ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Implementing an intelligent wireless sensor network for industrial preventive monitoring with Prismasense

Συγγραφέας Ανδρέας Παπαγεωργίου

Επιβλέπων	Συνεπιβλέπων
Καθ. Παύλος Σπυράκης	Δρ. Ιωάννης Χατζηγιαννάκης

ΠΑΤΡΑ, 2010

Abstract

The scope of this thesis was the implementation of an intelligent wireless sensor network for industrial preventive monitoring, based on PrismaSense Platform. PrismaSense is an existing application development platform. Our goal is to monitor wear in industrial machinery using a wireless sensor network.

Sensors can measure many things such as temperature, humidity, acceleration, acoustic emission etc. Our main focus was to use the onboard accelerometers on PrismaSense's End-nodes (Quax MS) hoping that when a machine is wearing out, it should produce intense vibrations.

A crude approach is to have every sensor sample every millisecond the acceleration and send it to the main server who is responsible for gathering all the information and deciding whether the machine is reaching the end of its lifetime.

What we tried to achieve is to give our sensors some Artificial Intelligence by letting them decide whether the machine is going to break down. To do that we need to have smart sensors who are able to make calculations on the data they produce. In fact that's what PrismaSense Platform is all about.

Each sensor is equipped with an MSP430, an ultra-low power microprocessor by Texas Instruments. On top they have an xbee RF module so that they can wirelessly send an receive data.

What we tried to do was two applications to manipulate the input data. First we implemented the Root Mean Squared statistical measure and then we moved our analysis to the frequency domain implementing the radix-2 decimation in time Fast Fourier Transform. Sensors are not sending raw data anymore. In the first case they wait for a given time period and then send the RMS value of a given number of acceleration samples and in the second case they send the transformed frequency values. Making our system cleverer the FFT is

further analyzed so that sensors should send an alarm, only when the systems produces values to exceed a threshold.

Enjoy Reading.

Ευχαριστίες

Πριν προχωρήσω θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες σε όλους αυτούς τους ανθρώπους που συνέβαλλαν στο να φέρω εις πέρας την παρούσα Διπλωματική Εργασία.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Επιβλέποντα καθηγητή αυτής της εργασίας κ. Παύλο Σπυράκη αλλά και τον συνεπιβλέποντα Δρ. Ιωάννη Χατζηγιαννάκη για την πολύτιμη βοήθειά του και τη διαρκή υποστήριξή του σε όλες τις φάσεις της παρούσας εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω την Πρίσμα Ηλεκτρονική ΑΒΕΕ για την γενναιόδωρη παροχή του απαραίτητου υλικού αλλά και λογισμικού για να μπορέσουμε να εργαστούμε πάνω σε αυτόν τον τομέα. Συγκεκριμένα θα ήθελα να ευχαριστήσω τον κύριο Σουκουλιά Πέτρο ο οποίος είχε την αρχική ιδέα για τη διπλωματική και τον κύριο Σεραφείμ Κάτσικα διευθυντή του τμήματος έρευνας και ανάπτυξης της Πρίσμα για τον πολύτιμο χρόνο του που μας αφιέρωσε βοηθώντας μας να καταλάβουμε τον τρόπο λειτουργίας της πλατφόρμας PrismaSense. Ακόμα θέλω να ευχαριστήσω τον κύριο Ζήση Καμαργιάννη για την βοήθειά του σε τεχνικά θέματα αλλά και τον Δημήτρη?? για τις βοήθεια που μας έδωσε στο θεωρητικό επίπεδο του Toolwear monitoring όπως επίσης και όλα τα υπόλοιπα παιδιά στο τμήμα έρευνας και ανάπτυξης της Πρίσμα που με ανέχτηκαν στα πόδια τους και με παρότρυναν συνεχώς.

Τέλος θέλω να ευχαριστήσω και το συμφοιτητή μου Γιώργο Γόντικα με τον οποίο η όλη ιδέα, ανάλυση και υλοποίηση της εφαρμογής αποτελεί προϊόν γόνιμης συνεργασίας.

Contents

This page intentionally left blank

Κεφάλαιο 1 – Εισαγωγή

Κίνητρο και σημασία του θέματος

Τα τελευταία χρόνια η πρόοδος που σημειώθηκε στην κατασκευή απλών μικρών κυκλωμάτων χαμηλής κατανάλωσης με χαμηλό κόστος οδήγησε σε ένα νέο τεχνολογικό επίτευγμα, τα ασύρματα δίκτυα αισθητήρων (WSNs). Τα δίκτυα αυτά συνδυάζουν ασύρματη επικοινωνία και δυνατότητες υπολογισμών με αισθητήρες φυσικών φαινομένων που μπορούν πολύ εύκολα να ενσωματωθούν στο φυσικό περιβάλλον. Το μέγεθος ενός αισθητήρα είναι μερικά κυβικά χιλιοστά και η επιθυμητή τιμή πολύ χαμηλή (10-100 ευρώ) συμπεριλαμβανομένου του μικροελεγκτή, του module επικοινωνίας, της τροφοδοσίας και του ίδιου του αισθητήρα. Όλες αυτές οι συσκευές βρίσκονται τοποθετημένες στην ίδια συσκευή και αποτελούν έναν κόμβο (node).

Τα ασύρματα δίκτυα αισθητήρων, τα οποία μπορούν να θεωρηθούν μία ειδική μορφή ad hoc δικτύων με μειωμένη έως καθόλου κινητικότητα, αναμένεται να βρουν προσοδοφόρο έδαφος ανάπτυξης αφού παρέχουν ένα αξιόπιστο περιβάλλον μέτρησης και ανάλυσης περιβαλλοντικών μεγεθών. Τα δίκτυα αυτά είναι “data centric” (λειτουργούν με επίκεντρο τα δεδομένα), δηλαδή σε αντίθεση με τα παραδοσιακά ad hoc δίκτυα όπου δεδομένα λαμβάνονται από συγκεκριμένο κόμβο, τα δεδομένα λαμβάνονται σύμφωνα με κάποιες ιδιότητες, για παράδειγμα «σε ποια περιοχή η θερμοκρασία είναι μεγαλύτερη από 30^{o}». Επομένως, απαιτείται μεγάλος αριθμός αισθητήρων για να μπορέσουμε να έχουμε αξιόπιστες τιμές συγκεκριμένων ιδιοτήτων σε μία περιοχή.

Ένας τυπικός αισθητήρας αποτελείται από έναν ασύρματο πομποδέκτη για να μπορεί να λαμβάνει και να στέλνει δεδομένα, έναν ενσωματωμένο επεξεργαστή, μία μικρή μονάδα μνήμης και τον αισθητήρα μέτρησης του φυσικού μεγέθους που επιθυμούμε. Όλα τα παραπάνω τροφοδοτούνται από μία ενσωματωμένη μπαταρία.

Μια περιοχή στην οποία αυτά τα δίκτυα αισθητήρων βρίσκουν μεγάλη εφαρμογή είναι η παρακολούθηση της φθοράς μηχανών. Ειδικά στον τομέα της βιομηχανικής παραγωγής όπου πολλές μηχανές και εξαρτήματα συνδυάζονται στην παραγωγή με την έξοδο ενός συστήματος να είναι είσοδος για ένα άλλο, είναι πολύ σημαντικό να παρακολουθείται κάθε στάδιο της παραγωγής έτσι ώστε αφενός μεν να έχουμε μια αυτοματοποιημένη και το δυνατόν ταχύτερη και αποδοτικότερη παραγωγή, αφετέρου δε το τελικό προϊόν να πληροί τις προδιαγραφές ποιότητας που έχουν οριστεί στον αρχικό σχεδιασμό του.

Πιο συγκεκριμένα η παρακολούθηση της φθοράς και των βλαβών των μηχανών είναι ένα ζήτημα που έχει προκαλέσει το ενδιαφέρον αρκετών ερευνητών τα τελευταία χρόνια. Οι λόγοι για τους οποίους η παρακολούθηση της κατάστασης αυτών των εξαρτημάτων θεωρείται τόσο σημαντική είναι :

- Η αυτοματοποιημένη, χωρίς τον ανθρώπινο παράγοντα, παραγωγή είναι εφικτή μόνο αν υπάρχει μια μέθοδος για την παρακολούθηση της φθοράς και διάγνωσης των βλαβών των διαφόρων εξαρτημάτων που απαρτίζουν την εκάστοτε μηχανή.
- Η φθορά των εργαλείων αυτών επηρεάζει σημαντικά την ποιότητα του εκάστοτε τελικού προϊόντος που θα κατασκευαστεί, άρα πρέπει να εξασφαλιστεί ότι λειτουργούν εντός αποδεκτών πλαισίων φθοράς.
- Η διάρκεια ζωής ενός εξαρτήματος και κατ' επέκταση η πλήρης αξιοποίησή του, συνδέεται άμεσα με την παρακολούθηση της φθοράς του και αυτό γιατί παρά τον προβλεπόμενο χρόνο ζωής που δίνει ο κατασκευαστής, ο πραγματικός χρόνος ζωής του διαφέρει. Για παράδειγμα ένα ρουλεμάν μπορεί να χαλάσει πολύ πριν την ώρα του(αν βγει ελαττωματικό) ή να αντέξει πολύ παραπάνω από τα όρια που δίνει ο κατασκευαστής του.
- Μέχρι σήμερα οι αλλαγές των εξαρτημάτων αυτών ήταν βασισμένες σε συμβατικές εκτιμήσεις των ορίων λειτουργίας τους, αγνοώντας έτσι ξαφνικές αστοχίες υλικού και βλάβες ενώ παράλληλα οδηγούσαν σε περιττά υψηλό αριθμό αλλαγών των

εξαρτημάτων γιατί δεν αξιοποιούνταν στο έπακρο τα υπάρχοντα. Αποτέλεσμα αυτού είναι το χάσιμο πολύτιμου χρόνου παραγωγής και επαγωγικά αρκετών χρημάτων.

Συμπερασματικά η ανάγκη για μεθόδους παρακολούθησης της φθοράς των μηχανημάτων κρίνεται επιτακτική για μια αυτοματοποιημένη, αποτελεσματική και οικονομικότερη παραγωγή. Είναι γνωστό άλλωστε ότι τα οικονομικά μεγέθη στις μοντέρνες βιομηχανίες είναι πολύ μεγάλα εξ' αιτίας του υψηλού κόστους των επενδύσεων που γίνονται για τον εξοπλισμό των εργοστασίων και άρα είναι προς το ίδιο τους το συμφέρον η βέλτιστη αξιοποίηση του εξοπλισμού αυτού αλλά και η ταχύτερη και αποδοτικότερη παραγωγή, με την αρωγή αυτοματοποιημένων συστημάτων παρακολούθησης της ορθής λειτουργίας της.

Στόχοι της διπλωματικής

Για τη διπλωματική αυτή, είχαμε την τύχη, εγώ και ο συνάδελφος μου Γιώργος Γόντικας να συνεργαστούμε με την μοναδική Ελληνική Εταιρεία κατασκευής ασύρματων αισθητήρων, την Prisma Electronics ABEE. Η Prisma μας παρείχε το απαραίτητο υλικό hardware και software, ώστε να διαπιστώσουμε στην πράξη πως μπορεί να υλοποιηθεί ένα σύστημα παρακολούθησης φθοράς με ασύρματους αισθητήρες.

Στόχος μας ήταν αρχικά να κατανοήσουμε το υπάρχον σύστημα, πως αυτό δουλεύει και που μπορεί να βρει εφαρμογή. Επίσης έγινε μια διερεύνηση των υπαρχουσών τεχνολογιών και πρωτοκόλλων για ασύρματη επικοινωνία μεταξύ των αισθητήρων. Τέλος προσπαθήσαμε να εξελίξουμε το υπάρχον σύστημα της Prisma και να προσθέσουμε ενός είδους ευφυΐα στον κάθε κόμβο-αισθητήρα του ασύρματου δικτύου έτσι ώστε να περάσουμε από το αρχικό στάδιο της παρακολούθησης της φθοράς σε ένα “έξυπνο” σύστημα που θα προβλέπει τις αστοχίες των μηχανών και

θα είναι σε θέση να ειδοποιεί τον εκάστοτε ενδιαφερόμενο για το πότε πρέπει να αλλάξει ένα εξάρτημα ενός μηχανήματος.

Συνεισφορά της διπλωματικής

Η συνεισφορά της διπλωματικής αυτής περά από το θεωρητικό αλλά και πειραματικό κομμάτι της, είναι ένα από τα λίγα παραδείγματα που μπορεί να βρει κανείς όπου υπάρχει μια συνεργασία του Πολυτεχνείου της Πάτρας και πιο συγκεκριμένα του τμήματός μου των Μηχανικών Η/Υ και του Ελληνικού ιδιωτικού τομέα. Από την εμπειρία που κέρδισα αυτόν τον ένα χρόνο είδα ότι σε μια Ελλάδα που βρίσκεται ίσως στη δυσκολότερη, από οικονομικής απόψεως, κατάστασή της τα τελευταία χρόνια, υπάρχει ελπίδα. Αν στραφούμε στην τεχνολογία και δώσουμε βάση στην έρευνα και ανάπτυξη καινοτόμων τεχνολογικά προϊόντων, μπορούμε να είμαστε αισιόδοξοι για το μέλλον όλων μας.

Ανοίγει λοιπόν μια πόρτα ανάμεσα στο πανεπιστήμιο μας και τον Ιδιωτικό τομέα και πιο συγκεκριμένα την Prisma Electronics ABEE. Γίνεται παρουσίαση του λογισμικού και του προϊόντος της και μελλοντικά αυτό μπορεί να αποτελέσει εφαλτήριο για περαιτέρω ανάπτυξη της συγκεκριμένης πλατφόρμας δικτύων ασύρματων αισθητήρων και κατανόηση του πως δουλεύουν και βρίσκουν εφαρμογή τα ασύρματα δίκτυα αισθητήρων.

Επίσης παρουσιάζεται αναλυτικά το ασύρματο μοντέλο επικοινωνίας Zigbee που χρησιμοποιείται κατά κόρον σε δίκτυα ασύρματων αισθητήρων και γίνεται μια αναφορά αυτού στα υπόλοιπα πρωτόκολλα ασύρματης επικοινωνίας.

Τέλος αναπτύσσεται στη γλώσσα C μια βιβλιοθήκη για Fast Fourier Transformation και RMS (Root Mean Square) που επεκτείνει τις δυνατότητες της πλατφόρμας της

Prisma και βοηθάει στην ανάλυση των μετρήσεων και εξαγωγή συμπερασμάτων για τη φθορά των μηχανημάτων.

Δομή της Διπλωματικής

Σε αυτό το σημείο κρίνεται απαραίτητο να τονισθεί ότι αυτή η διπλωματική είναι αλληλένδετη με την διπλωματική του συμφοιτητή μου Γιώργου Γόντικα στην οποία παρουσιάζεται το θεωρητικό κομμάτι των ασύρματων δικτύων αισθητήρων στον τομέα της βιομηχανίας και όχι μόνο. Μαζί συνεργαστήκαμε για να μπορέσουμε να ολοκληρώσουμε τη διπλωματική μας και ιδέες και δουλειά του καθενός βρίσκονται και σε αυτή αλλά και στην δική του διπλωματική.

Στο 2ο κεφάλαιο γίνεται μια μικρή εισαγωγή για το Preventive Maintenance, η ιστορία του και η εφαρμογή του στο toolware monitoring όπου και επικεντρωνόμαστε εμείς με την πλατφόρμα Prismasense της Prisma Electronics ABEE

Στο 3ο κεφάλαιο παρουσιάζεται αναλυτικά η πλατφόρμα Prismasense και μερικές εφαρμογές της ενώ γίνεται αναλυτική παρουσίαση του Hardware αλλά και του Software της για να μπορέσει ο αναγνώστης να καταλάβει σε βάθος τον τρόπο λειτουργίας της.

Στο 4ο κεφάλαιο υπάρχουν tutorials για το την εγκατάσταση του δικτύου αισθητήρων της Prisma αλλά και για ανάπτυξη νέων embedded εφαρμογών δικτύων ασύρματων αισθητήρων.

Στο 5ο κεφάλαιο βουτάμε στα βαθιά παρουσιάζοντας την ανάγκη για τεχνικές ανάλυσης σημάτων έτσι ώστε να μπορεί κάποιος να εξάγει χρήσιμη πληροφορία από τις μετρήσεις των αισθητήρων. Περιγράφονται αναλυτικά οι βιβλιοθήκες που αναπτύχθηκαν για τον FFT και RMS.

Στο 6ο και τελευταίο κεφάλαιο παραθέτουμε τις σκέψεις, συμπεράσματα αλλά και μελλοντικές κατευθύνσεις των ασύρματων δικτύων αισθητήρων.

Κεφάλαιο 2 - Preventive Maintenance

Μία σημαντική εφαρμογή στη βιομηχανία αφορά στο Preventive Maintenance (PM) το οποίο περιλαμβάνει ενέργειες με στόχο τη συντήρηση των μηχανών και των εγκαταστάσεων σε ικανοποιητική λειτουργική κατάσταση καθώς και συστηματική επιθεώρηση και διόρθωση αστοχιών πριν εξελιχθούν σε μεγάλα προβλήματα. Η συντήρηση αυτή στηρίζεται σε διαρκείς ελέγχους, μετρήσεις διορθώσεις και αντικαταστάσεις εξαρτημάτων, που στοχεύουν στην αποφυγή δημιουργίας αστοχιών. Γενικότερα, το Preventive Maintenance βασίζεται σε αυστηρό προγραμματισμό της συντήρησης των εξαρτημάτων των μηχανών και των εγκαταστάσεων.

Ο ποίο αξιόπιστος τρόπος συντήρησης που εφαρμόζεται τουλάχιστον 5000 χρόνια είναι η PM. Γνωρίζουμε ότι οι αρχαίοι Αιγύπτιοι εκτελούσαν ελέγχους και συντηρήσεις στις πυραμίδες των φαραώ.

Είναι πάρα πολλά τα ιστορικά παραδείγματα εφαρμογής PM στο στρατό, στη ναυσιπλοΐα, στα κτίσματα κτλ. Ωστόσο τα πράγματα έμελλε να αλλάξουν με την εμφάνιση των ηλεκτρονικών υπολογιστών και την επανάσταση του πυριτίου τη δεκαετία του 1960. Συγκεκριμένα το 1965 δημιουργήθηκε ένα από τα πρώτα λογισμικά για την υποστήριξη της συντήρησης των μηχανών στην εταιρεία λιπαντικών MOBIL. Το πρόγραμμα αυτό είχε την ονομασία MIDEC και αποτέλεσε την απαρχή της κυριαρχίας των CMMSs.

Ώσπου, φτάνουμε την δεκαετία του ‘90 και το μοντέλο της PM τέθηκε υπό αμφισβήτηση μαζί και με όλα τα γενικότερα συστήματα οργάνωσης και διοίκησης των επιχειρήσεων. Ήταν η εποχή όπου εμφανίστηκαν νέοι ανταγωνιστές με πιο επιθετικές στρατηγικές, χωρίς κανόνες και περιορισμούς. Πρώτα από όλα τέθηκε πάλι επί τάπητος το ερώτημα. Ποιος ο βασικός στόχος της συντήρησης;

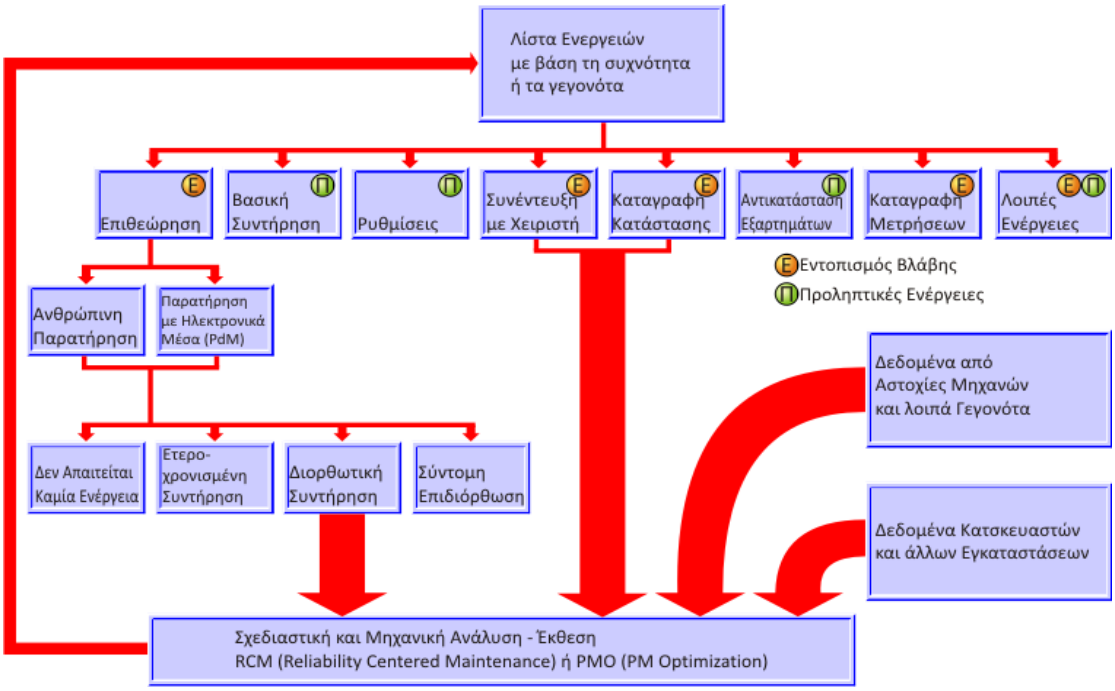
Το νέο δόγμα της συντήρησης, όπως παρουσιάζεται και στο βιβλίο του Joel Levitt, είναι: «Ο στόχος του τμήματος συντήρησης είναι η παροχή αξιόπιστων φυσικών αντικειμένων και άψογη υποστήριξη στους πελάτες, μειώνοντας και μερικώς εξαλείφοντας την ανάγκη για υπηρεσίες συντήρησης.»

Υπό αυτό το νέο δόγμα, οι σχεδιαστές των γραμμών παραγωγής κάθισαν στο ίδιο τραπέζι με τους συντηρητές, ώστε να γίνει καθολική επανασχεδίαση και να δοθούν νέες λύσεις και διαδικασίες.

Έτσι, σήμερα το μοντέλο συντήρησης που κυριαρχεί δεν είναι μόνο η PM αλλά PM και PdM. Δηλαδή, μία συντήρηση που δεν βασίζεται απλά στον σταθερό προγραμματισμό συντήρησης και αντικατάστασης εξαρτημάτων, αλλά βασίζεται στις πληροφορίες που συλλέγονται συνεχώς για την λειτουργία της εγκατάστασης. Οι πληροφορίες αυτές αναλύονται και επεξεργάζονται με διάφορους σύγχρονους αλγόριθμους, ώστε να προκύπτει ένα δυναμικό πρόγραμμα συντήρησης, που στοχεύει στην ελάχιστη δυνατή οικονομική επιβάρυνση του τελικού προϊόντος.

