool m_sending;
  uint8_t  i;
  uint8_t array[10];

  command result_t StdControl.init()    {

       i = 0;
```

```
      atomic m_int = 0;
      m_sending = FALSE;
      call Leds.init();
     // call Leds.redOn();
      call ADCControl.init();
     call ADCControl.bindPort( TOS_ADC_ADC0_PORT,
TOSH_ACTUAL_ADC_ADC0_PORT );
     dbg(DBG_BOOT,"Initialized\n");
     return SUCCESS;
  }

  command result_t StdControl.start()
  {
    call Timer.start( TIMER_REPEAT, 100);
    //call ADC.getData();
    return SUCCESS;
  }

  command result_t StdControl.stop()
  {
    return SUCCESS;
  }



  task void texniki1()
   {

      call Leds.redOn();
      atomic {
      uint16_t  sum=0;
      //uint16_t  tot=0;
      uint16_t  mean = 0;
      float std = 0;
```

```
    float thres = 0;
    uint8_t m_int = 0;
    //uint16_t max = 0;
    //uint16_t max1 = 0;
    uint16_t j=0;
    //uint16_t k=0;


    //m_int = 7;

    for (j=1; j<=10; j++) {
       sum = sum + array[j];
    }

    mean = sum/10;
    std = (mean/10);
    thres = mean + 2*std;

    for (j=1; j<=10; j++) {
     if (array[j]> thres) {
        m_int = m_int + 1;
     }
    }
  }
    call Leds.redToggle();
}

event result_t Timer.fired()
{
      uint16_t n=0;
         call ADC.getData();
         i = i + 1;
      if(( m_sending == FALSE )&(i%10 == 0))
      {
```

```
          atomic {
               post texniki1();
               //call Leds.yellowOn();
               body = ( struct IntMsg *)m_msg.data;
                    n = m_int;
                    body->val = n;
                    body->src = TOS_LOCAL_ADDRESS;
          }
                    if( call SendMsg.send( TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg ) == SUCCESS )
                    {
                m_sending = TRUE;
                   //call SendMsg.send(TOS_BCAST_ADDR,
sizeof(IntMsg), &m_msg );
                     call Leds.greenOn();
             call Leds.redToggle();
                  }
           }
        //call ADC.getData();
        //call Leds.yellowOn();
        return SUCCESS;
  }

  task void putData()
  {
     call OEcgSense.put(m_int);
     //call Leds.yellowOn();
  }

  async event result_t ADC.dataReady( uint16_t data )
  {
    //atomic array[i] = data;
    post putData();
    //post texniki1();
```

```
    //call Leds.greenOn();
    //post texniki1();
    //call Leds.yellowToggle();
    return SUCCESS;
  }