Η εξέλιξη του Predictive Maintenance (PM) είναι το Preventive Maintenance (PdM) ή Condition Based Maintenance. Με τη βοήθεια CMMS (Computerized Maintenance Management Systems), ο τρόπος αυτός συντήρησης βασίζεται σε διαδικασίες αξιολόγησης της κατάστασης του εξοπλισμού με περιοδική ή συνεχή παρακολούθηση αυτής. **Ο στόχος της PdM είναι να πραγματοποιηθεί μια συντήρηση όταν αυτή είναι οικονομικά πιο συμφέρουσα και πριν το εξάρτημα παύει να έχει τη βέλτιστη απόδοσή του.**

Στο παρακάτω διάγραμμα βλέπουμε πως συνδέονται οι λειτουργίες αυτές.



Η λίστα ενεργειών αποτελείται από προκαθορισμένες ή ασύγχρονες ενέργειες λόγω αλλαγής καταστάσεων. Οι ενέργειες αυτές χωρίζονται σε δύο κατηγορίες, στις ενέργειες πρόληψης και στις ενέργειες κατόπιν εντοπισμού μίας βλάβης. Με το αντίστοιχο σύμβολο δηλώνεται η κατηγορία στην οποία ανήκουν οι εκάστοτε ενέργειες. Συγκεκριμένα:

1. Με τον εντοπισμό της βλάβης πρέπει να εκτελεστεί επιθεώρηση των μηχανών, συνέντευξη με τον χειριστή, καταγραφή της κατάστασης και των μετρήσεων διαφόρων φυσικών μεγεθών.
2. Για προληπτικούς λόγους εκτελούνται βασικές συντηρήσεις, όπως λίπανση, διάφορες ρυθμίσεις, όπως σφίξιμο κοχλιών, και αντικατάσταση εξαρτημάτων.

Εκτός των παραπάνω ενεργειών μπορούν να υπάρχουν και άλλες κατηγορίες ανάλογα την ιδιαιτερότητα της παραγωγικής διαδικασίας.

Η επιθεώρηση των μηχανών βασίζεται στην ανθρώπινη παρατήρηση και στις καταγραφές διαφόρων παραμέτρων από ηλεκτρονικά όργανα μετρήσεων και αισθητήρες. Αυτό ακριβώς είναι και ο ορισμός της PdM.

Με την επιθεώρηση μπορούμε να αποφανθούμε είτε ότι δεν απαιτείται καμία ενέργεια, είτε ότι χρειάζεται μία συντήρηση εκτός προγραμματισμού, είτε ότι απαιτείται τροποποίηση της συντήρησης, είτε σύντομη επιδιόρθωση της μηχανής. Όλα τα δεδομένα, επισημάνσεις και συμπεράσματα των επιθεωρήσεων, παρατηρήσεις των χειριστών, καταγραφές των καταστάσεων, ιστορικό των μηχανών και στοιχεία των κατασκευαστών, συλλέγονται και αναλύονται στο κέντρο συντήρησης, ώστε να αναπροσαρμόζεται η λίστα των ενεργειών.

Όπως, είναι ευνόητο, αυτό το κέντρο συντήρησης θα πρέπει να έχει όσο το δυνατόν περισσότερες και ακριβέστερες πληροφορίες, στο ελάχιστο δυνατό χρονικό διάστημα.

Επειδή, η ανθρώπινη παρατήρηση είναι μεν η πιο σημαντική πηγή πληροφορίας, αλλά είναι υποκειμενική και όχι σε ξεκάθαρη μορφή ώστε να γίνει ψηφιακή επεξεργασία αυτής, το μοντέλο PdM εισάγει τη συνεχή ή μη παρατήρηση της κατάστασης των μηχανών με ηλεκτρονικούς αισθητήρες και διατάξεις. Η παρατήρηση αυτή, έχει σκοπό την συγκέντρωση πληροφοριών, ώστε να παράγονται αυτόματα συμπεράσματα για την κατάσταση της λειτουργίας των μηχανών.

Ζούμε στην ψηφιακή εποχή και όλα τείνουν να ψηφιοποιούνται, διότι οποιαδήποτε ψηφιακή πληροφορία αποθηκεύεται, επεξεργάζεται και διαδίδεται πολύ εύκολα με τα μέσα της εποχής μας.

Ομοίως και οι αισθητήρες έχουν μετατραπεί σε ψηφιακούς, δηλαδή το μετρούμενο φυσικό μέγεθος μετατρέπεται σε αναλογικό ηλεκτρικό σήμα και στη συνέχεια σε ψηφιακό, ώστε να σταλεί σε έναν ηλεκτρονικό υπολογιστή για αποθήκευση, ανάλυση και αναπαράσταση.

Η υλοποίηση της PdM, απαιτεί την χρήση ψηφιακών αισθητήρων για την παρακολούθηση διαφόρων φυσικών μεγεθών, που χαρακτηρίζουν την εκάστοτε κατάσταση μίας μηχανής.

Σήμερα, χρησιμοποιούνται ήδη αισθητήρες θερμοκρασίας, υγρασίας, κραδασμών, υπέρυθρων, ακουστικών συχνοτήτων και υπέρηχων, ενώ γίνεται προσπάθεια υλοποίησης αισθητήρων ανάλυσης των λιπαντικών. Διάφοροι αλγόριθμοι έχουν αναπτυχθεί και αναπτύσσονται διαρκώς για την επεξεργασία αυτών των μετρήσεων.

Το νέο μοντέλο συντήρησης απαιτεί την ύπαρξη ενός υπολογιστικού κέντρου που συλλέγει όλη τη διαθέσιμη πληροφορία την αποθηκεύει και την επεξεργάζεται. Δηλαδή, τα δεδομένα όλων των αισθητήρων θα πρέπει να μεταφέρονται σε ένα υπολογιστικό κέντρο. Στην πράξη μία τέτοια προσπάθεια συναντά ανυπέρβλητα εμπόδια, όπως η καλωδίωση και η μετάδοση του αναλογικού σήματος σε μεγάλες αποστάσεις, χωρίς την ύπαρξη θορύβου. Έτσι, καθώς ο ψηφιακός αυτοματισμός των μηχανών με PLCs, Industial Pcs κτλ, είναι πια δεδομένος στις γραμμές παραγωγής, υπάρχει κατάτμηση των αισθητήρων σε ημιανεξάρτητες υπολογιστικές μονάδες και η κάθε μονάδα στέλνει την ψηφιοποιημένη πληροφορία στην κεντρική βάση επεξεργασίας. Επίσης, επειδή ο όγκος των δεδομένων είναι μεγάλος και υπάρχει περιορισμός στη χωρητικότητα των δικτύων των υπολογιστών, προχωρήσαμε στην εκτέλεση μέρους της επεξεργασίας των μετρήσεων στις επιμέρους μονάδες και την αποστολή μόνο των αποτελεσμάτων αυτών στο κεντρικό σύστημα.

Δυστυχώς, στην εφαρμογή της θεωρίας στην πράξη, πάλι υπάρχουν πολλά προβλήματα, όπως:

1. Υπάρχει ακόμα μεγάλος αριθμός μηχανών που δεν διαθέτουν ενσωματωμένο σύστημα ψηφιακού ελέγχου ή ψηφιακούς αισθητήρες.
2. Η κάθε μηχανή έχει δικό της σύστημα ελέγχου και πρέπει να δημιουργηθούν λογισμικές εφαρμογές για το κάθε σύστημα, ώστε να υπάρχει συνεργασία των υποτμημάτων.
3. Δεν υποστηρίζουν όλα τα συστήματα κοινά πρωτόκολλα επικοινωνίας.
4. Σε περιπτώσεις, όπου απαιτούνται ειδικές μετρήσεις, δεν είναι εύκολη η διακοπή της λειτουργίας της μηχανής και η προσθήκη αισθητήρων σε αυτή.

5. Η τροποποίηση των μηχανών, ώστε να προσαρμοστούν σε νέες απαιτήσεις του PdM μοντέλου δεν είναι εύκολη και συχνά είναι δαπανηρή.

Για τους λόγους αυτούς και αρκετούς ακόμα, τους οποίους οι άνθρωποι της συντήρησης και της παραγωγής, γνωρίζουν καλλίτερα, υπήρξε τα τελευταία χρόνια η εισαγωγή του όρου «έξυπνος αισθητήρας».

Ένα δίκτυο έξυπνων αισθητήρων είναι ευρέως γνωστό με την ονομασία “SmartDust”.

Από το 2001, οπότε και ο Kristofer Pister εισήγαγε τον όρο SmartDust, έχουν γίνει μεγάλες τεχνολογικές προσπάθειες, ώστε να υλοποιηθεί αυτή η ιδέα. SmartDust ονομάζεται ένα σύνολο από διατάξεις μεγέθους κόκκου άμμου, οι οποίες περιλαμβάνουν ηλεκτρομηχανικούς αισθητήρες, διαθέτουν υπολογιστική μονάδα, επικοινωνούν ασύρματα και είναι ενεργειακά αυτόνομες.

Φυσικά, απέχουμε ακόμα από την πλήρη υλοποίηση του SmartDust. Ωστόσο, στην παρούσα χρονική στιγμή η τεχνολογία μας δίνει τη δυνατότητα υλοποίησης των πρώτων έξυπνων αισθητήρων με αποδεκτά χαρακτηριστικά και πολύ ενθαρρυντικά αποτελέσματα στις εφαρμογές αυτών.

Την τελευταία πενταετία υπήρξε ένα ξέσπασμα τεχνολογικών επιτευγμάτων στον τομέα των αισθητήρων MEMS, στον τομέα της ψηφιακής ασύρματης επικοινωνίας, στην παραγωγή επεξεργαστών χαμηλού κόστους και με πολύ χαμηλή κατανάλωση ισχύος και τέλος στην αποθήκευση και παραγωγή ηλεκτρικής ενέργειας από το φως, τη μηχανική κίνηση κτλ.

Οι εφαρμογές των έξυπνων αισθητήρων σήμερα είναι πολλές και σε διάφορους τομείς. Όμως, ιδιαίτερη συζήτηση, μελέτες και προσπάθειες γίνονται στον τομέα της Βιομηχανίας και συγκεκριμένα στις μεταφορές και στη συντήρηση. Με την εγκατάσταση ενός τέτοιου δικτύου αισθητήρων μπορεί να επιτευχθεί πραγματικά το μοντέλο του PdM.

Οι έξυπνοι αισθητήρες μπορούν να αποτελέσουν ένα εντελώς αυτόνομο δίκτυο συλλογής μετρήσεων, το οποίο θα παρακολουθεί τη κάθε μηχανή και το κάθε νευραλγικό σημείο της γραμμής παραγωγής, θα αναλύει τις μετρήσεις και θα εξάγει συμπεράσματα για την κατάσταση αυτών. Η πληροφορία αυτή θα είναι συνεχής, ενώ η τροποποίηση και προσαρμογή του συστήματος σε νέες συνθήκες και απαιτήσεις θα είναι απλή και ανέξοδη.

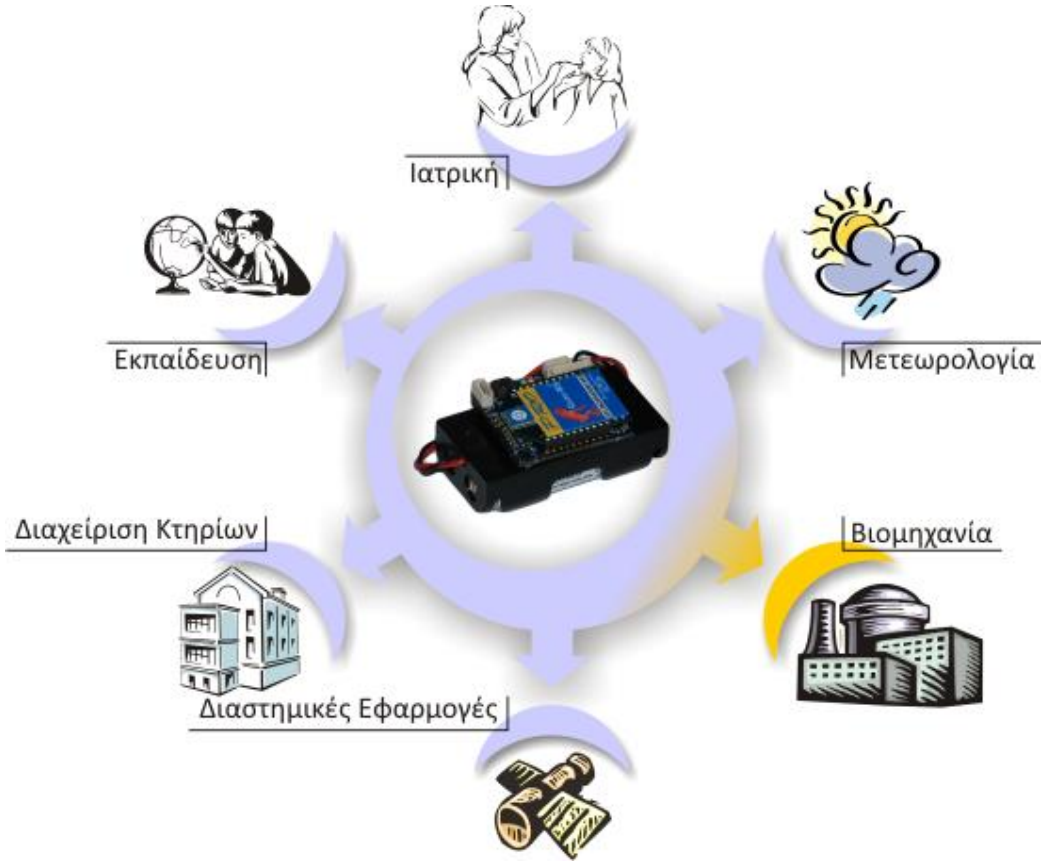
Το προσωπικό της συντήρησης θα έχει ένα εργαλείο που θα λειτουργεί και θα προγραμματίζεται εντελώς ανεξάρτητα από οποιοδήποτε σύστημα αυτοματισμού και ελέγχου του εργοστασίου και της εκάστοτε μηχανής. Ένα αυτόνομο σύστημα που θα το τροποποιεί , θα το επεκτείνει και θα το διαμορφώνει, χωρίς την παρακώληση της παραγωγικής διαδικασίας και την αναγκαστική εμπλοκή άλλων τμημάτων.

Τέτοια συστήματα θα αποτελούν στο άμεσο μέλλον, την ομπρέλα των γραμμών παραγωγής που θα εξασφαλίζουν τη συνεχή λειτουργία αυτών με το χαμηλότερο δυνατό κόστος.

Κεφάλαιο 3 - PrismaSense Platform

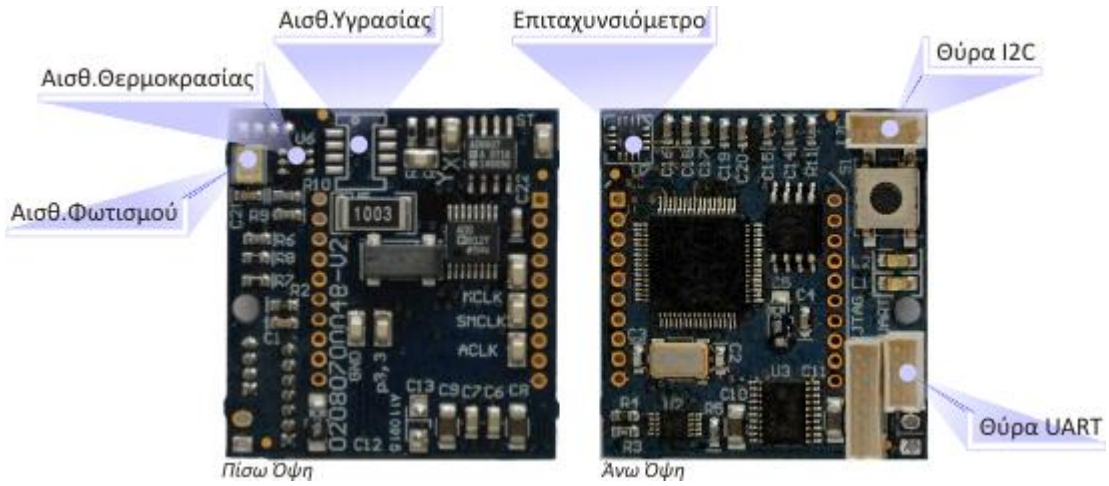
Τι είναι το PrismaSense

Το PrismaSense είναι μία πλατφόρμα ανάπτυξης εφαρμογών που βασίζεται σε έξυπνους αισθητήρες (Smart Sensors). Το πεδίο εφαρμογών μιας τέτοιας πλατφόρμας είναι αρκετά ευρύ, όπως φαίνεται και στην παρακάτω εικόνα, εμείς όμως θα εστιάσουμε σε εφαρμογές στη βιομηχανία.



PrismaSense Parts

Quax MS

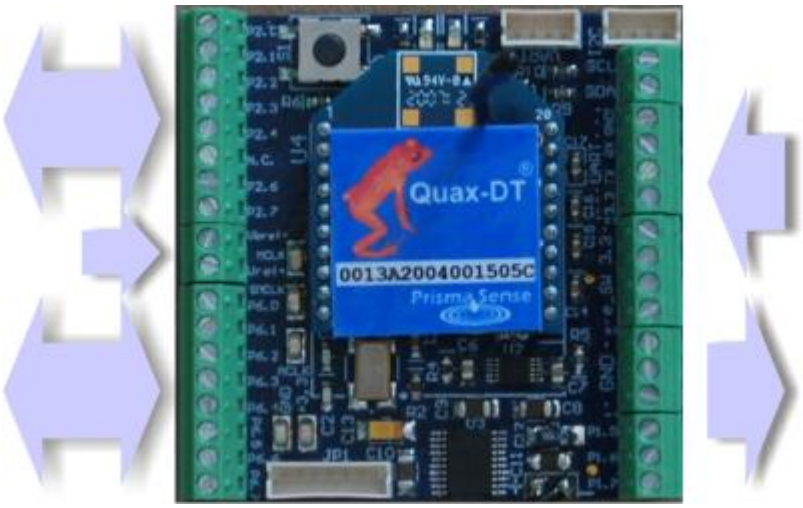


Το Quax-MS είναι ο βασικός πολυαισθητήρας του Prisma Sense. Έχει την δυνατότητα να λαμβάνει μετρήσεις να τις επεξεργάζεται, να τις αποθηκεύει και να στέλνει τα αποτελέσματα μέσω ασύρματης επικοινωνίας ZigBee. Επίσης, ανάλογα το λογισμικό του μπορεί να λαμβάνει εντολές για καθορισμό της λειτουργίας του.

- Έχει διαστάσεις 35mm x 33mm.
- Διαθέτει τον επεξεργαστή MSP430 της Texas Instruments στα 8MHz με 40KB Flash και 10KB RAM.
- Έχει ενσωματωμένους αισθητήρες φωτισμού, θερμοκρασίας, υγρασίας και επιταχυνσιόμετρο δύο αξόνων.
- Διαθέτει σειριακές θύρες I2C και UART για απευθείας σύνδεση με H/Y ή προσθήκη κι άλλων αισθητήρων.
- Διαθέτει RealTimeClock και ειδική διάταξη για ξεχωριστό έλεγχο της τροφοδοσίας των αισθητήρων από τον επεξεργαστή για την μείωση της καταναλισκόμενης ισχύος στο ελάχιστο δυνατό.
- Στην standard έκδοση τροφοδοτείται από δύο αλκαλικές μπαταρίες AA και η αυτονομία του εξαρτάται από το λογισμικό του. Σε κατάσταση ηρεμίας ξεπερνάει τα δύο χρόνια λειτουργίας χωρίς την αλλαγή των μπαταριών.
- Επιπλέον, διαθέτει εξωτερική μνήμη τύπου eeprom 512Kbit , για την αποθήκευση των μετρήσεων.

Διατίθεται με διάφορους πομποδέκτες ZigBee στα 2,4GHz ανάλογα το είδος της κεραίας, όπου διατίθεται με chip antenna, απλή διαφορική ή εξωτερική, και ανάλογα την ισχύ του σήματος εκπομπής, στα 1mW ή 10mW. Οι μετρούμενες αποστάσεις εκπομπής με ισχύ 10mW φτάνουν τα 300μέτρα σε εσωτερικό χώρο και 1χμ σε εξωτερικό.

Quax DT



Εκτός του QuaxMS στο Prisma Sense περιλαμβάνεται και μία συσκευή για την σύνδεση οποιουδήποτε εξωτερικού κυκλώματος αισθητήρων, το QuaxDT.

Έχει τα ίδια βασικά χαρακτηριστικά με τον πολυαισθητήρα QuaxMS με την διαφορά ότι δεν διαθέτει ενσωματωμένους αισθητήρες, αλλά παρέχει διάφορες θύρες για την σύνδεση εξωτερικών κυκλωμάτων αισθητήρων ή άλλων ηλεκτρονικών διατάξεων. Συγκεκριμένα διαθέτει:

1. 16 Digital I/O
2. 6 Inputs Analog to Digital Converter
3. 2 Outputs Digital to Analog Converter
4. 1 Uart Port
5. 1 SCI/I2C Port
6. 3 PWM Outputs

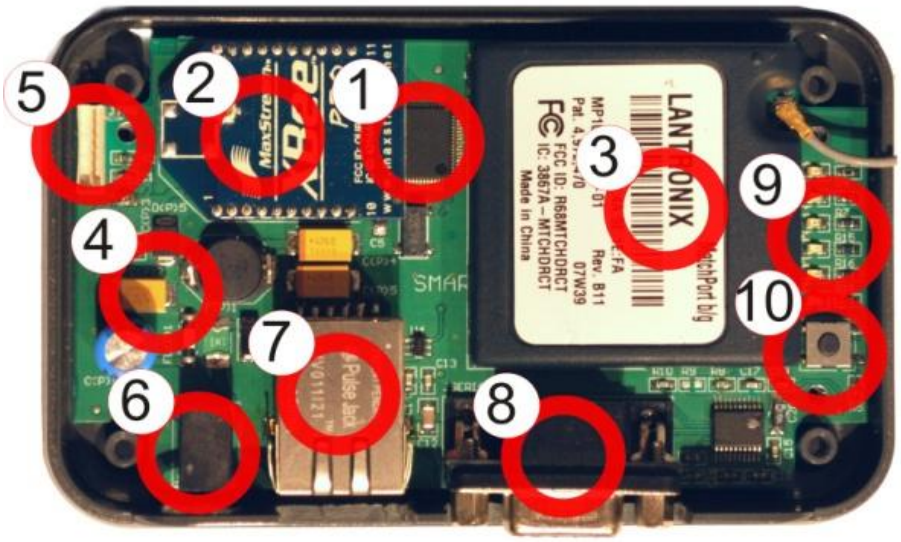
Η συσκευή αυτή μας δίνει δύο βασικές δυνατότητες:

1. Να δοκιμάσουμε την προσθήκη μίας νέας ηλεκτρονικής διάταξης στο σύστημα των έξυπνων αισθητήρων, η οποία θα είναι πλήρως λειτουργική και στη συνέχεια να προχωρήσουμε σε σχεδιασμό του νέου αισθητήρα.

2. Να συνδέσουμε με το σύστημα των έξυπνων αισθητήρων συσκευές μετρήσεων ή ελέγχου διαφόρων τύπων που πρέπει να συνδεθούν στο κεντρικό σύστημα συντήρησης.

Είναι ευνόητες οι δυνατότητες που μας δίνει το σύστημα για διασύνδεση με άλλα συστήματα και επέκταση αυτού.

Gateway



1. Επεξεργαστής
2. Module ZigBee
3. Module Wifi b/g
4. Switching Μετατροπέας Τάσης
5. Ακροδέκτης προγραμματισμού (JTAG)
6. Ακροδέκης τροφοδοσίας
7. Ακροδέκτης Ethernet (RJ-45)
8. Ακροδέκτης σειριακής RS232(DB9 Female)
9. Leds Ενδείξεων
10. Πλήκτρο επιλογής ενσύρματου ή ασύρματου δικτύου υπολογιστών

Ο ρόλος του PrismaSence Gateway είναι να δημιουργεί το ασύρματο δίκτυο των αισθητήρων και να εγκαθιστά επικοινωνία με τον κεντρικό εξυπηρετητή του κέντρου συντήρησης.

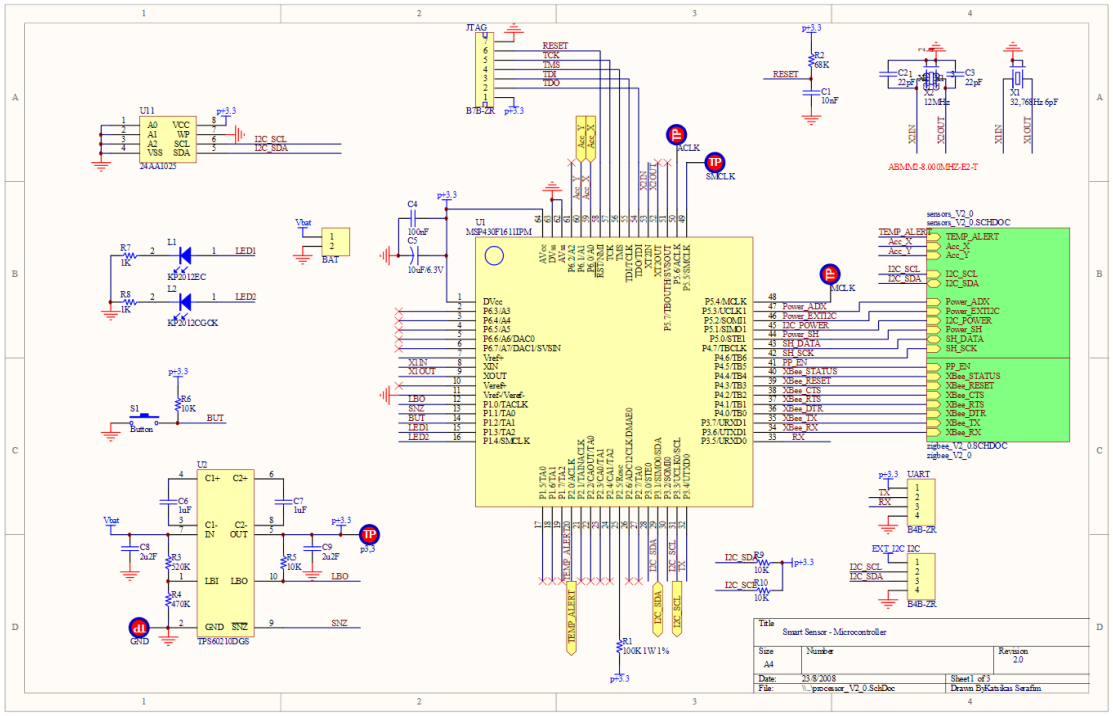
Επίσης αναλαμβάνει να διαχειριστεί το δίκτυο zigbee, δηλαδή αποφασίζει ποιες συσκευές θα συνδεθούν σε αυτό, διατηρεί τον πίνακα δρομολογήσεων των πακέτων, ελέγχει τον θόρυβο στο κανάλι εκπομπής και συγχρονίζει τις συνδεόμενες συσκευές.

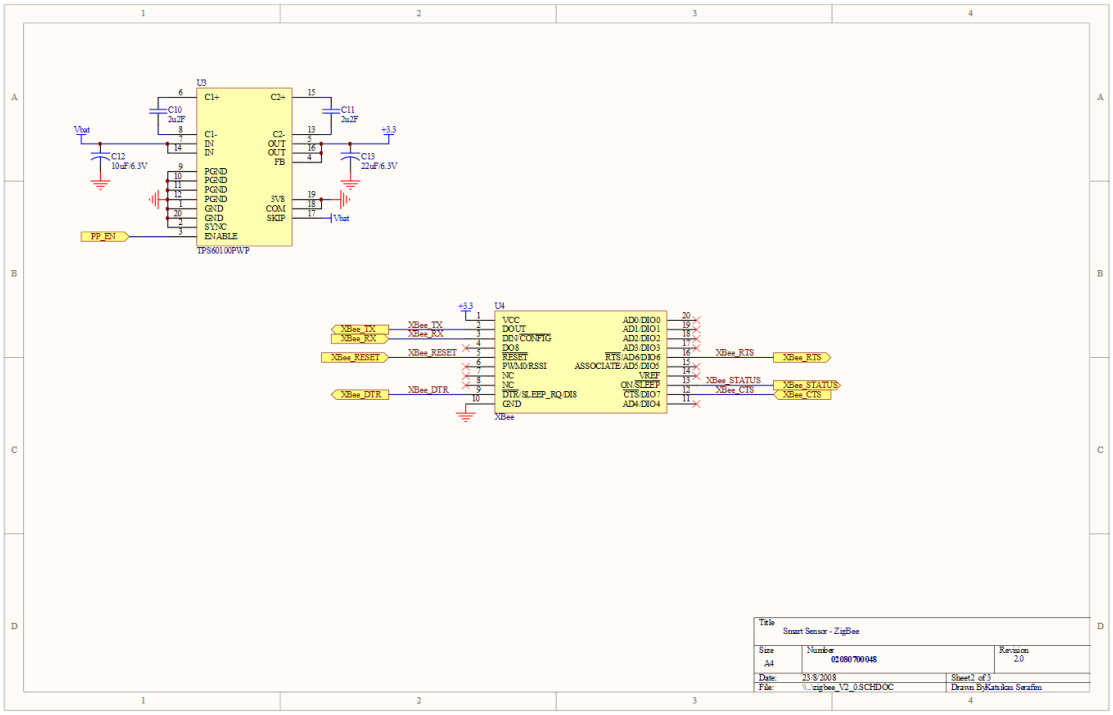
Επιπλέον, αναλαμβάνει να μεταφέρει όλη την πληροφορία που έχει ως παραλήπτη το κεντρικό σύστημα σε αυτό, με την σύνδεσή της είτε σε ασύρματο δίκτυο υπολογιστών WIFI 802.11b/g, είτε σε ενσύρματο δίκτυο Ethernet είτε σε απλή σειριακή επικοινωνία RS232. Υποστηρίζει και την αμφίδρομη επικοινωνία μεταξύ των αισθητήρων και του υπολογιστικού κέντρου.

Όπως, φαίνεται και στο διάγραμμα, πάνω αριστερά, η δομή του δικτύου ZigBee είναι πλήρως επεκτάσιμη και πολύ ευέλικτη. Απλά με την προσθήκη συσκευών μπορεί να επεκταθεί η κάλυψη του δικτύου πολύ εύκολα και με μικρό κόστος. Επίσης, το δίκτυο καθίσταται ακόμα πιο ευέλικτο καθώς δεν υπάρχει πρακτικός περιορισμός στην συνύπαρξη ξεχωριστών δικτύων ZigBee.

Συγκεκριμένα, μπορεί να γίνει διαχωρισμός του δικτύου των αισθητήρων σε επιμέρους δίκτυα ανάλογα το φυσικό αντικείμενο που ελέγχουν (μηχανές ή τμήματα παραγωγής). Για κάθε επιμέρους δίκτυο θα τοποθετηθεί ένα διαφορετικό gateway που θα υλοποιήσει το δικό του δίκτυο με μοναδικό κωδικό αριθμό σύνδεσης και σε διαφορετικό κανάλι εκπομπής. Αυτά τα gateways μπορούν αν συνδεθούν όλα στο υπολογιστικό κέντρο μέσω ενσύρματου ή ασύρματου δικτύου υπολογιστών.

Στις περιπτώσεις όπου υπάρχουν απομακρυσμένα σημεία μπορεί να συνδεθεί το gateway σε ένα GPRS Modem. Επίσης, ειδικά σε περιπτώσεις, όπως οι μεταφορές, οι αισθητήρες μπορούν να συλλέγουν τις μετρήσεις και να τις αποστέλλουν στο κεντρικό σύστημα μόλις συνδεθούν ξανά στο δίκτυο.


JTAG
RESET
TCK
TMS
TDI
TDO
B/B-ZR
U11
24AA1025
I2C_SCL
I2C_SDA
LED1
LED2
BAT
Vbat
BUT
U1
MSP430F1611IPM
ACLK
SMCLK
MCLK
U2
TPS60210DGS
GND
LBO
SNZ
U3
TPS60100PWP
PP_EN
Vbat
U4
XBee
XBee_DIN
XBee_DOUT
XBee_RESET
XBee_DTR
XBee_RTS
XBee_CTS
XBee_STATUS
sensors_V2_0
zigbee_V2_0
UART
EXT_I2C I2C
Title
Smart Sensor - Microcontroller
Size
A4
Number
Revision
2.0
Date:
23/8/2008
Sheet1 of 3
Drawn ByKatsikas Serafim


U3
TPS60100PWP
PP_EN
Vbat
U4
XBee
XBee_DIN
XBee_DOUT
XBee_RESET
XBee_DTR
XBee_RTS
XBee_CTS
XBee_STATUS
Title
Smart Sensor - ZigBee
Size
A4
Number
0208070048
Revision
2.0
Date:
23/8/2008
Sheet2 of 3
Drawn ByKatsikas Serafim

Harware

MSP430

Ο μικροελεγκτής που χρησιμοποιούμε είναι ο MSP430 της Texas Instruments. Είναι ένας 16 bit, mixed – signal[1] μικροελεγκτής και είναι σχεδιασμένος για embedded εφαρμογές χαμηλού κόστους και κυρίως, πολύ χαμηλής κατανάλωσης. Είναι επίσης κατάλληλος για εφαρμογές μετρήσεων, ασύρματη ραδιοεπικοινωνία και battery – powered εφαρμογές.

Εφαρμογές

Ο MSP430 είναι δημοφιλής επιλογή ανάμεσα στους κατασκευαστές embedded συσκευών χαμηλής κατανάλωσης. Η κατανάλωση ρεύματος σε idle mode μπορεί να φτάσει και κάτω από 1 microamp. Η μέγιστη συχνότητα της CPU είναι 25 MHz και μπορεί να μειωθεί για να πετύχουμε χαμηλότερη κατανάλωση. Χρησιμοποιούνται επίσης 6 διαφορετικά modes χαμηλής κατανάλωσης στα οποία υπάρχη δυνατότητα απενεργοποίησης ρολογιών και CPU. Αυτό επιτρέπει στον MSP430 να πέφτει σε sleep mode ενώ τα περιφερειακά του θα συνεχίζουν να λειτουργούν αναξάρτητα από την κατάσταση του επεξεργαστή ο οποίος καταναλώνει και μεγάλα ποσά ενέργειας. Επιπλέον, ο wake up χρόνος του ολοκληρωμένου είναι μικρότερος από 1 microsecond, επιτρέποντας έτσι στον μικροελεγκτή να μπορεί να παραμένει σε sleep mode για μεγαλύτερο χρονικό διάστημα επιτρέποντας έτσι την μείωση της μέσης κατανάλωσης. Σημειωτέον ότι η μετρική MHz είναι δαιφορετική από την Million instruction per second (MIPS), έτσι μπορούμε με διαφορετικές αρχιτεκτονικές να πετύχουμε δαιφορετικούς ρυθμούς MIPS σε μικρότερες

1 Περιέχει αναλογικά και ψηφιακά κυκλώματα στο ίδιο ολοκληρωμένο

συχνοτητες ρολογιού, το οποιο έχει ως αποτέλεσμα τη μείωση της δυναμικής κατανάλωσης για ισοδύναμη ποσότητα ψηφιακής επεξεργασίας.

Η συσκευή παρέχεται σε διάφορες εκδόσεις περιέχοντας τα παρακάτω περιφεριακά: Εσωτερικό ταλαντωτή, Timer με PWM[2], watchdog timer, USART, SPI, I2C, 10/12/14/16 – bit ADCs και brownout reset κύκλωμα. Κάποιες εκδόσεις περιέχουν συγκρητές, οι οποίοι μπορούν να χρησιμοποιηθούν μαζί με τους timers για να υλοποιήσουν απλές ADC, on – chip operation amplifiers για επεξεργασία σήματος, 12 – bit DAC, LCD driver, hardware multiplier, USB και DMA. Εκτός από τις παλιότερες EPROM και τις mask ROM, όλες οι συσκευές είναι προγραμματίσιμες μέσω JTAG ή μέσω bootstrap loader χρησιμοποιώντας RS-232.

Υπάρχουν, παρ' όλα αυτά, περιορισμοί που αποτρέπουν τη χρήση του σε πιο περίπλοκα embedded συστήματα. Ενα από τα κύρια μειονεκτήματα του MSP430 είναι ότι δεν έχει εξωτερικό δίαυλο μνήμης και έτσι περιορίζεται στην on – chip μνήμη (256 KB Flash Memory και 16 KB RAM) η οποία μπορεί να αποδειχτεί πολύ μικρή για εφαρμογές που απαιτούν μεγαλους buffers ή data tables. Επίσης, ενώ διαθέτει έναν αρκετά καλό DMA controller, είναι πολύ δύσκολο να χρησιμοποιηθεί για μεταφορά δεδομένων εκτός ολοκληρωμένου λόγω της έλειψης DMA output strobe.

Υπάρχουν πέντε σειρές μικροελεγκτών MSP430. Εμεις χρησιμοποιούμε τη σειρά 1xx η οποία είναι μία βασική έκδοση με ενσωματωμένο LCD controller. Προσφέρει 8 MIPS, λειτουργία 1.8 – 3.6 V, μέχρι 60 KB Flash, και ένα ευρύ φάσμα από ψηφιακά περιφεριακά.

2 Pulse – width modulation

Power Specs :

- 0.1 μA RAM retention
- 0.7 μA real-time clock mode
- 200 μA / MIPS active
- Feature Fast Wake-Up From Standby Mode in <6 μs

Device Parameters :

- Flash Options: 1–60 KiB
- ROM Options: 1–16 KiB
- RAM Options: 512 B–10 KiB
- GPIO Options: 14, 22, 48 pins
- ADC Options: Slope, 10 & 12-bit SAR
- Other Integrated peripherals: Analog Comparator, DMA, Hardware Multiplier, SVS, 12-bit DAC

MSP430 CPU

Το CPU του MSP430 χρησιμοποιεί von Neumann αρχιτεκτονική με ενιαίο χώρο διευθύνσεων για εντολές και δεδομένα. Η μνήμη είναι byte – addressed και ζευγάρια από bytes συνδυάζονται με little endian για να κατασκευασθούν 16 bit λέξεις.

Ο επεξεργαστής περιέχει 16 καταχωρητές των 16 bit. Ο R0 είναι ο program counter, R1 ο stack pointer, ο R2 status register και ο R3 ένας ειδικός καταχωρητής που ονομάζεται constant generator, ο οποίος δίνει πρόσβαση σε 6 συχνά

χρησιμοποιούμενες τιμές χωρίς να απαιτείται κάποιο περετέρω όρισμα. Οι καταχωρητές από R4 μέχρι R15 είναι διαθέσιμοι για γενική χρήση.

Το σύνολο εντολών είναι σχετικά απλό, υπάρχουν 27 εντολές σε 3 οικογένειες. Οι περισσότερες εντολές είναι διαθέσιμες σε 8 – bit και 32 – bit εκδόσεις, και χρησιμοποιούνται ανάλογα με το B/W bit. Αν αυτό είναι 1 τότε πρόκειται για 8 bit εντολή ενώ αν είναι 0 πρόκειται για 16 bit εντολή.

Accelerometer

Το onboard επιταχυνσιόμετρο που χρησιμοποιείται και θα μας απασχολήσει είναι το ADXL320 και κατασκευάζεται από την analog devices. Πρόκειται για ένα μικρό και λεπτό iMEMS επιταχυνσιόμετρο το οποίο μπορέι να μετρήσει από -5g έως και 5g σε δύο άξονες με μεγάλη ακρίβεια. Μπορεί να μετρήσει και δυναμικές επιταχύνσεις όπως δονήσεις αλλά και στατικές όπως αυτή της βαρύτητας.

Παρακάτω φαίνεται το λειτουργκό του διάγραμμα :

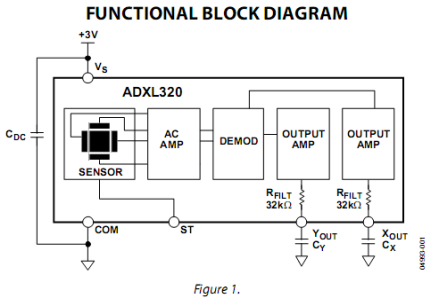


Figure 1.

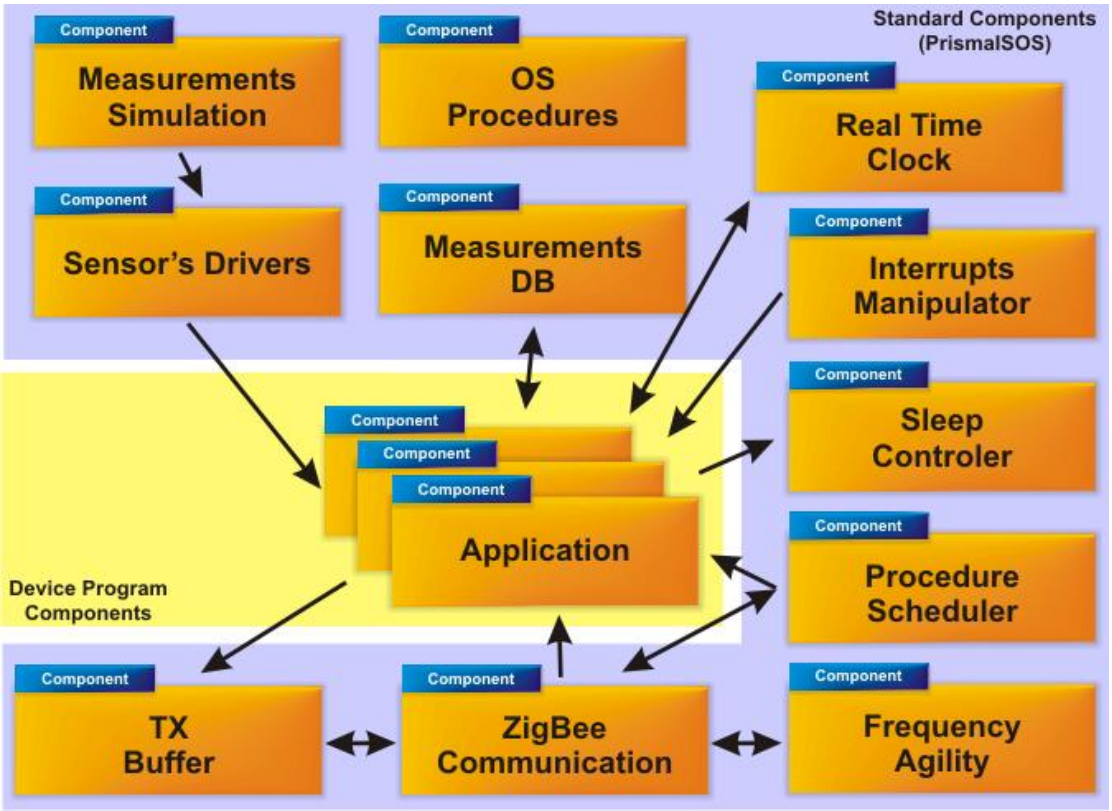
Software

Sensor OS

Το PrismaSense, περιλάμβανει και το κατάλληλο λογισμικό, το οποίο επιτρέπει τον προγραμματισμό του συστήματος. Το πρώτο μέρος του πακέτου αυτού περιλαμβάνει ένα λειτουργικό σύστημα ειδικά προσαρμοσμένο στους αισθητήρες Quax.

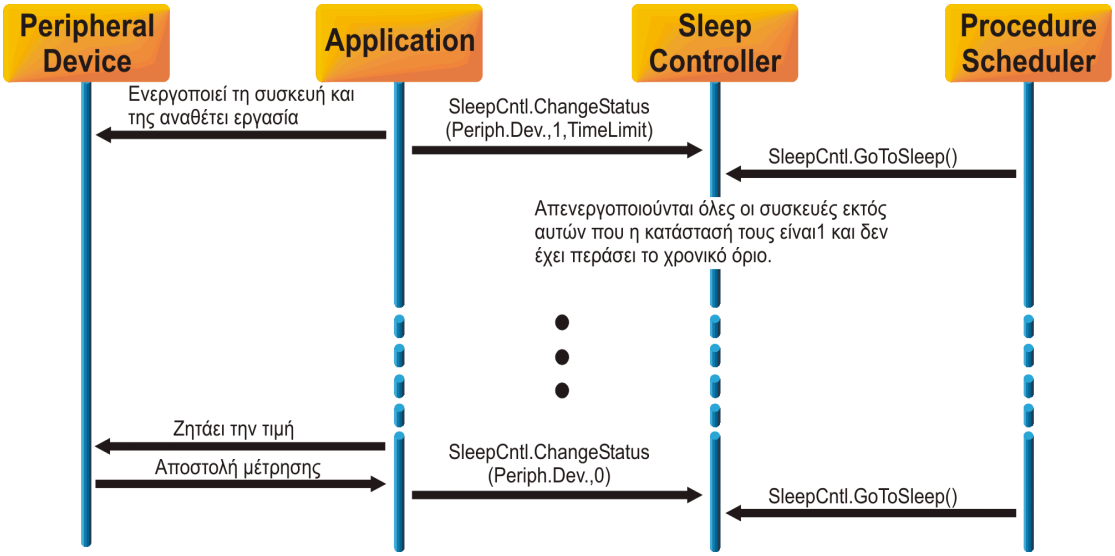
Δεν υπάρχει περιορισμός στην επιλογή λειτουργικού συστήματος αφού μπορούν να χρησιμοποιηθούν συστήματα όπως το MicroC, το TinyOS κτλ. Ωστόσο το ISOS (Intelligent Sensor Operating System είναι ειδικά προσαρμοσμένο στους ασύρματους αισθητήρες και ειδικότερα για βιομηχανικές εφαρμογές.