  event result_t SendMsg.sendDone( TOS_MsgPtr msg,
result_t success )
  {
      m_sending = FALSE;
      call Leds.greenToggle();
      return SUCCESS;
  }
}
```

Βιβλιογραφία - Αναφορές

[1] *'Sensor Networks for Medical Care'*, Victor Shnayder, Borrong Chen, Konrad Lorincz, Thaddeus R. F. Fulford-Jones and Matt Welsh. Technical Report TR- 08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.

[2] *'Managing Care through the Air'*, Philip E. Ross. IEEE Spectrum, December 2004

[3] *'Survey on Sensor Networks',* Praveen Rentala, Ravi Musunuri, Shashidlar Gandham, Udit Saxena. Survey Paper submitted as per the requirements of Mobile Computing (CS 6392) Course, University of Texas at Dallas

[4] *'A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation',* Emil Jovanov, Aleksandar Milenkovic, Chris Otto and Piet C de Groen. Journal of NeuroEngineering and Rehabilitation 2005, 2:6

[5] *'Wireless Sensor Networks for Habitat Monitoring',* Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson. WSNA'02, September 28, 2002, Atlanta, Georgia, USA

[6] CELNET, Final Report, 16 March 2005, Areej Al Al-Khalidi, Qaies Alkal Alkaldi di, Khaled Almazrooei, Jeremy Carrier, Sean Kinney, Sean Veit

[7] *'Compatibility of IEEE802.15.4 (ZigBee) with IEEE802.11 (WLAN), Bluetooth and Microwave Ovens in 2,4 GHz ISM-Band'* . Test Report- Revised Version V0.3: 12th September 2004. Prof. Dr.-Ing. Axel Sikora, Steinbeis Transfer Centre for Embedded Design and Networking, University of Cooperative Education Loerrach, Germany

[8] *'Heart Rate and EKG Monitor using the MSP430FG439'*, Texas Instruments, SLAA280 October 2005

[9] *'The Platforms enabling Wireless Sensor Networks',* Jason Hill, Mike Horton, Ralph Kling and Lakshman Krishnamurthy. COMMUNICATIONS OF THE ACM, June 2004/Vol.47, No.6

[10] *'The NesC Language: A Holistic Approach to Networked Embedded Systems',* David Gay, Phil Levis, Robert von Behren, Matt Welsh, Eric Brewer and David

Culler. Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and implementation (2003)

[11] *'Smoke Detector Wireless Sensor Network',* Report by Team WI-FIVE-O

[12] *'Wireless Sensor Networks: a survey',* I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. Computer Networks 38 (2002) 393-422

[13] *'Sensor Networks for Emergency Response: Challenges and Opportunities',* K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, M. Welsh. PERVASIVE Computing, October December 2004, p.16-23

[14] '*Deciphering TinyOS Serial Packets'*, Jeff Thorn, Octave Tech Brief #5-01

[15] *'Telos: Enabling Ultra-Low Power Wireless Research'*, Joseph Polastre, Robert Szewczyk and David Culler, In Proceedings of IPSN/SPOTS, Los Angeles, CA, April 25-27, 2005

[16] *'A Portable, Low-Power, Wireless Two-Lead EKG System'*, Thaddeus R. F. Fulfold-Jones, Gu-Yeon Wei, Matt Welsh. Proceedings of the 26th Annual International Conference of the IEEE EMBS San Francisco, CA, USA, September 1-5, 2004

[17] *'The Emergence of Networking Abstractions and Techniques in TinyOS',* Philip Levis, Sam Madden, David Gay, Joseph Polastre, Robert Szewczyk, Alec Woo, Eric Brewer and David Culler.

[18] *'Wireless Sensor Network Designs'*, A. Hac. 2003 John Willey & Sons, Ltd

[19] *'Designing a ZigBee-ready IEEE 802.15.4- comliant radio transceiver'*, Khanh Tuan Le, www.rfdesign.com, November 2004

[20] *'Meet the ZigBee Standard Article'*, http://www.sensorsmag.com

[21] '*Wireless Sensor Network Programming'*, http://www.andrew.cmu.edu/user/get/wsn/index.html

[22] *'A Closer Look at the Advanced CODAS Moving Average Algorithm'*, DATAQ INSTRUMENTS

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

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

[**25]** *'Application of Adaptive and Non Adaptive Filters in ECG Signal Processing'*, Kamran Jamshaid, Omar Akram, Farooq Sabir, Dr. Syed Ismail Shah,Dr. Jamil Ahmed

[26] '*Ventricular beat detection in single channel electrocardiograms*', Ivan A Dotsinsky and Todor V Stoyanov, BioMedical Engineering OnLine 2004, 3:3

[27] 'The *Principles of Software QRS Detection*', Bert-Uwe Köhler, Carsten Hennig, Reinhold Orglmeister, IEEE ENGINEERING IN MEDICINE AND BIOLOGY, January/February 2002

[28] *'Εισαγωγή στη Βιοϊατρική Τεχνολογία και Ανάλυση Ιατρικών Σημάτων',* Δ. Κουτσούρης, Σ. Παυλόπουλος, Α. Πρέντζα, Εκδ. Τζιόλα, Θεσσαλονίκη 2003

[29] '*Μετρήσεις και έλεγχοι στη Βιοϊατρική Τεχνολογία',* Δ. Κουτσούρης, Δ. Γιόβα, ΕΜΠ, Αθήνα 2005

[30] Διπλωματική εργασία, *'Υλοποίηση στη γλώσσα περιγραφής υλικού Verilog των δύο πρώτων φίλτρων του αλγορίθμου των J. Pan & W.J. Tompkins για την ανίχνευση του συμπλέγματος QRS του ΗΚΓ',* Ευσταθία Δ. Κορκού, ΕΜΠ, Αθήνα, Μάρτιος 2005

[31] Διπλωματική εργασία, *'Μετάδοση και επεξεργασία ζωτικών σημείων μέσω ασύρματων δικτύων αισθητήρων',* Ιωάννης Γ. Νέλλας, Σταύρος Β. Πινάτσης, ΕΜΠ, Αθήνα, Μάρτιος 2006

[32] Διπλωματική εργασία, *'Ασύρματα δίκτυα ασθητήρων για τη μη επεμβατική παρακολούθηση βιοσήματος',* Λοίζος Λοίζου, ΕΜΠ, Αθήνα, Μάιος 2008

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

[34] Moteiv, www.moteiv.com

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

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

[37] www.zigbee.com

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

[39] http://www.tinyos.net/tinyos-1.x/doc/tutorial

[40] http://www.physionet.org

[41] '*A real-time QRS detection algorithm'*, *IEEE Trans. Biomed. Eng.*, J. Pan and W. J. Tompkins, BME-32 (3):230-236, 1985

[42] *'Instruments and Methods Statistical techniques to select detection thresholds for peak signals in ice-core data',* 2005 L. Karlöf, T.A. Oigard, F. Godtliebsen, M. Kaczmarska, H. Fischer

[43] *'Real time electrocardiogram QRS detection using combined adaptive threshold'*, Ivaylo I Christov

[44] *'Αλγόριθμοι και Πολυπλοκότητα',* Στάθης Ζάχος, ΕΜΠ, Αθήνα 2005