Το βασικό του πλεονέκτημα είναι ότι, με πολύ απλό τρόπο και λίγες γραμμές κώδικα μπορεί να δημιουργηθεί μία νέα εφαρμογή, χωρίς να μας απασχολεί ο τρόπος μετάδοσης των δεδομένων, η λειτουργία του επεξεργαστή για την εξοικονόμηση ενέργειας, η λήψη των μετρήσεων, ο συγχρονισμός του ρολογιού, η αξιοπιστία του λογισμικού κτλ.

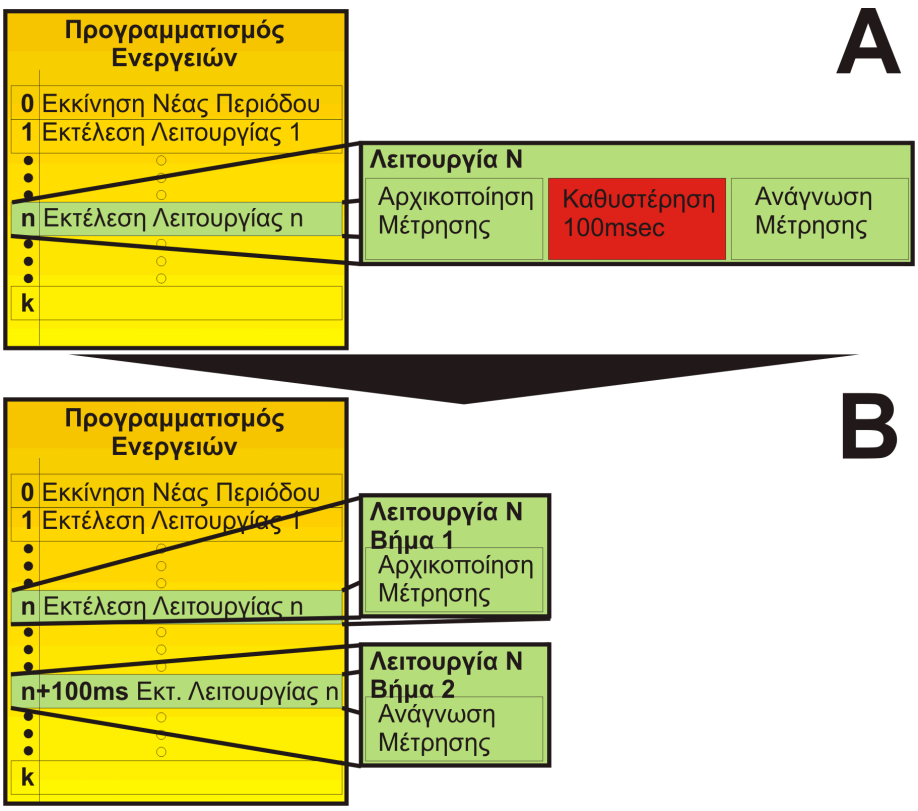


Παρακάτω βλέπουμε λίγο πιο συγκεκριμένα τη λειτουργία του λειτουργικού σε χαμηλότερα επίπεδα.

Η διεργασία που εκτελείται, ανάλογα με την εφαρμογή που εκτελείται (πχ ανάγνωση θερμοκρασίας, υγρασίας κλπ), ενεργοποεί την αντίστοιχη περιφεριακή συσκευή και της αναθέτει εργασία (πχ ανάγνωση θερμοκρασίας) αλλά και μεταφέρει στον Sleep Controller ένα σήμα αλλαγής κατάστασης της συγκεκριμένης περιφεριακής συσκευής στην τιμή 1 συνοδευόμενη από ένα χρονικό όριο μετά το πέρας του οποίου παύει να ισχύει η κατάσταση αυτή και η συγκεκριμένη συσκευή μπορί να μπει σε sleep mode. Ο Procedure Scheduler εν το μεταξύ στέλνει εντολή GoToSleep στον Sleep Controller, με την οποία απενεργοποιούνται όλες οι περιφεριακές συσκευές εκτός από αυτές που η κατάσταση τους είναι 1 και δεν έχει περάσει το χρονικό όριο που ορίσαμε. Μετά από κάποιο χρονικό διάστημα που ορίζουμε εμείς ώστε να διασφαλίσουμε τη ακαιρεότητα των δεδομένων της περιφεριακής συσκευής, εκτελέιται η διεργασία Ανάγνωση, κατά την οποία η εφαρμογή ζητάει από την περιφεριακή συσκευή την τιμή που πήρε από τον αισθητήρα μέτρησης και αφού λάβει την απάντηση, ειδοποιεί τον sleep controller ότι πλέον η κατάσταση της συγκεκριμένης περιφεριακής συσκευής είναι 0, το οποίο σημαίνει ότι μετά από την περιοδική εντολή του Scheduler, ο sleep Controller θα θέσει αυτή τη συσκευή σε sleep mode.



Ένα παράδειγμα της λειτουργίας του Scheduler φαίνεται παρακάτω. Εστω ότι υπέρχουν δύο διεργασίες, η «Αρχικοποίηση Μέτρησης» και η «Ανάγνωση Μέτρησης». Εστω ότι η Ανάγνωση πρέπει να εκτελεστεί 100 msec αργότερα από τη Αρχικοποίηση για να διασφαλιστεί η εγγυρότητα των δεδομένων. Η λίστα του Scheduler σ' αυτή την περίπτωση φαίνεται στην εικόνα που ακολουθεί.



Παρακάτω φαίνεται η λειτουργία μιας εφαρμογής (πχ αναγνωσης μέτρησης επιτάχυνσης) στο επίπεδο του Scheduler.

Συνάρτηση αρχικοποίησης εφαρμογής

- Αρχικοποίηση Μεταβλητών
- Εισαγωγή της F1 στον Scheduler
- Ενεργοποίηση του Sleep

Συνάρτηση F1

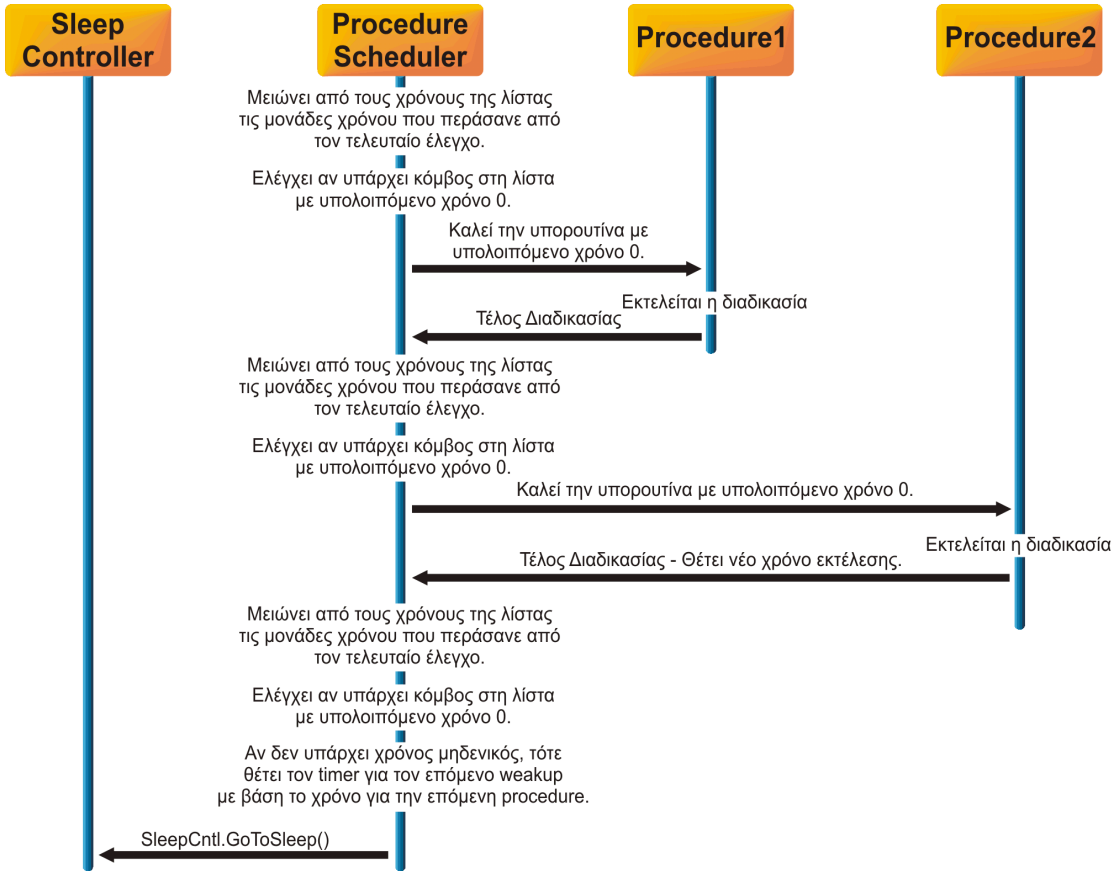
- Ελεγχος διαθεσιμότητας του I2C (μέσω Semaphores)
- Ενεργοποίηση Αισθητήρα
- Αρχικοποίηση Αισθητήρα
- Εντολή λήψης Μέτρησης
- Ελευθέρωση του semaphore και εισαγωγή της F2 στον Scheduler

- Ενεργοποίηση Sleep με τροφοδοσία στον αισθητήρα

Συνάρτηση F2

- Ελεγχος Διαθεσιμότητας του I2C (μέσω Semaphores)
- Λήψη Μέτρησης από Αισθητήρα
- Αποστολή πακέτου
- Εισαγωγή της F1 στον Scheduler σε χρόνο X
- Απενεργοποίηση Sleep. Ο επεξεργαστής θα μπει σε sleep mode όταν το πακέτο θα έχει σταλεί.

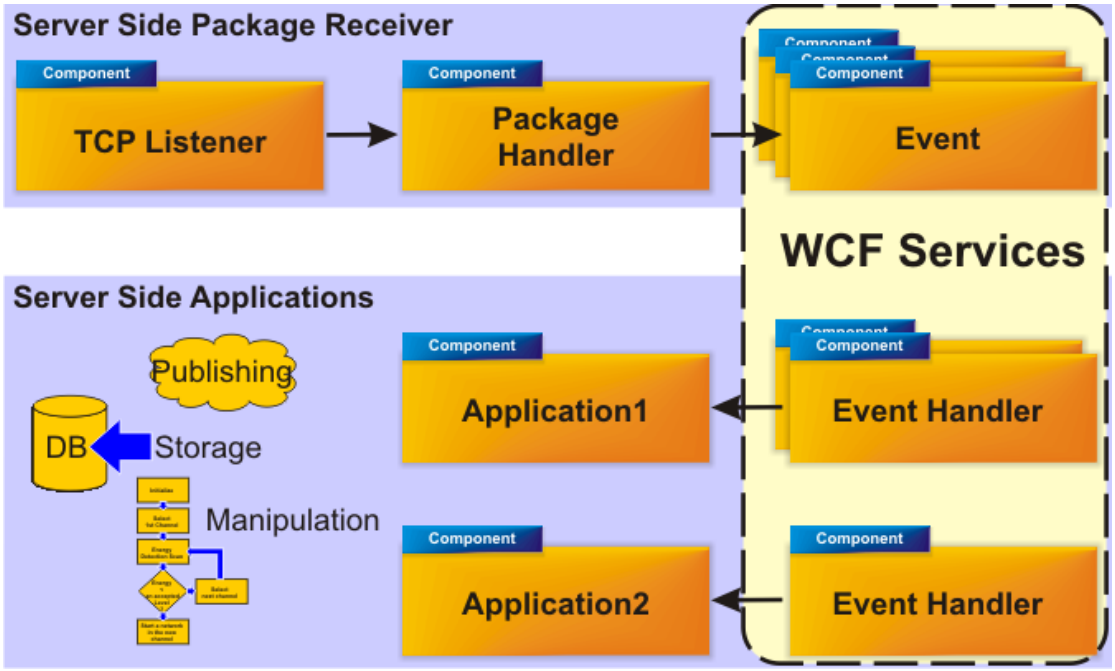
Αναλυτικότερα, ο Scheduler περιοδικά μειώνει από τους χρόνους της λίστας τις μονάδες χρόνου που πέρασαν από τον τελευταίο έλεγχο και ελέγχει αν υπάρχει κόμβος στη λίστα με υπολοιπόμενο χρόνο 0. Αν ναι, εκτελεί τη διεργασία που ορίζεται στο συκεκριμένο κόμβο. Αν όχι, τότε θέτει τον timer με wake up τιμή με βάση το χρόνο της επόμενης διαδικασίας. Παρακάτω φαίνεται η παραπάνω διαδικασία.



Server Side OS

Το δεύτερο καμμάτι του software του PrismaSense είναι το server side λογισμικό, το οποίο παρέχει εφαρμογές σε WCF Services, ώστε από οποιοδήποτε υπολογιστή συνδεδεμένο στο δίκτυο του server να μπορεί να αποστέλλει και να λαμβάνει πακέτα από τους ασύρματους αισθητήρες. Επίσης, παρέχεται εφαρμογή

αποθήκευση των δεδομένων και των μετρήσεων σε MIMOSA βάση δεδομένων και εφαρμογή για εξαγωγή των δεδομένων σε excel.



Εφαρμογές της πλατφόρμας PrismaSense

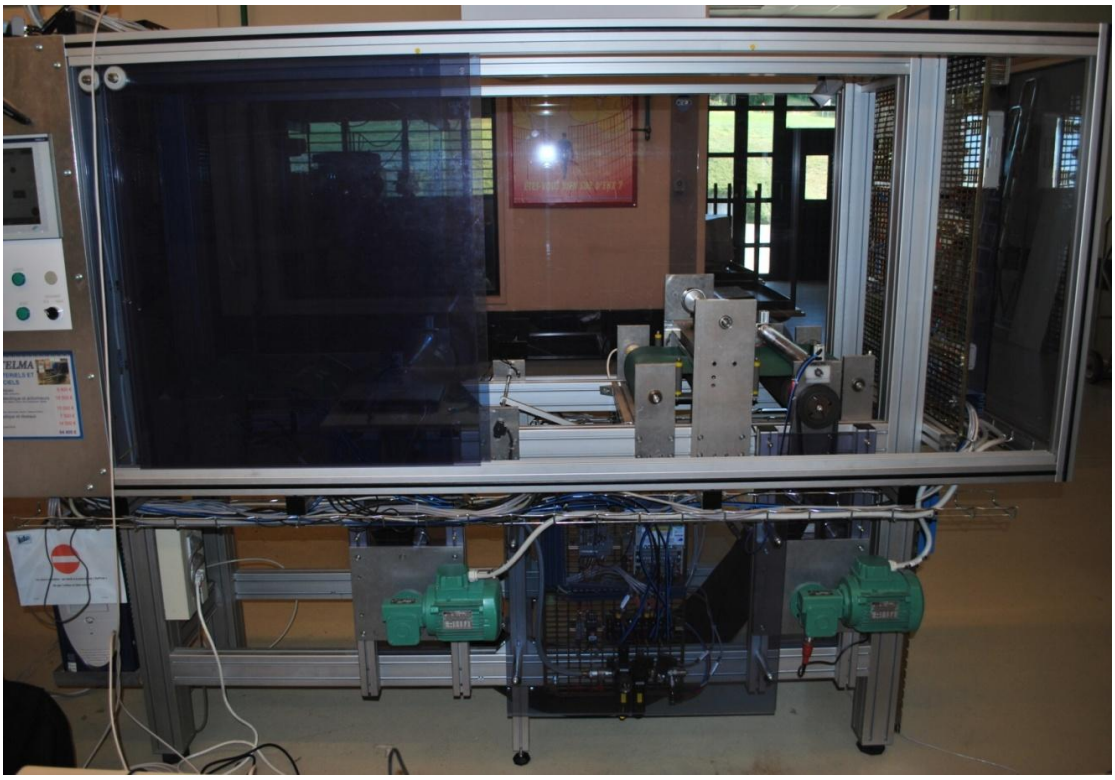
Εφαρμογή στο Telma Platform

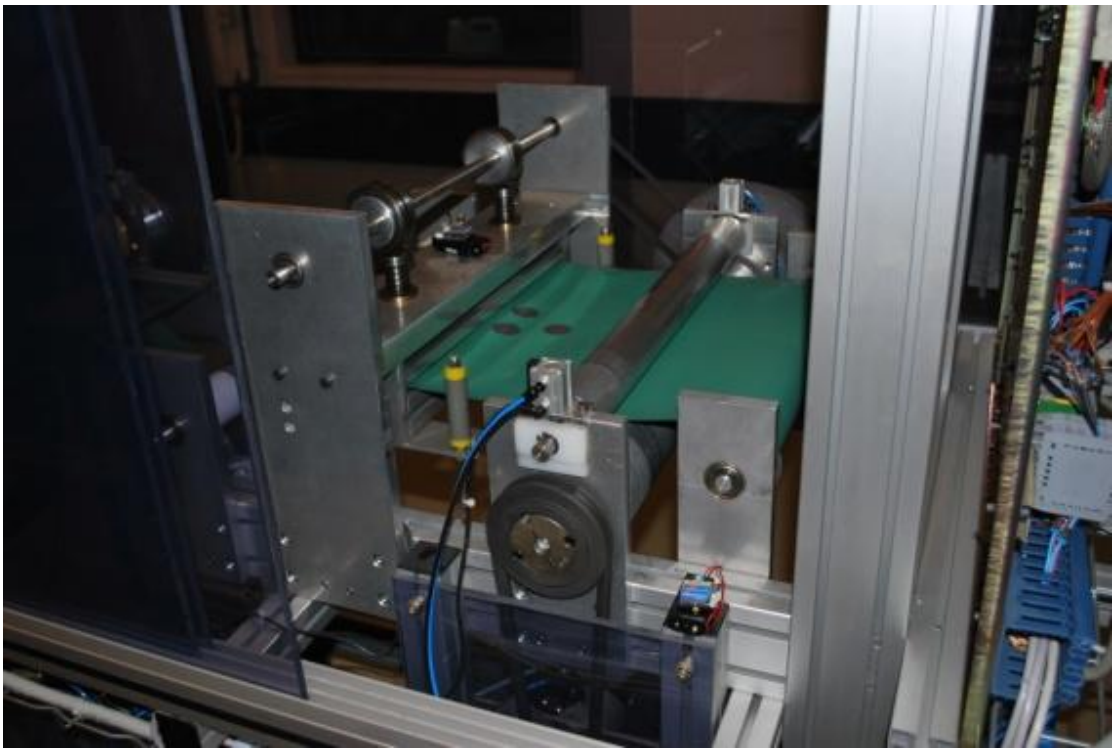
Στα πλαίσια ενός ερευνητικού προγράμματος το PrismaSense εγκαταστάθηκε και δοκιμάσθηκε στο Πανεπιστήμιο της Nancy στη Γαλλία, όπου ειδικεύεται στον έλεγχο μηχανών παραγωγής και σε αλγόριθμους υπολογισμού της ζωής αυτών. Συγκεκριμένα εγκαταστάθηκε σε μία μηχανή παραγωγής συγκεκριμένων εξαρτημάτων της αυτοκινητοβιομηχανίας Peugeot και συνδέθηκε μέσω μία βάσης δεδομένων τύπου MIMOSA, με το σύστημα ελέγχου και συντήρησης TELMA, που αναπτύσσεται από το πανεπιστήμιο.

Οι αισθητήρες μετρούν την θερμοκρασία των κινητήρων της μηχανής, τους κραδασμούς σε ένα έμβολο πρέσας και τη γωνία ενός άξονα που μετράει την πίεση του ιμάντα μεταφοράς.

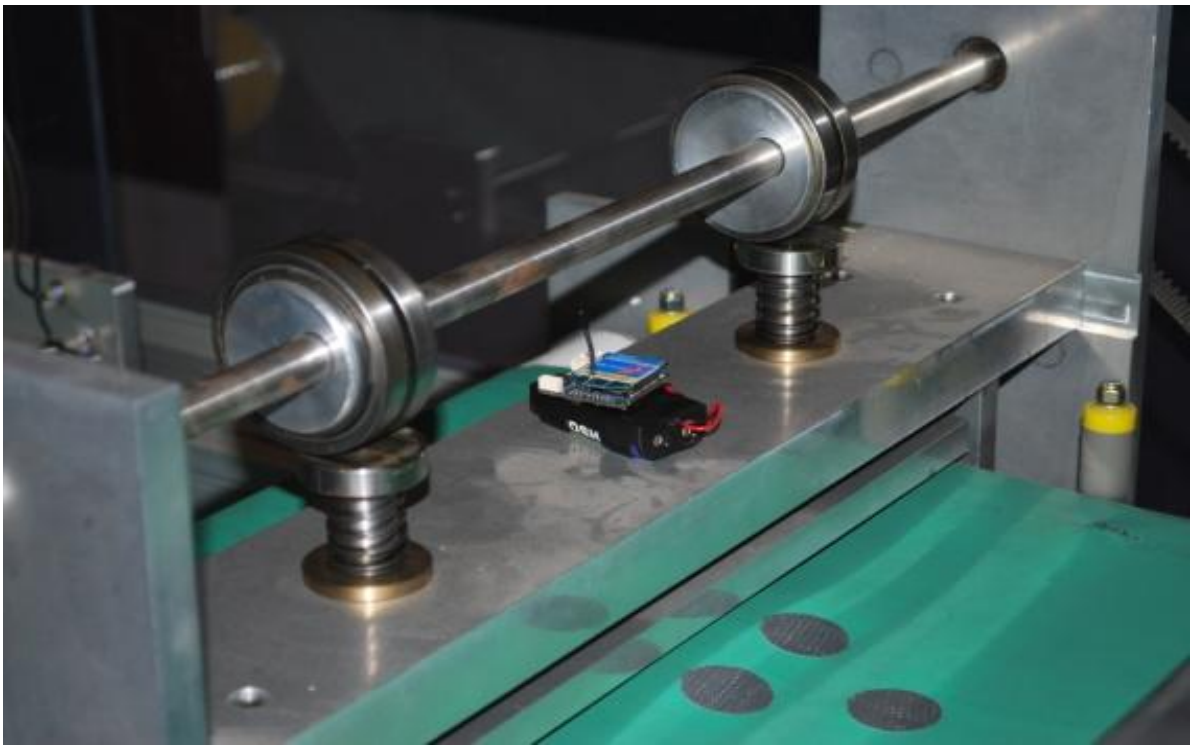
Οι μετρήσεις αποστέλλονται στη βάση δεδομένων μέσω ασύρματου WiFi δικτύου και αναλύονται από το λογισμικό Telma, ώστε να εξαχθούν αυτόματα συμπεράσματα για την κατάσταση της μηχανής και των εξαρτημάτων.

Παρακάτω μπορούμε να δούμε κάποιες φωτογραφίες από την εφαρμογή αυτή.





Αισθητήρας που μετρά τη θερμοκρασία του κινητήρα

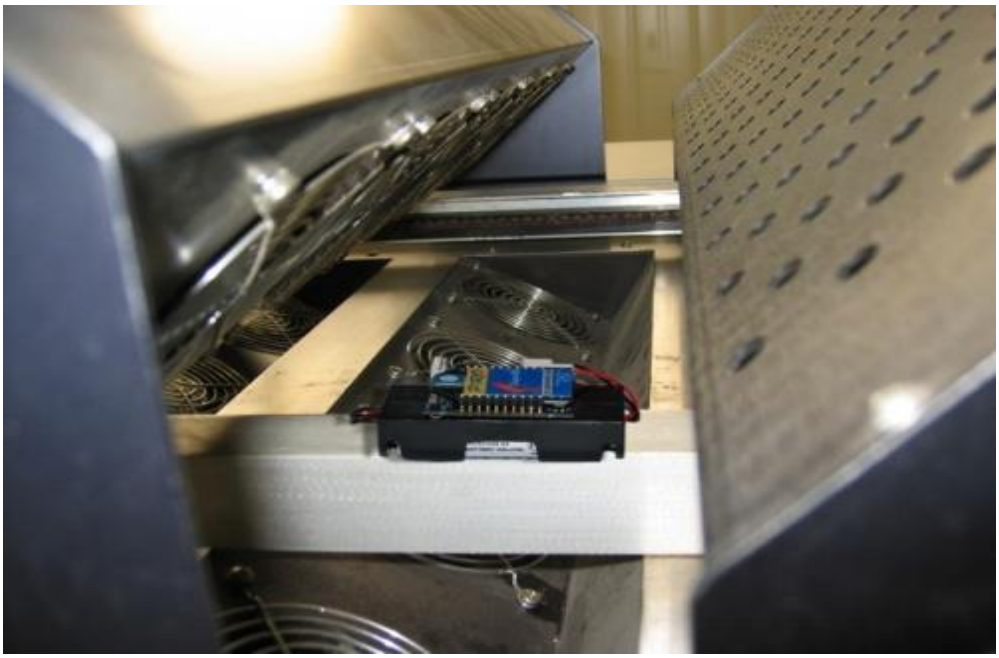


Αισθητήρας που μετρά του κραδασμούς στο έμβολο

Αισθητήρας που μετρά τη γωνία ενός άξονα για να μπορεί να υπολογίσει την πίεση του ιμάντα μεταφοράς.

Εφαρμογή σε Φούρνο Συγκόλλησης SMT

(Πρόκειται για μία μέθοδος κατασκευής ηλεκτρονικών κυκλωμάτων κατά την οποία τα εξαρτήματα (SMCs, or Surface Mounted Components), συγκολούνται απευθείας στην επιφάνεια των PCBs (Printed Circuit Boards).



Στην εφαρμογή αυτή οι αισθητήρες είναι επικαλυμμένοι με ειδική διάφανη ρητίνη για την προστασία τους από την θερμοκρασία και τοποθετήθηκαν στις διάφορες ζώνες του φούρνου. Με τον τρόπο αυτό καταγράφονται συνεχώς κατά την διάρκεια της λειτουργίας αυτού, η ακριβής θερμοκρασία και επίπεδο υγρασίας στην κάθε ζώνη. Αυτοί οι δύο παράγοντες είναι καθοριστικοί για την σωστή συγκόλληση των εξαρτημάτων. Οι καταγραφές αποστέλλονται σε έναν εξυπηρετητή και αποθηκεύονται στο ιστορικό της παραγωγής. Επίσης, οποιαδήποτε απόκλιση από την επιθυμητή τιμή, οδηγεί σε επισταμένο έλεγχο της μηχανής και επαναρύθμιση ή συντήρηση αυτής. Φυσικά είναι ευνόητο ότι μία και μόνο αστοχία της μηχανής ή απόκλιση σε κάποια ζώνη λίγων βαθμών κελσίου, μπορεί να καταστρέψει μία ολόκληρη παραγωγή ηλεκτρονικών με μεγάλο κόστος οικονομικό και όχι μόνο.

Χρησιμοποιώντας αισθητήρες μειώνονται στο ελάχιστο τέτοιες περιπτώσεις, διότι το σφάλμα θα καταγραφεί έγκαιρα.

Κεφάλαιο 4 - PrismaSense Tutorials

Η πλατφόρμα PrismaSence αποτελείται από τα παρακάτω :

1. 1 WiFi Gateway
2. 1 Quax DT (temperature) sensor
3. 1 Quax MS (temperature, humidity, light) sensor
4. 1 Quax MS (temperature ,acceleration, light) sensor
5. 1 UART converter (TTL to RS232)
6. 2 antennas
7. 1 Gateway power supply
8. 1 external temperature/humidity sensor
9. 1 JTAG adapter
10. 1 Gateway wall mounting kit
11. 6 AA batteries
12. 1 CD-ROM containing software and documents

Εγκατάσταση απαραίτητου λογισμικού

Για να μπορέσει κανείς να προγραμματίσει στην πλατφόρμα PrismaSence θα πρέπει να προμηθευτεί με το Code Composer TM Essentials Professional V2.0 της Texas Instuments. Το συγκεκριμένο λογισμικό είναι απαραίτητο για embedded programming αφού περιέχει κατάλληλο compiler για να μπορεί ο κώδικάς μας να μεταφραστεί σωστά για τον επεξεργαστή MSP430 που χρησιμοποιείται στην πλατφόρμα της Prisma. Μάλιστα από το site της Texas Instruments μπορεί κανείς να κατεβάσει δωρεάν για 30 μέρες το καινούργιο Code Composer v4.0 το οποίο όμως έχει compatibility issues με τον τρέχον κώδικα γι΄αυτό πειράξαμε 2 DLLs με τα οποία μπορεί κανείς να λειτουργήσει ακόμα με το Code Composer 2.

Αφού εγκαταστήσουμε το λειτουργικό Code Composer πρέπει να εγκαταστήσουμε και κάποια επιπλέον προγράμματα έτσι ώστε να είμαστε σε θέση να λάβουμε τις μετρήσεις από τους αισθητήρες στον υπολογιστή μας, χρησιμοποιώντας το Server Program της Prisma.

Ακολουθούν αναλυτικές οδηγίες για το ποια προγράμματα πρέπει να εγκατασταθούν για να μπορέσουμε να εργαστούμε.

1. MaxStream X_CTU, with all the firmware updates
2. Lantronix Device Installer version 4.2.0.0 or newer
3. Lantronix CprDotNetDL_Web version 4.1.0.2 or newer
4. Prisma's PS_ServerDataManipulation

Αφού τελειώσουμε με το installation των παραπάνω, για να χρησιμοποιήσουμε το γραφικό περιβάλλον του προγράμματος PS_ServerDataManipulation και να μπορούμε να εξάγουμε τα δεδομένα στο excel ή σε mimosa database θα πρέπει να δηλώσουμε την διεύθυνση των αισθητήρων στο quaxes.xml αρχείο που έχει δημιουργηθεί στο κατάλογο My Computer→C→Program Files→Prisma Electronics SA→PS_SDM→ Quaxes.xml μετά το installation του PS_ServerDataManipulation.

Για να το κάνουμε αυτό ανοίγουμε το αρχείο quaxes.xml με κάποιον text editor και αλλάζουμε τις υπάρχουσες διευθύνσης με τα δεκαεξαψήφια serial numbers που βρίσκονται στην πάνω μεριά του εκάστοτε αισθητήρα :

```
<?xml version="1.0" encoding="utf-8"?>
<sensors name="formal_blue">
 <quax addr="0013A20040534F41" onimg_pos="283,616" sec_limit="20">
  Qaux DT
  <sensor sensor_type="T" meas_loc_site="6789000200000002" meas_loc_id="24"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="41"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
 </quax>
 <quax addr="0013A20040534F4B" onimg_pos="328,32" sec_limit="20">
  Quax MS - Temperature, Humidity, Light
  <sensor sensor_type="T" meas_loc_site="6789000200000002" meas_loc_id="25"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="41"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
  <sensor sensor_type="H" meas_loc_site="6789000200000002" meas_loc_id="25"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="41"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
  <sensor sensor_type="L" meas_loc_site="6789000200000002" meas_loc_id="25"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="41"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
 </quax>
 <quax addr="0013A20040534F3E" onimg_pos="229,108" sec_limit="20">
  Accelerometer
  <sensor sensor_type="AX" meas_loc_site="6789000200000002" meas_loc_id="26"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="83"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor><sensor
sensor_type="AY" meas_loc_site="6789000200000002" meas_loc_id="27"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="83"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
  <sensor sensor_type="AY" meas_loc_site="6789000200000002" meas_loc_id="26"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="83"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor><sensor
sensor_type="AY" meas_loc_site="6789000200000002" meas_loc_id="27"
eu_db_site="0000000000000000" eu_db_id="0" eu_type_code="83"
mc_db_site="6789001300000013" mc_db_id="113" mc_type_code="4"></sensor>
 </quax>
</sensors>
```

Στο παραπάνω παράδειγμα δηλώνουμε τρεις αισθητήρες.

- Ο πρώτος έχει διεύθυνση *0013A20040534F41* και μετράει θερμοκρασία. (temperature sensor)
- Ο δεύτερος έχει διεύθυνση *0013A20040534F4B* και μετράει θερμοκρασία, υγρασία και ποσότητα φωτός. (temperature, humidity and light sensor)
- Ο τρίτος έχει διεύθυνση *0013A20040534F3E* και μετράει επιτάχυνση. (acceleration sensor)

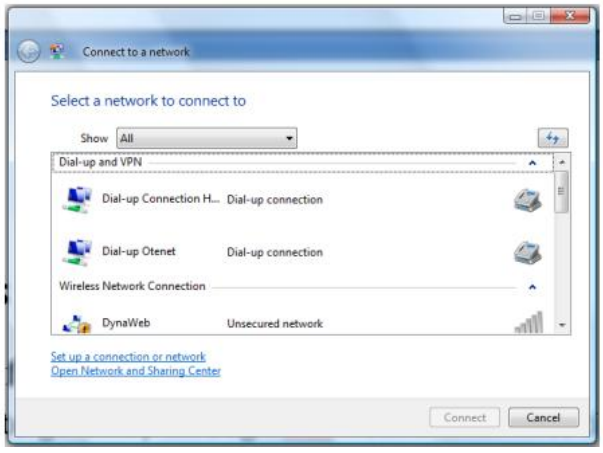
Σύνδεση του Gateway με τον Router

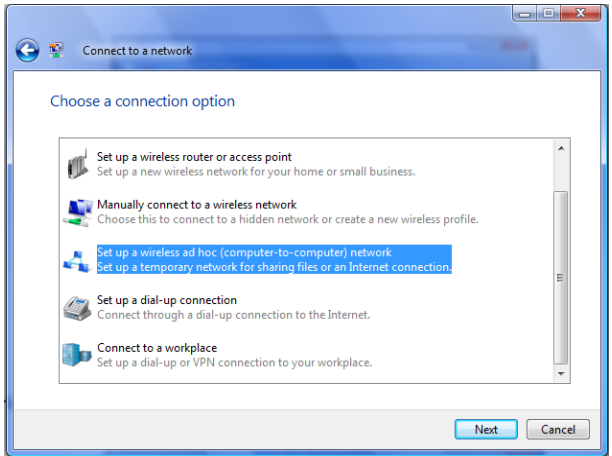
Έπειτα για να μπορέσουμε να συνδέσουμε το gateway με τον υπολογιστή μας μέσω wifi για πρώτη φορά, είναι απαραίτητο να δημιουργήσουμε ένα νέο ad-hoc δίκτυο και μετά να συνδέσουμε το gateway για να αλλάξουμε τις ρυθμίσεις.

Τα βήματα παρουσιάζονται αναλυτικά παρακάτω :

1. **Αρχικά δημιουργούμε ένα νέο ad-hoc wireless network με ένα Access Point ή laptop. Με όνομα LTRX_IBSS και security type : open.**

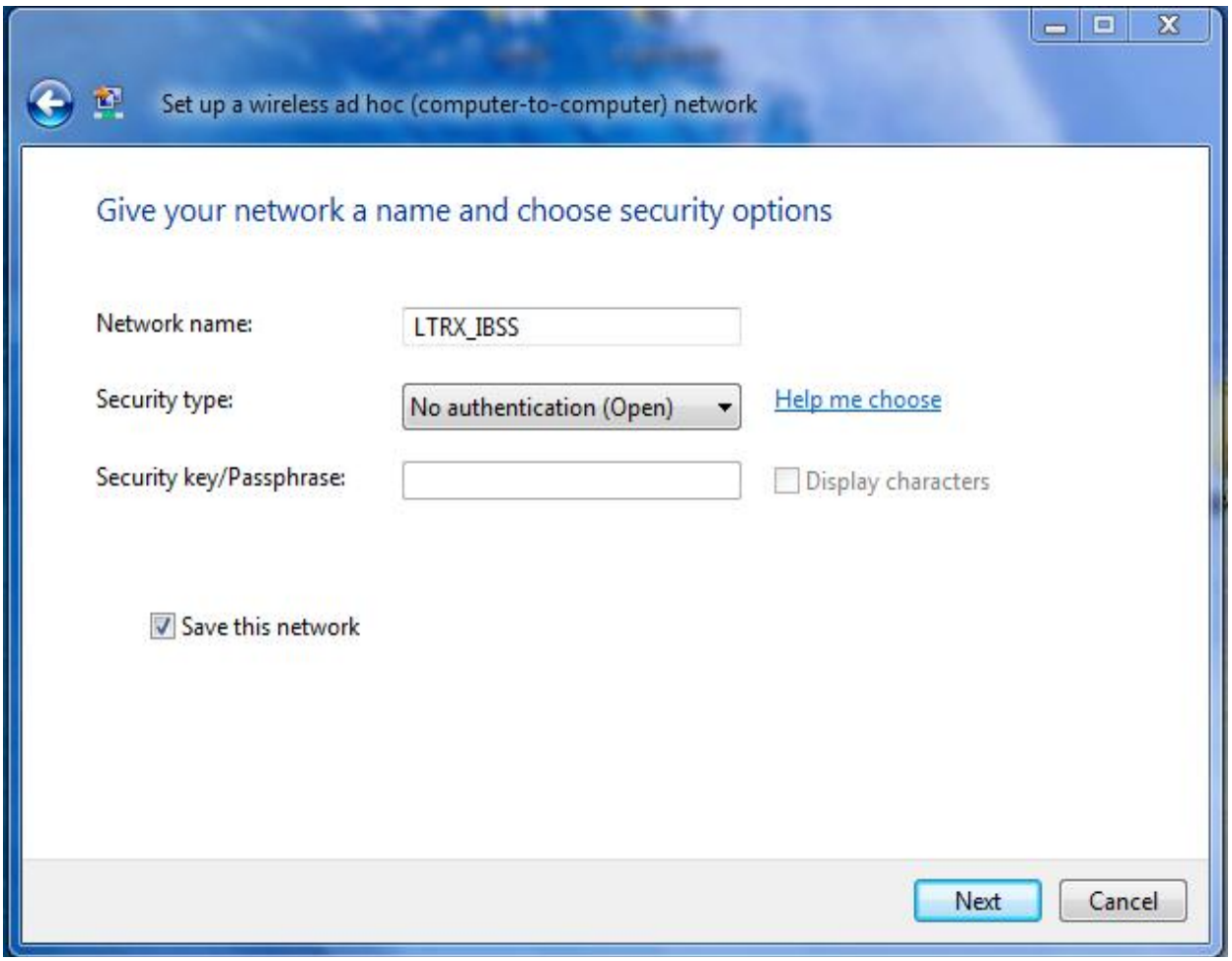
   Για να κάνουμε κάτι τέτοιο αρχικά ανοίγουμε το Connect to a Network Dialogue



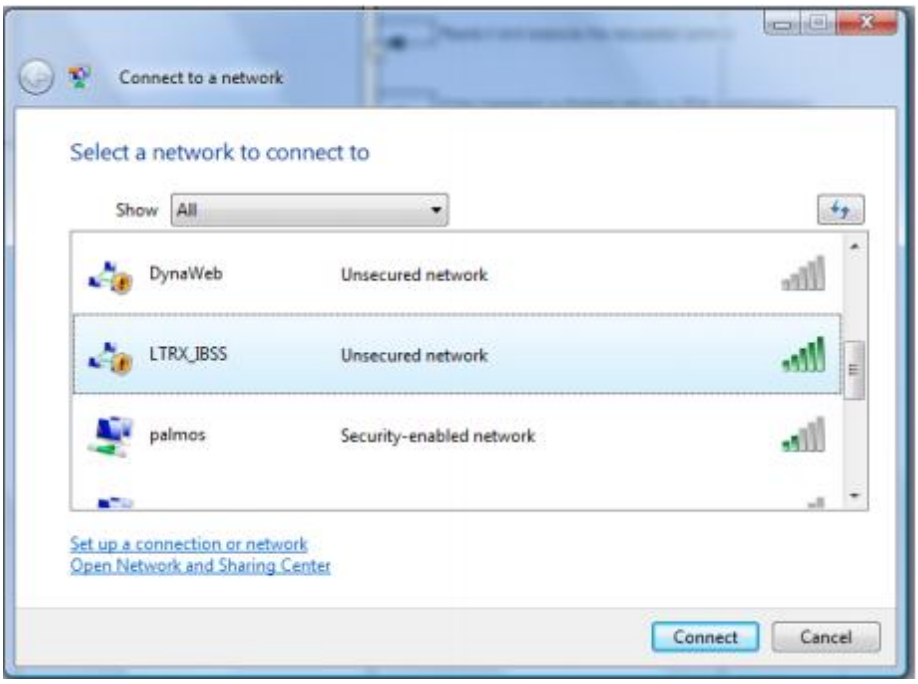


Έπειτα πατάμε Set up a connection or network.

Στα settings του δίνουμε το όνομα LTRX_IBSS και security type no authentication



Επιλέγουμε το δίκτυο που φτίακαμε και πατάμε connect



2. **Συνδέουμε το gateway σε μια πηγή ρεύματος**

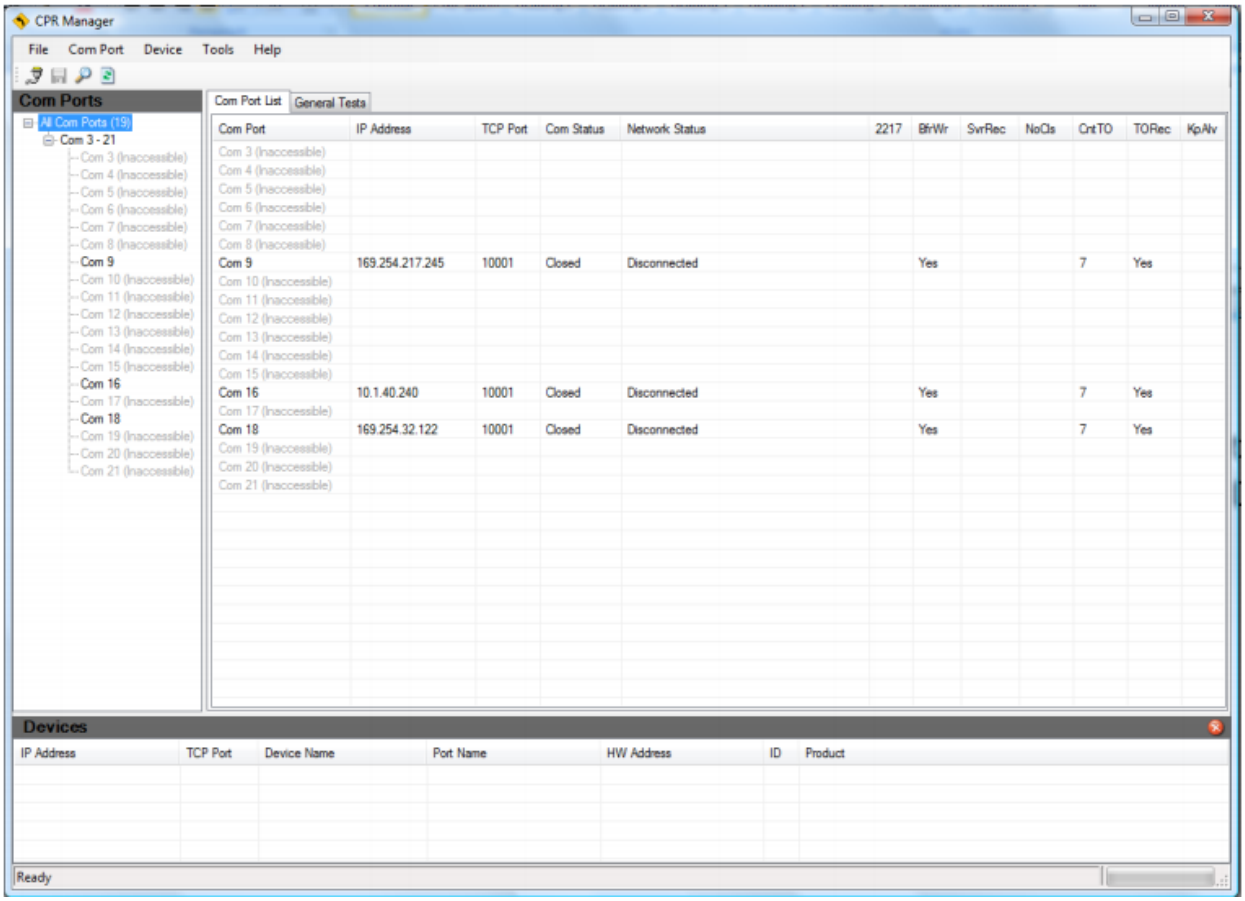
Το τρίτο led πρέπει να παραμείνει αναμένο αν το gateway έχει συνδεθεί στο δίκτυο

3. **Δημιουργούμε μια virtual communication port**

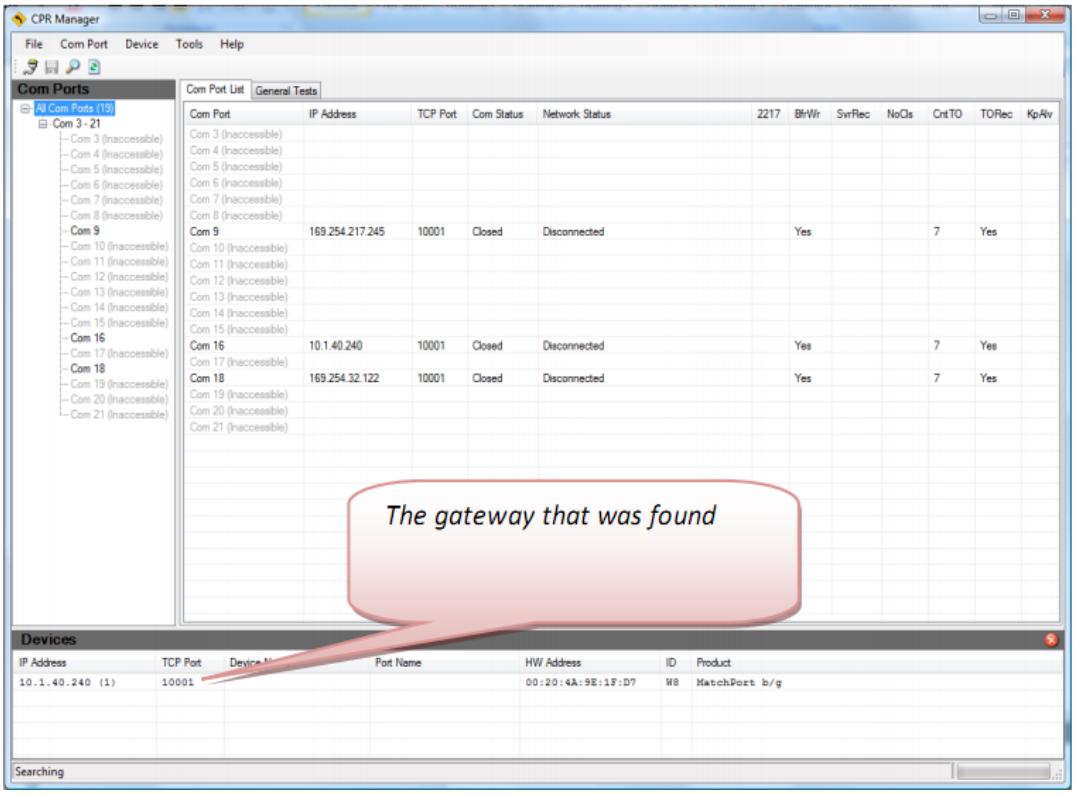
   Αφότου πετύχουμε την σύνδεση του gateway, μια εικονική θύρα επικοινωνίας στο server πρέπει να δημιουργηθεί για να μπορέσουν να ληφθούν τα δεδομένα.

   Για να γίνει κάτι τέτοιο :

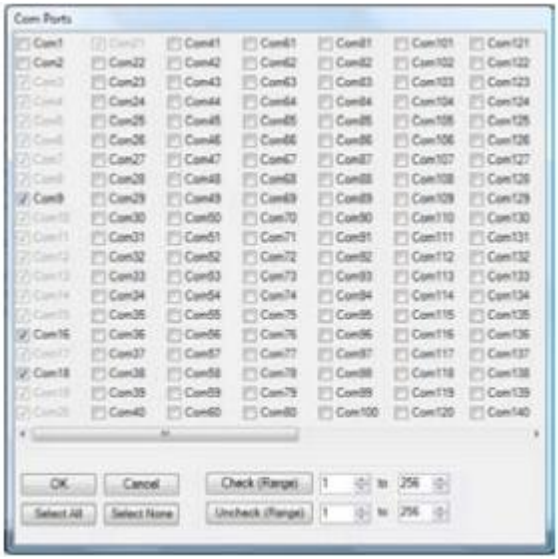
   1. Τρέχουμε το Lantronix CPR Manager



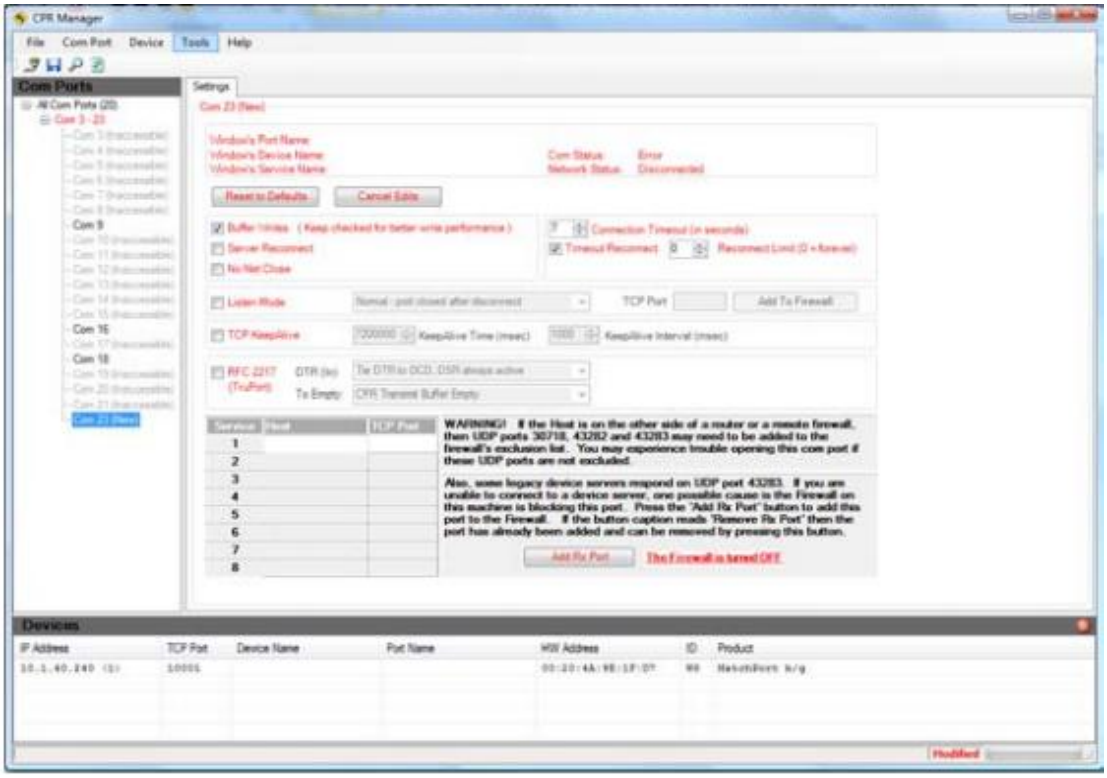
2. Επιλέγουμε Devise → Search



3. Επιλέγουμε Com Port → Add and Remove



4. Επιλέγουμε μια ελεύθερη com port



5. Επιλέγουμε το πρώτο κελί στο host table και κάντε double click στο device το οποίο βρίσκετε στο device list. Αυτόματα η ip address και η port θα γραφοτύν στο επιλεγμένο κελί.

6. Επιλέγουμε Com Port → Save settings

   Η νέα com port μπορεί τώρα να χρησιμοποιηθεί για να πραγματοποιηθεί μια σύνδεση με το gateway.

Το server-side μέρος και λήψη μερήσεων

4. Τρέχουμε το PS_ServerDataManipulation Application

Η εφαρμογή PS_ServerDataManipulation (PS_SDM) έχει δύο απαιτήσεις. Καταρχήν χρειάζεται μια virtual com port, η οποία συνδέεται με το Wireless Smart Sensor Network Gateway (WSSN GW), και δεύτερον χρειάζεται ένα valid configuration xml file όπως προαναφέραμε. Με αυτά, το PS_SDM μπορεί να συλλέξει και να χειριστεί τα δεδομένα που στέλνουν οι WSSN Collectors (αισθητήρες).

1. Σύνδεση στο WSSN GW.

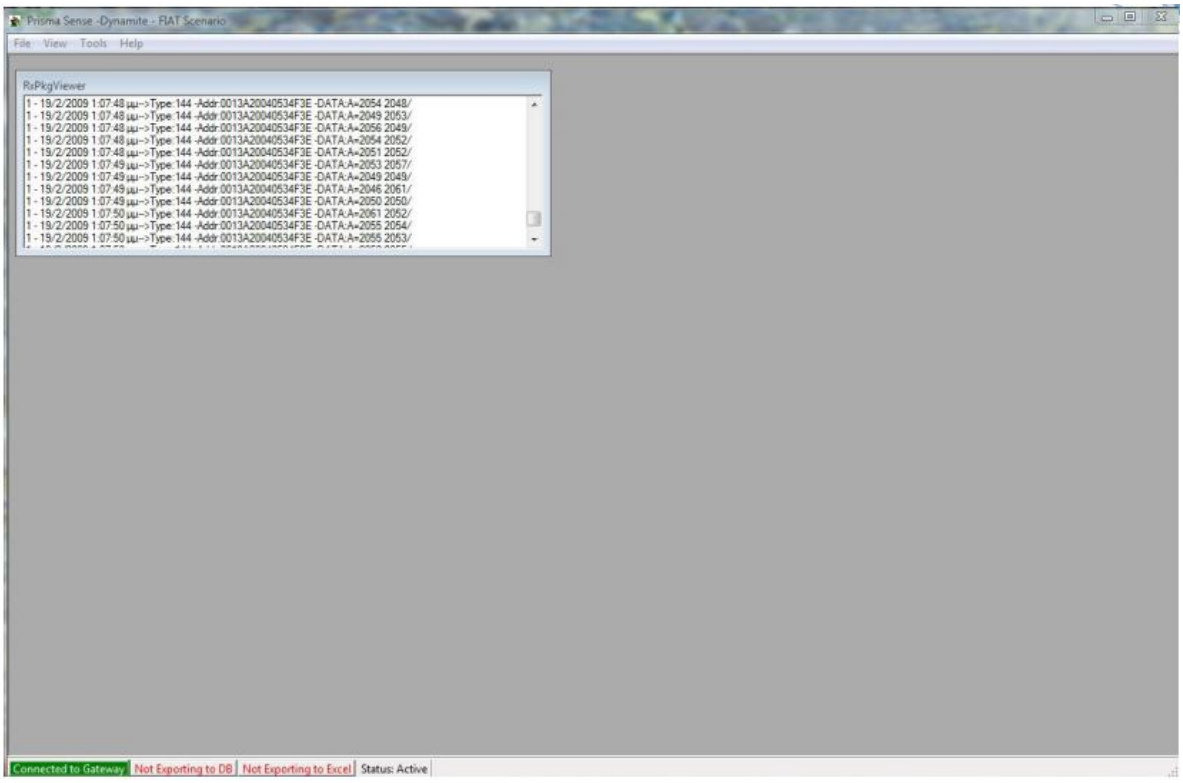
   Για να συνδεθούμε στο WSSN GW επιλέγουμε File → Connect to Gateway. Στη φόρμα που εμφανίζεται συμπληρώνουμε τον αριθμό της virtual com port στην οποία έχει συνδεθεί το GW και πατάμε connect.

   Αν η σύνδεση είναι επιτυχής το κάτω αριστερό μέρος της εφαρμογής θα γίνει πράσινο και θα εμφανίσει το μήνυμα “Connected to Gateway”. Όταν συνδεθεί στο GW το PS_SDM λαμβάνει δεδομένα από τους αισθητήρες. Αυτό φαίνεται από την εναλλαγή χρώματος στο κάτω αριστερό μέρος της εφαρμογής.

2. Βλέποντας τα ληφθέντα πακέτα.

Για να δούμε τα πακέτα που λήφθηκαν επιλέγουμε View → Rx Pkgs Viewer.

Ένα pop up παράθυρο θα εμφανιστεί το οποίο θα εμφανίζει τα ληφθέντα πακέτα και τον ώρα που αυτά λήφθηκαν από το PS_SDM



3. Network Display

   Το PS_SDM έχει μια γραφική απεικόνιση για να δείχνει την τοπολογία των αισθητήρων που σχηματίζουν το WSSN (wireless smart sensor network) που είναι συνδεδεμένο στο GW. Για να το δούμε επιλέγουμε View → Graphical Display.

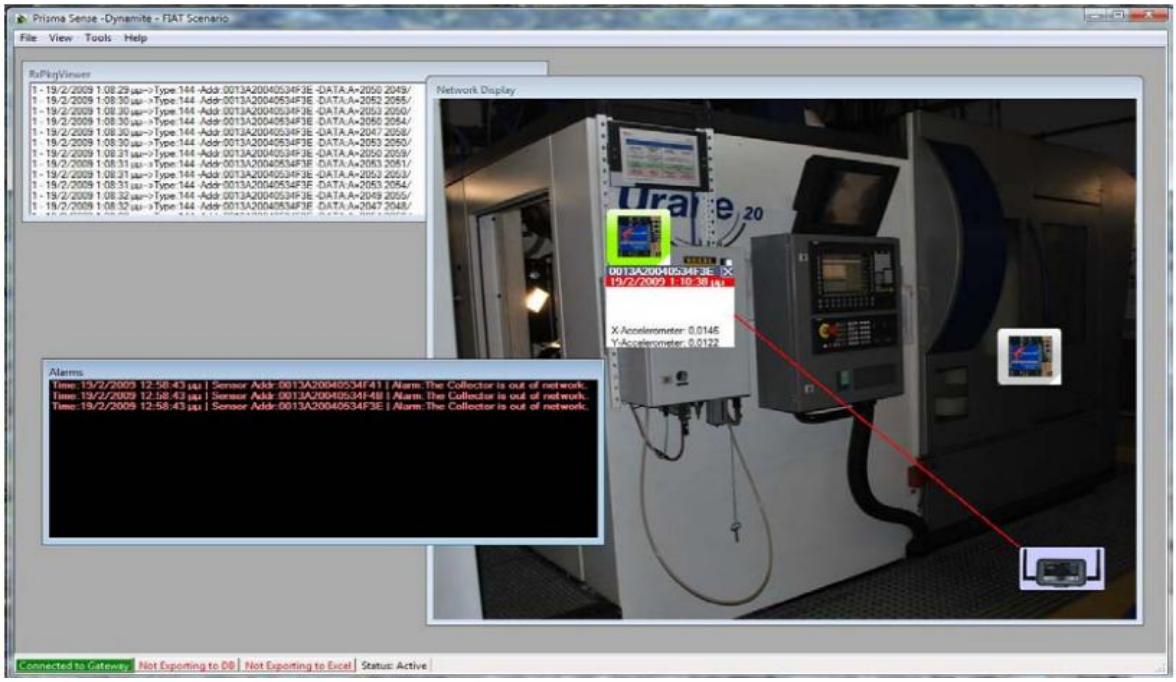
   Στο pop up παράθυρο που εμφανίζεται υπάρχει μια εικόνα στο background που αντιπροσωπεύει την περιοχή που είναι παραταγμένο το WSSN. Σε συγκεκριμένα σημεία στην εικόνα υπάρχει γραφική αναπαράσταση για κάθε αισθητήρα ο οποίος έχει δηλωθεί στο xml αρχείο. Παραταύτα αν το PS_SDM δεν λάβει κάποιο πακέτο από ένα συγκεκριμένο αισθητήρα για μια χρονική περίοδο μεγαλύτερη από αυτή του χρονικού ορίου που έχει δηλωθεί στο xml αρχείο, ο αισθητήρας αυτός θα παρουσιαστεί ως απρόσιτος και ένα αντίστοιχο alarm θα χτυπήσει.

   Επιπλέον κάνοντας right-click σε έναν αισθητήρα ένα drop down menu

   εμφανίζεται με τις τελευταίες ληφθείσες τιμές. Αυτές οι τιμές ανανεώνονται κάθε φορά που έρχεται ένα νέο πακέτο. Τέλος μπορούμε να αλλάξουμε σ' έναν αισθητήρα τη θέση με drag and drop.

4. Alarms viewer

Επιλέγοντας View → Alarms Viewer ένα pop up παράθυρο εμφανίζεται το οποίο κρατάει όλα τα alarms που έχουν προκύψει από τη στιγμή που η εφαρμογή συνδέθηκε στο GW.



Embedded Programming με το Code Composer Essentials

Το Code Composer Essentials είναι ένα IDE βασισμένο στη πλατφόρμα ανάπτυξης λογισμικού Eclipse, το οποίο συνιστά η texas instruments για να προγραμματίσει κανείς στους μικροεπεξεργαστές MSP430 που κατασκευάζει.

Υποστηρίζει

- ανάπτυξη σε γλώσσα C
- code space μέχρι και 8K bytes

Έχει επίσης Debugger

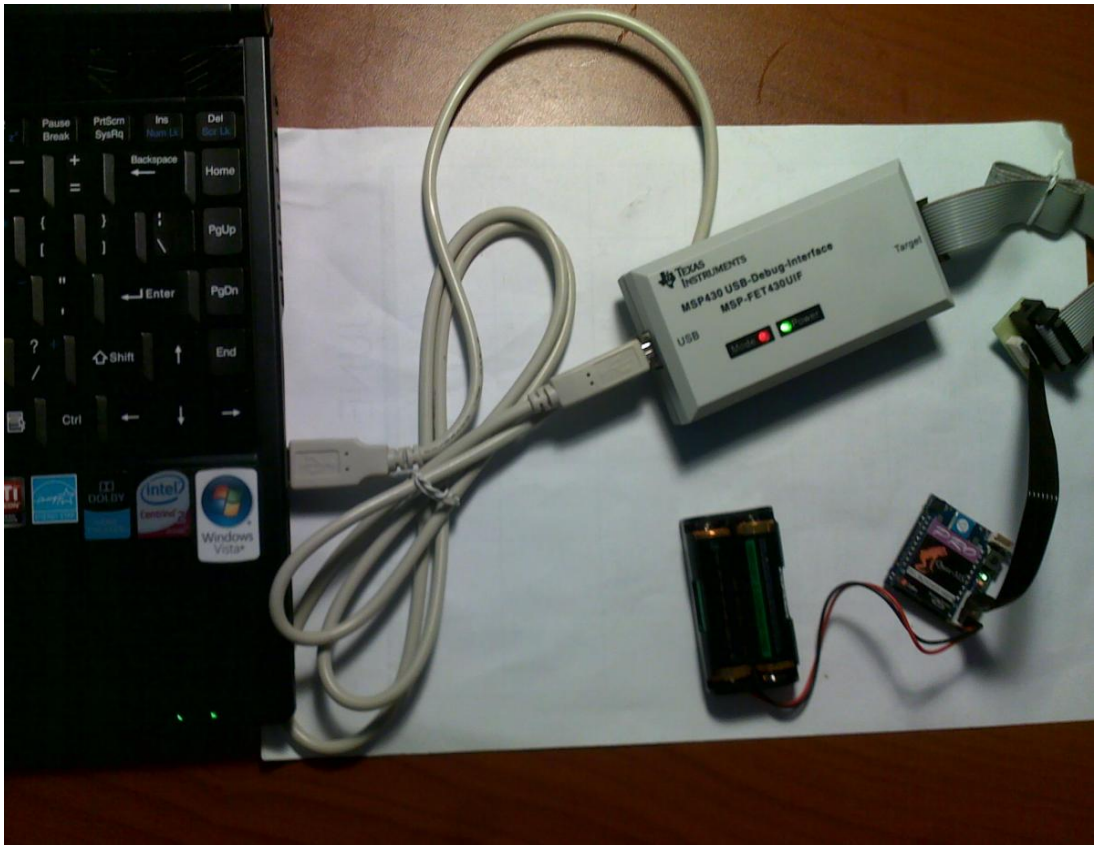
- ολοκληρωμένο visual projcect manager,
- virtual και hardware breakpoints
- ενσωματωμένο text editor με λειτουργίες
  - code completion
  - highlighting syntax errors
  - auto parameters information.

Επίσης κατα τη διάρκεια του debugging μπορεί κάποις να

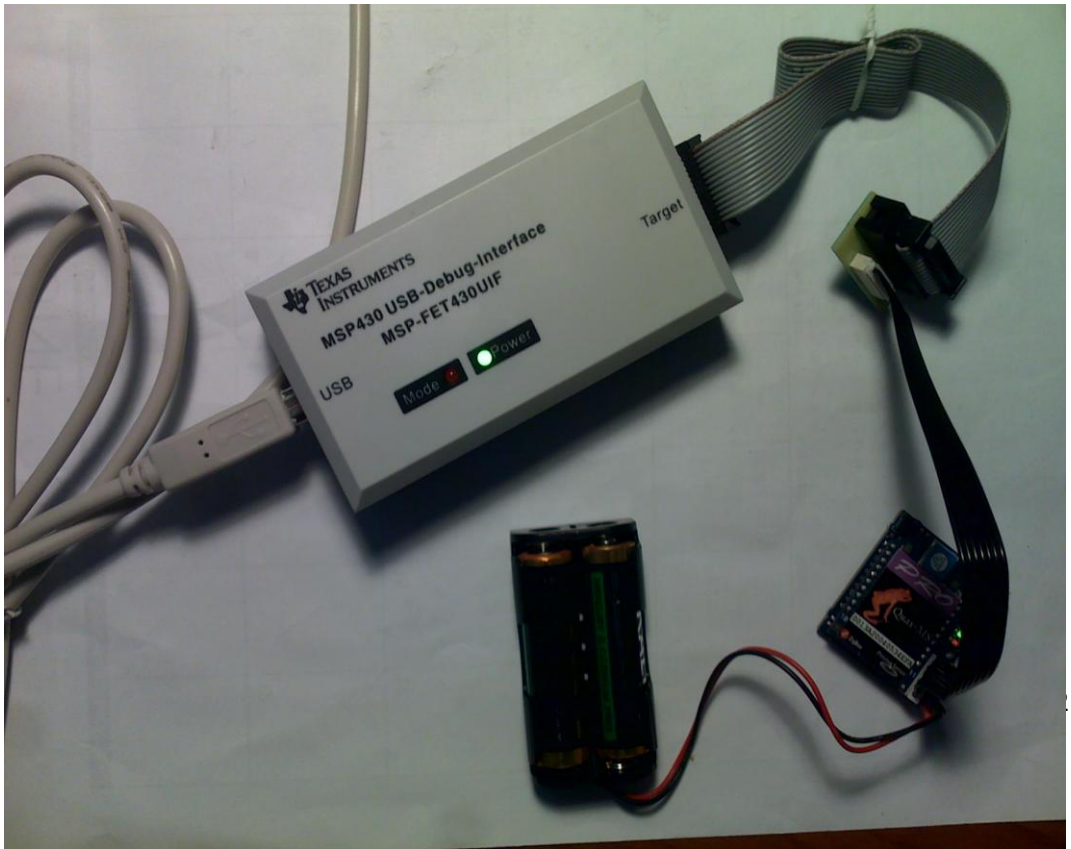
- προσπελάσει μεταβλητές και εκφράσεις,
- δει τα περιεχόμενα της μνήμης και των registers.

Για να τρέξει κάποιος ένα πρόγραμμα σε έναν επεξεργαστή χρειάζεται να έχει και τον programmer της texas instument, για να μπορεί να περάσει τα προγράμματα στον επεξεργαστή.

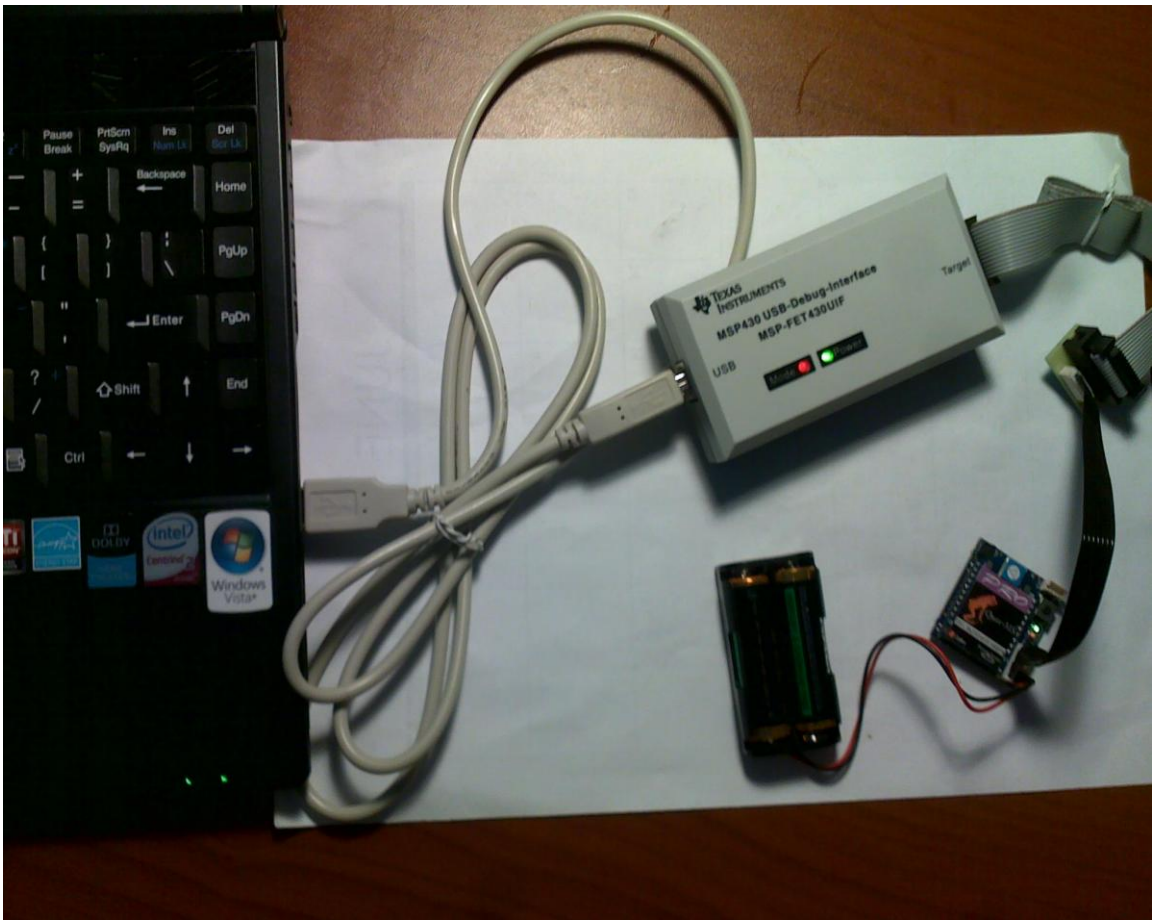
Παρακάτω φαίνεται ο programmer συνδεδεμένος με τον υπολογιστή στο ένα άκρο, και με τον sensor της Prisma ο οποίος έχει επάνω έναν επειξεργαστή MSP430:



Σε idle λειτουργία ο programmer έχει το Power led αναμμένο πράσινο.



Όταν περνάνε τα δεδομένα από τον υπολογιστή στον MSP, δηλαδή όταν κάνουμε debug, και τρέχουμε τη εφαρμογή μας, το Mode led είναι αναμμένο κόκκινο :



Είναι σημαντικό να προσέξουμε να μην αποσυνδέσουμε τον programmer όσο το mode led είναι αναμμένο καθώς μπορεί να προκληθεί βλάβη στο firmware και μετά να πρέπει να κάνουμε update εταιτο firmware.

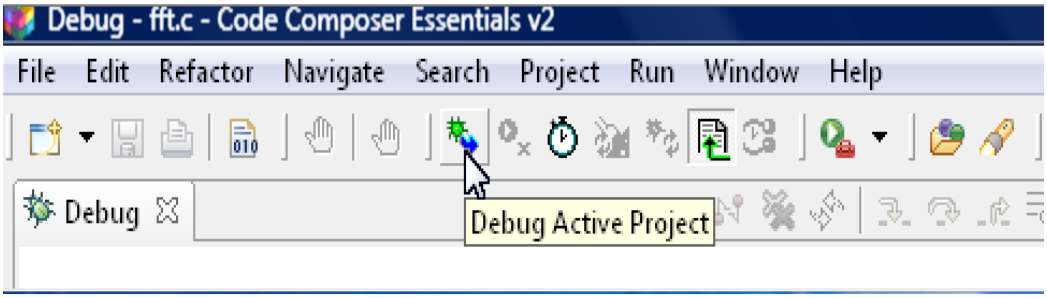
Επίσης σε αρκετές περιπτώσεις το πρόγραμμά μας μπορεί να μην γίνεται debug στον MSP και ενώ φαινομενικά όλα να λειτουργούν σωστά.Σε αυτή την περίπτωση ακολουθούμε την κλασσική τακτική του hard-reset. Αποσυνδέουμε δηλαδή τον programmer από την usb θύρα του υπολογιστή μας, διακόπτωντας έτσι την τροφοδοσία ρεύματος και έπειτα τον ξανασυνδέουμε (πάντα όμως όταν το Mode led είναι σβηστό).

Παρακάτω βρίσκεται ένα μικρό tutorial για το πως φτιάχνουμε ένα πρόγραμμα με τον CCE :

1. Ανοίγουμε το Code Composer Essentials v 2.0 (αν τρέχουμε windows 7 το τρέχουμε σε compatibility mode για vista).

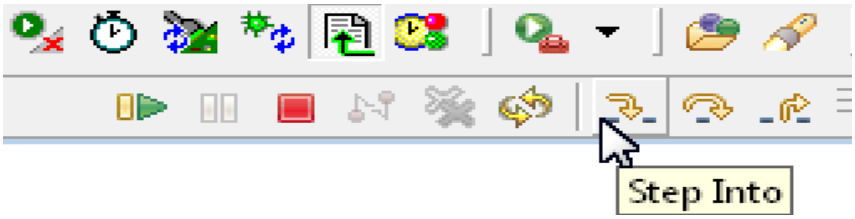
2. Επιλέγουμε να ανοίξουμε ένα υπάρχον Project ή δημιουργούμε ένα καινούργιο πατώντας File → New → Standard Make C project.

3. Αφού γράψουμε τον κώδικά μας επιλέγουμε Run → Debug Active Project. Σε αυτό το σημείο πρέπει να σιγουρευτούμε ότι ο Programmer και ο MSP είναι συνδεδεμένοι.

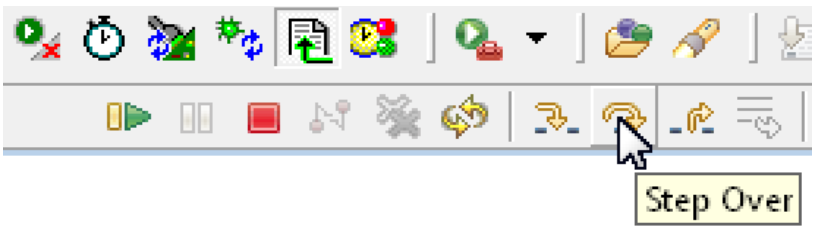


4. Έπειτα έχουμε αρκετές επιλογές για debugging. Μπορούμε να πατήσουμε halt και continue για να διακόψουμε ανά πάσα στιγμή την εκτέλεση του κώδικα.

Μπορούμε επίσης να κάνουμε βηματική εκτέλεση του κώδικα με τα κουμπιά step in και step over. Με το δεύτερο εκτελείται μια γραμμή του κώδικα σε γλώσσα C, ενώ με το πρώτο εκτελείται μια γραμμή της assembly που αντιστοιχεί στον κώδικα C οπότε για να εκτελεστεί η μια γραμμή της C θα

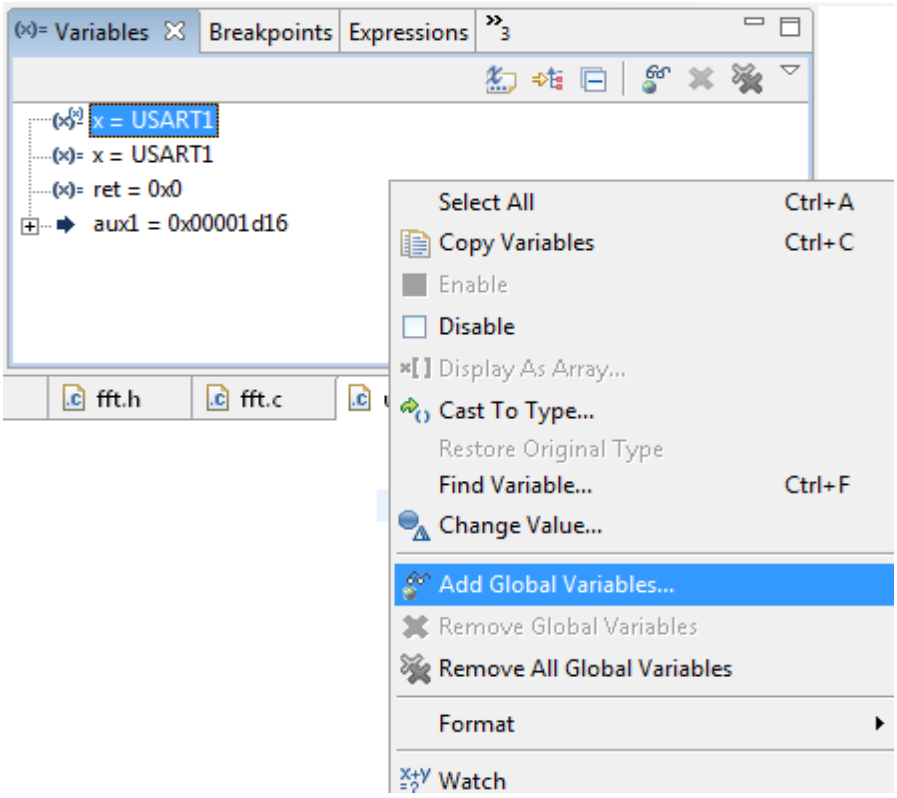


χρειαστεί να πατήσουμε πολλές φορές το κουμπί step in.



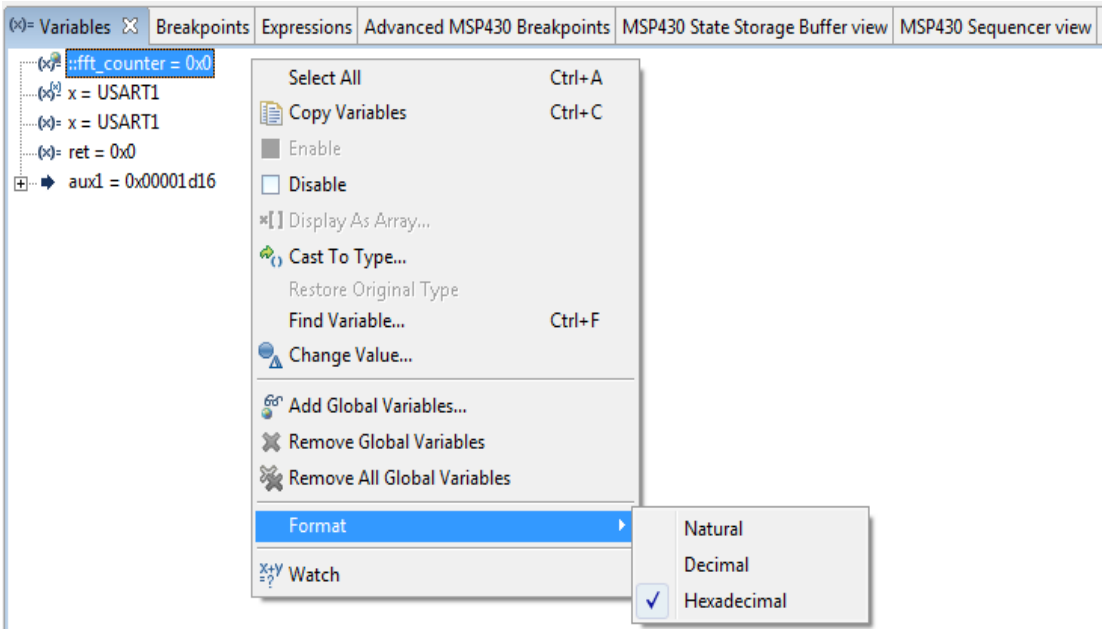
5. Επίσης μπορούμε να εισάγουμε σε πραγματικό χρόνο breakpoints ώστε να εκτελέσουμε τον κώδικά μας μέχρι ενός ορισμένου σημείου αν αυτό μας βοηθάει στο debugging. Για την εισαγωγή των breakpoints απλά κάνουμε double click στο νούμερο της γραμμής που θέλουμε να εισαχθούν.

6. Για να δούμε τις τιμές των μεταβλητών μας κάνουμε δεξί κλικ στην περιοχή variables και επιλέγουμε Add a new global variable. Ένα pop up window εμφανίζεται και επιλέγουμε όποια μεταβλητή επιθυμούμε να



παρακολουθήσουμε

7. Επίσης μπορούμε να αλλάξουμε το formatting της μεταβλητής αυτής κάνοντας δεξί κλικ επάνω της (αφού την έχουμε προσθέσει και επιλέγοντας decimal, hex, η real.



Γενικότερα η όλη φιλοσοφία του embedded debugging είναι κάπως διαφορετική. Δεδομένου ότι δεν έχουμε κάποιο τρόπο να βλέπουμε άμεσα τις τιμές των μεταβλητών, τοποθετούμε breakpoints στα σημεία που θέλουμε και εκτελούμε το πρόγραμμα μας. Όταν αυτό φτάσει στα breakpoints, προσθέτουμε στον πίνακα των μεταβλητών, τις εκφράσεις που θέλουμε να παρακολουθήσουμε. Έπειτα διαλέγουμε σε τι μορφή θέλουμε αυτές να εμφανίζονται (hexadecimal, decimal, natural) και αφού δούμε τις τρέχουσες τιμές κάνουμε βηματική εκτέλεση είτε γραμμή-γραμμή με step over, είτε μέχρι ένα άλλο σημείο που έχουμε βάλει κι άλλο break point πατώντας το run.

Κεφάλαιο 5 - Τεχνικές επεξεργασίας σήματος για Wireless sensor networks

Η επεξεργασία σημάτων είναι απαραίτητη στην διαδικασία παρακολούθησης της φθοράς καθώς χάρη σ΄ αυτήν μπορούμε να διακρίνουμε το πραγματικό σήμα από τον θόρυβο και να έχουμε μια ξεκάθαρη και σωστή εικόνα για την κατάσταση του εκάστοτε εξαρτήματος που παρακολουθούμε και κατ' επέκταση ολόκληρης της γραμμής παραγωγής. Η χρησιμοποιούμενη όμως τεχνική επεξεργασίας σήματος θα πρέπει να μπορεί να εκτελεστεί γρήγορα χωρίς να καταναλώνει πολύ υπολογιστική ισχύ, γιατί μην ξεχνάμε ότι στα ασύρματα δίκτυα αισθητήρων το θέμα της εξοικονόμησης ενέργειας είναι ένα μείζον ζήτημα αφού οι αισθητήρες λειτουργούν με μπαταρία και δεν είναι συνδεδεμένοι με κάποια μόνιμη παροχή ενέργειας.

Η ανάλυση στο πεδίο του χρόνου ενός σήματος είναι το πρώτο πράγμα που γίνεται αφού γίνει δειγματοληψία του σήματος. Στην πράξη, κάποια στατιστικά δεδομένα αποθηκεύονται με βάση το πεδίο του χρόνου, γιατί η αποθήκευση όλων των δεδομένων που παίρνεις από τα αισθητήρια όργανα απαιτεί πολύ μεγάλο χώρο αποθήκευσης. Χαρακτηριστικά, όπως προαναφέραμε, μερικές στατιστικές παράμετροι υπολογίζονται από τα δεδομένα που λαμβάνονται από τους αισθητήρες και αυτές οι παράμετροι σώζονται και χρησιμοποιούνται για τη διάγνωση της φθοράς. Η μέση τετραγωνική τιμή (RMS), ο αριθμητικός μέσος όρος, η μέση απόκλιση, η τυπική απόκλιση και η κυρτότητα είναι τυπικά παραδείγματα των στατιστικών παραμέτρων στο πεδίο του χρόνου. Οι παράμετροι αυτοί συνήθως περιέχουν όλο το συχνοτικό φάσμα των σημάτων που μετριούνται, οπότε είναι και ευαίσθητοι αρκετά στο θόρυβο.

Η ανάλυση στο πεδίο των συχνοτήτων μέσω του ταχύ μετασχηματισμού Fourier (FFT) είναι ένα μέσο καθορισμού του συχνοτικού περιεχομένου του μετρούμενου σήματος. Η βασική ιδέα είναι ότι η φθορά ενός μηχανήματος επηρεάζει το συχνοτικό περιεχόμενο του σήματος. Η μέθοδος FFT χρησιμοποιείται για την εξάλειψη των άχρηστων πληροφοριών και επικεντρώνεται στη διαχείριση της

χρήσιμης πληροφορίας.

Η αρχική εφαρμογή της PrismaSense έστελνε απευθείας τις μετρούμενες τιμές από τους αισθητήρες στον κεντρικό server. Έπειτα ο server ήταν υπεύθυνος για την όποια επεξεργασία των τιμών αυτών αλλά και την εξαγωγή συμπερασμάτων. Στόχος μας ήταν να επεκτείνουμε αυτή την εφαρμογή υλοποιώντας έναν RMS και έναν FFT αλγόριθμο και προσδίδοντας κάνοντας το ασύρματο δίκτυο αισθητήρων πιο "έξυπνο", στη μεριά των αισθητήρων και όχι σε αυτή του server.

Κάτι τέτοιο είναι θεμιτό γιατί :

- Εκμεταλλευόμαστε την υπολογιστική ισχύ του αισθητήρα, προσδίδοντας του έτσι ευφυΐα.
- Κάνουμε το σύστημα πιο άμεσο, αφού πλέον οι αποφάσεις λαμβάνονται στον αισθητήρα και όχι στον server και αφότου ληφθεί ένας μεγάλος όγκος δεδομένων
- Μειώνουμε την κίνηση των πακέτων που στέλνονται από τον αισθητήρα στο server, μειώνοντας έτσι και το κόστος των χαμένων πακέτων
- Βελτιώνουμε σημαντικά την κατανάλωση πόρων στους αισθητήρες, αφού ο κάθε αισθητήρας πλέον στέλνει λιγότερα δεδομένα. Έτσι το σύστημα μπορεί να λειτουργήσει για μεγαλύτερο χρονικό διάστημα πριν χρειαστεί αλλαγή μπαταριών

RMS

Ο RMS -root mean squared- ή και quadratic μέσος είναι ένα στατιστικό εργαλείο που δείχνει το μέγεθος μιας διαφορετικής ποσότητας και είναι ιδιαίτερα χρήσιμο σε περιπτώσεις που έχουμε θετικές και αρνητικές τιμές. Αυτό μας αρέσει, λόγω του ότι οι τιμές του επιταχυνσιόμετρο έχουν αυτή τη μορφή. Το επιταχυνσιόμετρο που υπάρχει πάνω στην πλακέτα QUAX μετράει **±5 g**.

Στην περίπτωση ενός set n τιμών

$$\{x_1, x_2, \ldots, x_n\}$$

ο τύπος για το RMS είναι :

$$x_{\mathrm{rms}} = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}{n}}.$$

Θεωρητικά όταν δεν υπάρχουν ισχυροί κραδασμοί ο RMS θα μας δίνει μικρότερες τιμές κοντά ενώ σε ξαφνικές μεταβολές των δονήσεων η τιμή αυτή αναμένεται να είναι μεγάλη.

Η υλοποίηση του RMS είναι απλή :

```
uint16_t rms(uint16_t *x, int N)
{
        int i;
        uint16_t sum =0;
  for(i=0; i<N; i++)  {
        sum += (x[i]*x[i]);
  }
  return sqrt(sum/N);
}
```

Το application για τον RMS είναι το ακόλουθο :

```
extern uint16_t A0result,A1result;
extern adc12_config_t adc_conf;
extern adc12_ctrl_mem_t adc_mem0, adc_mem1;
uint8_t app_acceleration_state;
uint16_t app_acceleration_periode;

uint8_t i;
uint16_t x1[RMS_SIZE];
uint16_t x2[RMS_SIZE];

uint16_t val1;
uint16_t val2;
```

```
void app_acceleration_init()
{
        i=0;
        app_acceleration_state=0;
        app_acceleration_periode=10;
        ProcSchedInsert (app_acceleration_measurement,20,100);
}

void app_acceleration_measurement()
{
        char y[30];

        if (app_acceleration_state==0) //Step1 Enable and setup ad converter
        {
                accel_setup (&adc_conf,&adc_mem0,&adc_mem1);
                app_acceleration_state=1;
                peripherals_sleep.sc_accel=0;
                ProcSchedInsert (app_acceleration_measurement,1,100);
        }
        else//Step2 Read measurement and send it
        {
                accel_meas (SEQ);
                ADC12_disable();
                app_acceleration_state=0;
                peripherals_sleep.sc_accel=1;
                ProcSchedInsert (app_acceleration_measurement,app_acceleration_periode,100);

                i++;
                x1[i] = A0result;
                x2[i] = A1result;
```

```
                if(i==RMS_SIZE)
                {
                        peripherals_sleep.sc_xbee=0;
                        sleep_disable();

                        val1 = rms(x1, RMS_SIZE);
                        val2 = rms(x2, RMS_SIZE);

                        sprintf(y,"A=%4d %4d/", val1, val2);
                        app_txx.data = y;
                        app_txx.data_length = 16;
                        xb802_send(&app_txx);
                        i=0;
                }
        }
}
```

Τα A0result και A1result ορίζονται αλλού και χρησιμοποιούνται για την αποθήκευση της εκάστοτε μέτρησης του αισθητήρα επιτάχυνσης. Το πρόγραμμα δουλεύει ως εξής:

Αρχικά καλείται η συνάρτηση *app_acceleration_init* η οποία καλεί την *ProcSchedInsert* η οποία εισάγει μια νέα procedure στη λίστα του scheduler. Αυτή η νέα procedure δεν είναι άλλη από την app_acceleration_measurement. Τα άλλα δύο ορίσματα με την περίοδο και την προτεραιότητα που έχει μια procedure.

Αφού λοιπόν μπει η app_acceleration_measurement είναι ζήτημα χρόνου το πότε θα καλεστεί. Με το που κληθεί λοιπόν αρχικά καθορίζει τον τρόπο που θα χρησιμοποιηθεί το επιταχυνσιόμετρο με την *accel_meas (SEQ)* και έπειτα απενεργοποιεί τον analog-to-digital converter.

Με την *peripherals_sleep.sc_accel=1;* βάζουμε σε sleep mode το επιταχυνσιόμετρο και επανεισάγουμε στον scheduler την ίδια διεργασία δεδομένου ότι θέλουμε να λειτουργεί συνέχεια και όχι για να πάρουμε μόνο μια μέτρηση.

Στους δύο πίνακες x1 και x2 βάζουμε αρχικά την εκάστοτε μέτρηση της τιμής της επιτάχυνσης του αισθητήρα. Έχουμε δύο πίνακες γιατί έχουμε μετρήσεις στον οριζόντιο αλλά και τον κάθετο άξονα. Επίσης έχουμε μια μεταβλητή i η οποία λειτουργεί ως counter, επιτρέποντας μας να μπούμε στο if-block μόνο όταν έχουμε πάρει #RMS_SIZE μετρήσεις.

Όταν γίνει κάτι τέτοιο βγάζουμε από το sleep mode το xbee module καλούμε τον RMS για την επεξεργασία των ληφθέντων μετρήσεων και τοποθετούμε τα δεδομένα στην μεταβλητή y την οποία και στέλνουμε καλώντας την συνάρτηση *xb802_send(&app_txx);*

Αυτή η διαδικασία συνεχίζεται επ' άπειρο παρακολουθώντας έτσι το σύστημα μας.

FFT

Ο Fast Fourier Transform (FFT) αποτελεί μία αποτελεσματική υλοποίηση του Discrete Fourier Transform (DFT) και ένα πολύ σημαντικό εργαλείο για εφαρμογές ψηφιακής επεξεργασίας σήματος. Λόγω της δομής του, αποτελεί και ένα αποτελεσματικό benchmark όσον αφορά την απόδοση μιας εφαρμογής ψηφιακής επεξεργασίας σήματος.

Υπάρχουν διάφορες αλγόριθμοι για την υλοποίηση του FFT. Εμείς θα υλοποιούμε τον γνωστό Cooley Turkey FFT αλγόριθμο ο οποίος εκμεταλλεύεται τη συμμετρία των ριζών του FFT και χρησιμοποιεί την τεχνική διαίρει και βασίλευε. Ανήκει στους radix-2 FFT αλγορίθμους.

Η εξίσωση του Discrete Fourier Transform (DFT) είναι η ακόλουθη :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \text{ , } k = 0 \text{ to } N-1 \text{ where } W_n = e^{-j2\pi/N}$$

Όπως μπορεί κανείς να παρατηρήσει χρειάζονται αρκετοί υπολογισμοί. Πιο συγκεκριμένα χρειάζονται N^2 μιγαδικοί πολλαπλασιαμοί και άλλες τόσες μιγαδικές προσθέσεις για έναν DFT N-σημείων. Ένας αλγόριθμος που μπορεί να βελτιώσει δραματικά τον αριθμό των υπολογισμών είναι ο radix-2 FFT, ο οποίος εκμεταλλεύται την περιοδικότητα του twiddle factor :

$$W_N^{nk}$$

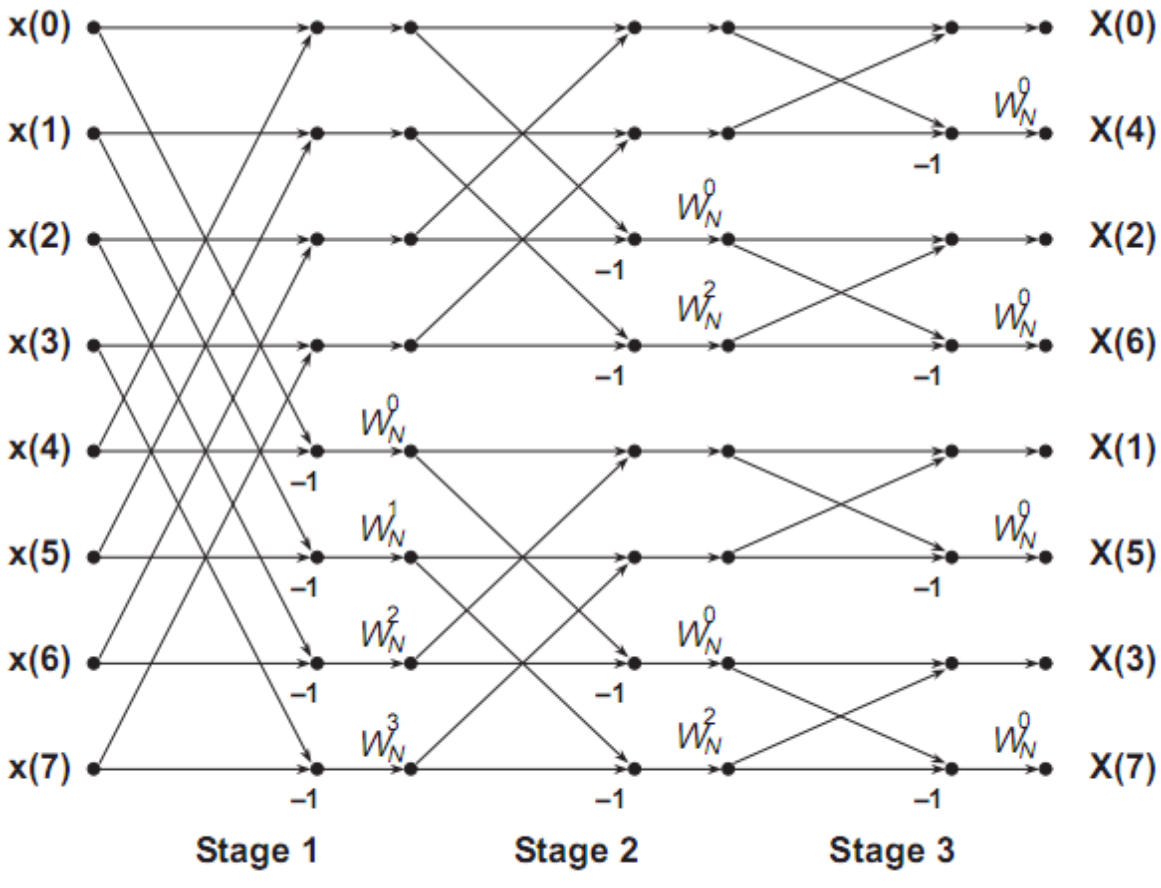
Για παράδειγμα όταν το μέγεθος του πίνακα τιμών εισόδου n = N τότε :

$$W_N^{nk} = W_N^{Nk} = e^{-j\left(\frac{2\pi}{N}\right)NK} = e^{-j2\pi k} = -1.$$

Η radix-2 FFT εξίσωση είναι η ακόλουθη :

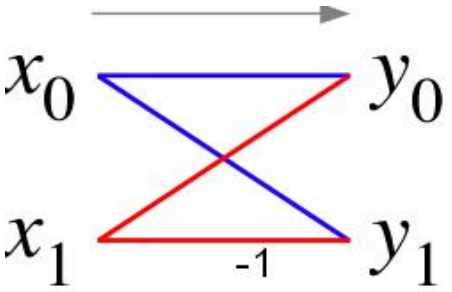
$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{nk}$$

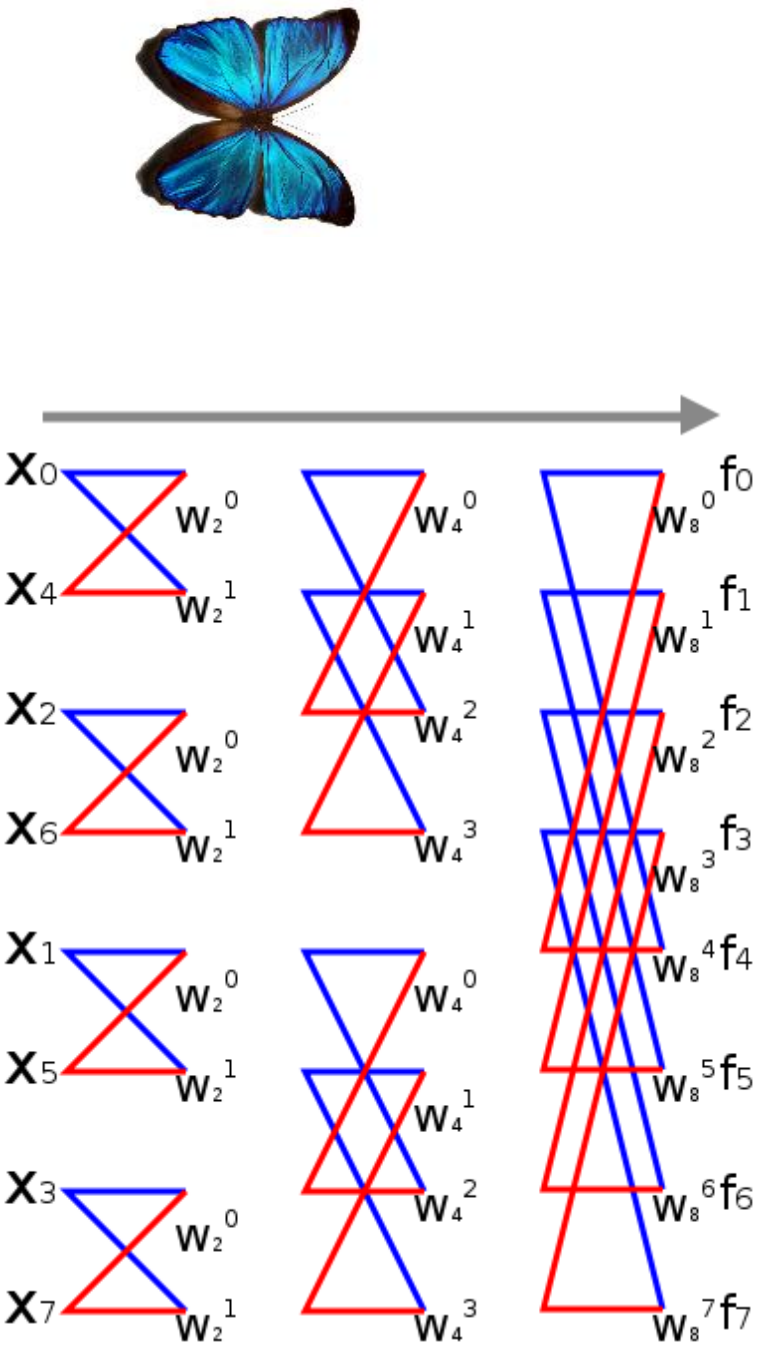
Αυτή η εξίσωση απλώς διαιρεί τον DFT σε 2 μικρότερους DFT οι οποίοι με τη σειρά τους διαιρούνται και αυτοί σε 2 μικρότερους κ.ο.κ. Αποτελείται από logN στάδια και κάθε στάδιο από N/2 πεταλούδες. Κάθε πεταλούδα αποτελείταιαπό 2 προσθέσεις για τα δεδομένα εισόδου και έναν πολλαπλασιασμό με το twiddle factor. Σχηματικά:



Radix-2 FFT for N=8

Ο όρος πεταλούδα χρησιμοποιείται για να εκφράσει τον υπολογισμό που συνδυάζει τα αποτελέσματα ενός μικρότερου DFT σε έναν μεγαλύτερο DFT ή ανάποδα (σπάσιμο ενός μεγάλου DFT σε δύο υπομετασχηματισμούς). Το όνομα πεταλούδα προέρχεται από το σχήμα του διαγράμματος ροής δεδομένων στην radix-2 περίπτωση.





Για να υλοποιήσουμε τον Cooley Turkey FFT αλγόριθμο για τον MSP αρχικά ορίσαμε μια δομή complex η οποία αναπαριστά έναν μιγαδικό αριθμό με το πραγματικό και το φανταστικό του μέρος :

```
typedef struct complex_t {
   double re;
   double im;
} complex;
```

Έπειτα ορίσαμε τέσσερεις συναρτήσεις με τις πέντε βασικές πράξεις μιγαδικών αριθμών που χρειάζεται να κάνει κανείς. Την μιγαδική πρόσθεση, αφαίρεση, πολλαπλασιασμό, το μέτρο και την τριγωνομετρική μορφή :

```
complex complex_from_polar(double r, double theta_radians) {
   complex result;
   result.re = r * cos(theta_radians);
   result.im = r * sin(theta_radians);
   return result;
}
```

```
double complex_magnitude(complex c) {
   return sqrt(c.re*c.re + c.im*c.im);
}
```

```
complex complex_add(complex left, complex right) {
   complex result;
   result.re = left.re + right.re;
   result.im = left.im + right.im;
   return result;
}

complex complex_sub(complex left, complex right) {
   complex result;
   result.re = left.re - right.re;
   result.im = left.im - right.im;
   return result;
}

complex complex_mult(complex left, complex right) {
   complex result;
   result.re = left.re*right.re - left.im*right.im;
   result.im = left.re*right.im + left.im*right.re;
   return result;
}
```

Έπειτα φτιάξαμε μια πρώτη αρχική υλοποίηση του FFT η οποία όμως δεν χρησιμοποιούσε την τεχνική του διαίρει και βασίλευε και έκανε πολλές πράξεις :

```
#define PI  3.1415926535897932

void DFT_simple_1(complex* x, int N, complex* X) {
   int k, n;
   for(k=0; k<N; k++) {
      X[k].re = X[k].im = 0.0;
      for(n=0; n<N; n++) {
         X[k] = complex_add(X[k], complex_mult(x[n], complex_from_polar(1, -2*PI*n*k/N)));
      }
   }
}

complex* DFT_simple_2(complex* x, int N) {
   int k, n;
   complex* X = (complex*) malloc(sizeof(struct complex_t)* N);
   for(k=0; k<N; k++) {
      X[k].re = X[k].im = 0.0;
      for(n=0; n<N; n++) {
         X[k] = complex_add(X[k], complex_mult(x[n], complex_from_polar(1, -2*PI*n*k/N)));
      }
   }
   return X;
}
```

Στην πρώτη περίπτωση δηλώνουμε τον πίνακα στον οποίο θέλουμε να αποθηκευτεί το μετασχηματισμένο Fourier σήμα ενώ στη δεύτερη μας το επιστρέφει η ίδια η συνάρτηση. Αυτό έγινε γιατί δεν ήμασταν σίγουροι αν η malloc θα δούλευε σωστά στον MSP αλλά τελικά αποδείχθηκε ότι δουλεύει σωστά.

Τέλος υλοποιήσαμε μια τον Cooley Turkey FFT :

Όπως προαναφέρθηκε ο radix-2 decimation in time αλγόριθμος για τον FFT χρησιμοποιεί μια divide-and-conquer (διαίρει και βασίλευε) τεχνική για να βελτιώσει την απόδοση. Ξεκινάει διαχωρίζοντας το σήμα εισόδου x (που είναι αποθηκευμένο σε ένα πίνακα) σε 2 μικρότερους πίνακες d και e. Το d περιέχει τα περιττά στοιχεία του αρχικού πίνακα και το e τα άρτια στοιχεία.

Για παράδειγμα :

x = (1,2,3,4,5,6)

d = (2,4,5)

e = (1,3,5)

Έπειτα υπολογίζουμε αναδρομικά τους FFT των d και e και από αυτούς μετά κατασκευάζουμε το X που περιέχει τις τιμές Fourier χρησιμοποιώντας τις εξισώσεις

$$\begin{aligned} X_k &= E_k + e^{-\frac{2\pi i}{N}k} D_k \quad 0 \leq k < N/2 \\ X_{k+N/2} &= E_k - e^{-\frac{2\pi i}{N}k} D_k \quad 0 \leq k < N/2 \end{aligned}$$

```
complex* CT_FFT(complex* x, int N /* must be a power of 2 */) {
   complex* X = (complex*) malloc(sizeof(struct complex_t) * N);
   complex * d, * e, * D, * E;
   int k;
   if (N == 1) {
      X[0] = x[0];
      return X;
   }
   e = (complex*) malloc(sizeof(struct complex_t) * N/2);
   d = (complex*) malloc(sizeof(struct complex_t) * N/2);
   for(k = 0; k < N/2; k++) {
      e[k] = x[2*k];
      d[k] = x[2*k + 1];
   }
   E = CT_FFT(e, N/2);
   D = CT_FFT(d, N/2);
   free(e);
   free(d);
   for(k = 0; k < N/2; k++) {
      /* Multiply entries of D by the twiddle factors e^(-2*pi*i/N * k) */
      D[k] = complex_mult(complex_from_polar(1, -2.0*PI*k/N), D[k]);
   }
   for(k = 0; k < N/2; k++) {
      X[k]      = complex_add(E[k], D[k]);
      X[k + N/2] = complex_sub(E[k], D[k]);
   }
   free(D);
   free(E);
   return X;
}
```

Και ενώ σε ένα desktop pc κάποιος δεν μπορεί να δει σημαντική διαφορά στους δύο αυτούς αλγορίθμους, στον μικροεπεξεργαστή MSP η διαφορά είναι πολύ μεγάλη και σημαντική.

Σε σχέση με το application του RMS η περίπτωση του FFT είναι κάπως πιο περίπλοκη καθώς δεν αρκεί να στείλουμε μία τιμή κάθε φορά αλλά έναν ολόκληρο πίνακα δεδομένων. Για να γίνει κάτι τέτοιο υλοποιούμε την συνάρτηση *void tx_acc_pkg(enum InterruptID x)* η οποία καλείται όταν χτυπήσει το interrupt XBEE_TXOK, δηλαδή όταν έρθει ένα acknowledge ότι ένα πακέτο παραδόθηκε σωστά στον server.

Σε αντίθετη περίπτωση, αν δεν υλοποιούσαμε αυτή τη συνάρτηση, δεν θα μπορούσαμε να εγγυηθούμε για την σωστή παράδοση των πακέτων καθώς τα πακέτα θα στέλνονταν με τόσο μεγάλο ρυθμό που ο server θα αδυνατούσε να τα πάρει όλα.

```
extern uint16_t A0result,A1result;
extern adc12_config_t adc_conf;
extern adc12_ctrl_mem_t adc_mem0, adc_mem1;
uint8_t app_acceleration_state;
uint16_t app_acceleration_periode;
```

```
#define FFT_SIZE 4

complex *p1;

complex *p2;

uint8_t fft_counter = 0;

uint8_t i = 0;

complex x1[FFT_SIZE], x2[FFT_SIZE];


void tx_acc_pkg(enum interruptID x)

{

        char y[50];

                if((fft_counter < FFT_SIZE))

        {

                sleep.sc_xbee = 0;

                sleep_disable();

                peripherals_sleep.sc_xbee=0;


                //send next package

                sprintf(y,"counter=%d Re_p=%4u Im_p=%4u/ Re_p=%4u Im_p=%4u        /",
                fft_counter,(p1 + fft_counter)->re, (p1 + fft_counter)->im,(p2                 +
fft_counter)->re, (p2 + fft_counter)->im);


                fft_counter++;

                app_txx.data = y;

                app_txx.data_length = 50;

                xb802_send(&app_txx);


        }

}
```

```
void app_acceleration_init()
{
        app_acceleration_state=0;
        app_acceleration_periode=10;
        ProcSchedInsert (app_acceleration_measurement,20,100);
        IntHandlerInsert (XBEE_TXOK, tx_acc_pkg);
        fft_counter = FFT_SIZE+5;//5 arbitrary
}



void app_acceleration_measurement()
{
        char y[30];

        if (app_acceleration_state==0) //Step1 Enable and setup ad converter
        {
                accel_setup (&adc_conf,&adc_mem0,&adc_mem1);
                app_acceleration_state=1;
                peripherals_sleep.sc_accel=0;
                ProcSchedInsert (app_acceleration_measurement,1,100);
        }
        else//Step2 Read measurement and send it
        {

                accel_meas (SEQ);
                ADC12_disable();

                app_acceleration_state=0;
                peripherals_sleep.sc_accel=1;
      ProcSchedInsert(app_acceleration_measurement,app_acceleration_period
        e,100);
```

```
                //store FFT_SIZE measurements to FFT them
                x1[i].re = A0result;
                x1[i].im = 0.0;
                x2[i].re = A1result;
                x2[i].im = 0.0;

                i++;

                if(i==FFT_SIZE-1)
                {
                        fft_counter = 0;

                        p1 = CT_FFT(x1,FFT_SIZE);
                        p2 = CT_FFT(x2,FFT_SIZE);
                        i=0;

                        peripherals_sleep.sc_xbee=0;
                        sleep_disable();

                        sprintf(y,"Initiating FFT array/");
                        app_txx.data = y;
                        app_txx.data_length = 16;
                        xb802_send(&app_txx);

                }

        }
}
```

Κεφάλαιο 6 - Συμπεράσματα και Μελλοντικές κατευθύνσεις

Συμπεράσματα

Η πίεση του χρόνου δυστυχώς δε μας άφησε να ολοκληρώσουμε πλήρως το έργο μας γιατί σκοπός είναι το σύστημα αυτό από monitoring να γίνει preventing δηλαδή μόνο αφότου ο Fourier Transform δει ύποπτα δεδομένα, να στείλει ένα alarm value.

Επίσης θα μπορούσαν να γίνουν optimizations στον FFT αλγόριθμο, αφαιρώντας την αναδρομή και υπολογίζοντας offline τα twiddle factors. Όμως πρέπει να είναι γνωστό εξ' αρχής το μέγεθος του FFT και αυτό δεν μπορούμε παρά να το επιλέξουμε τυχαία τώρα , μιας και δεν έχουν γίνει ακόμα οι απαραίτητες μετρήσεις για να δούμε τι μεγέθη Fourier βγάζουν πιο ακριβές αποτελέσματα.

Πάραυτα έγινε μια πρώτης τάξης προσέγγιση των preventive intelligent wireless sensor συστημάτων και χαράχτηκε η πορεία για το μέλλον. Η επαλήθευση της ορθότητας του συστήματος καθώς και η περαιτέρω ανάπτυξή του βρίσκεται στα σκαριά.

Επίσης έγινε μια παρουσίαση της μοναδικής Ελληνικής πλατφόρμας για ασύρματα δίκτυα αισθητήρων η οποία ελπίζουμε να αποτελέσει ισχυρό κίνητρο για μελλοντική έρευνα και ανάπτυξη πάνω σε αυτόν τον τομέα.

Μελλοντικές Κατευθύνσεις

Είναι κοινός τόπος ότι τα συστήματα ασύρματων αισθητήρων έχουν αναπτυχθεί ιδιαίτερα τα τελευταία χρόνια και μελλοντικά θα κάνουν ολοένα και πιο εμφανή την παρουσία τους σε πολλούς κλάδους και χώρους. Όσο η τεχνολογία εξελίσσεται οι κόμβοι αισθητήρων θα

αποκτούν περισσότερες δυνατότητες αλλά και αυτονομία. Έτσι τέτοια συστήματα αναμένεται να κάνουν τη διαφορά σε διάφορα τεχνολογικά προϊόντα.

Στο χώρο της βιομηχανικής παραγωγής τα δίκτυα ασύρματων αισθητήρων ρέπουν πλέον προς το preventive maintenance, από το απλό monitoring, δηλαδή να είναι σε θέση να προλαμβάνουν ζημιές προβλέποντας ‘τες.

Κάποιος μπορεί εδώ να εξελίξει δύο πράγματα. Αφενός την τεχνολογία ανάλυσης και πρόβλεψης/εκτίμησης του κόστους ζωής των υλικών χρησιμοποιώντας διάφορα μαθηματικά μοντέλα και αφετέρου την τεχνολογία του ίδιου του ασύρματου δικτύου το οποίο θα πρέπει να μπορεί από μόνο του χωρίς να υπάρχει κάποιος κεντρικός server, να αποφασίζει για το αν υπάρχει μια βλάβη (η διαφαίνεται μελλοντικά) λειτουργώντας με έξυπνο και κατανεμημένο τρόπο. Και οι δύο κατευθύνσεις είναι εξίσου σημαντικές και αναμένεται να έχουν άμεση εφαρμογή σε πολλούς τομείς.