ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΟΥ ΤΜΗΜΑΤΟΣ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΣΥΡΜΑΤΟΣ ΚΟΜΒΟΣ ΑΠΟΜΑΚΡΥΣΜΕΝΟΥ ΕΛΕΓΧΟΥ ΚΛΙΜΑΤΙΣΤΙΚΩΝ ΕΓΚΑΤΑΣΤΑΣΕΩΝ

ΦΟΙΤΗΤΗΣ : Αλέξανδρος Μανιφάβας
ΑΜ: 14040

Επιβλέπων επίκουρος καθηγητής: Ph.D Ε.Μ.Π.(Ελλάδα) Δουμένης Γρηγόρης

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή κ.Δουμένη Γρηγόρη για την πολύτιμη βοήθεια του κατά την διάρκεια διεκπεραίωσης της πτυχιακής μου εργασίας, καθώς και τους συμφοιτητές μου Γιάννη Παπαννικολάου , Γιάννη Μασκλαβάνο και Βασίλη Λισγάρα και για την εξαίρετη συνεργασία μας για την υλοποίηση του project. Επίσης την οικογένεια μου για την στήριξη καθ' όλη την διάρκεια .

Δήλωση πνευματικής ιδιοκτησίας

Η παρούσα εργασία αποτελεί προϊόν αποκλειστικά δικής μου προσπάθειας. Όλες οι πηγές που χρησιμοποιήθηκαν περιλαμβάνονται στη βιβλιογραφία και γίνεται ρητή αναφορά σε αυτές μέσα στο κείμενο όπου έχουν χρησιμοποιηθεί.

ΥΠΟΓΡΑΦΗ

_____

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία, υλοποιήθηκε ένας τρόπος εξοικονόμησης ενέργειας σε HVAC συστήματα. Αρχικά παρουσιάζονται θέματα σχετικά με το Διαδίκτυο των πραγμάτων, τα ασύρματα δίκτυα αισθητήρων και τα ενσωματωμένα συστήματα. Για την υλοποίηση του project χρησιμοποιήθηκε ένας μικροελεγκτής , ο MSP430F5529 , της εταιρίας Texas Instruments ο οποίος σε συνεργασία με αισθητήρες γίνεται παρακολούθηση των κλιματιστικών μονάδων του ΤΕΙ ΗΠΕΙΡΟΥ και ο έλεγχος τους μέσω από την ιστοσελίδα "http://m2m.ce.teiep.gr/hvac/". Ο τρόπος ανταλλαγής μηνυμάτων βασίζεται στο πρωτόκολλο επικοινωνίας MQTT σε μορφή JSON.

Λέξεις κλειδιά: MSP430, MQTT, Γλώσσα C/C++, HVAC system

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΟΝΩΝ

ΠΡΟΛΟΓΟΣ

Στην σήμερον ημέρα η τεχνολογία εξελίσσεται συνεχώς. Με την χρήση του Ιντερνέτ ο κάθε άνθρωπος ανεξαρτήτου ηλικίας μπορεί να αποκτά γνώσεις σε οποιοδήποτε κλάδο. Συγκεκριμένα τα ηλεκτρονικά και ο προγραμματισμός είναι κλάδοι οι οποίοι αν συνδυαστούν μπορούν να επιλύσουν πολλά προβλήματα και να κάνουν την ζωή πιο εύκολη.

ΣΚΟΠΟΣ

Η υλοποίηση ενός project με το οποίο θα γίνει εξοικονόμηση ενέργειας σε κλιματιστικές μονάδες και κεντρικός έλεγχος μέσω ιστοσελίδας.

1.1 ΤΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

Εικόνα 1: Το Διαδίκτυο των πραγμάτων

Το Internet Of Things ή «Διαδίκτυο των Πραγμάτων» είναι η λύση στον αυτοματισμό και στην εύκολη διαχείριση από τους χρήστες. Είναι ένα δίκτυο όχι μόνο υπολογιστών αλλά και ηλεκτρονικών αντικειμένων τα οποία διασυνδέονται μεταξύ τους. Τα ηλεκτρονικά αντικείμενα αυτά εμπεριέχουν ενσωματωμένα ηλεκτρονικά συστήματα τα οποία μπορεί να είναι διάφορες οικιακές συσκευές, μέσα τηλεπικοινωνίας, μέσα μεταφοράς. Κάθε σύστημα μπορεί να χρησιμοποιεί συστήματα αναγνώρισης ραδιοσυχνοτήτων (RFID), είδη αισθητήρων, δηλαδή απαραίτητη προϋπόθεση για την επιτυχία του καινούριου αυτού διαδικτύου είναι να καταστεί το σημερινό διαδίκτυο πιο ασφαλές και πιο εύχρηστο στον χρήστη. Το Διαδίκτυο των Πραγμάτων είναι η εξυπνότερη λύση για την ολοκλήρωση και αυτοματοποίηση των υπηρεσιών που παρέχουν τα ενσωματωμένα συστήματα. Το διαδίκτυο θα γίνει ένα «νευρικό σύστημα» που θα καταλήγει σε συσκευές με αισθητήρες και ενεργοποιητές "actuators" που θα συνεργάζονται για έξυπνες υπηρεσίες όπως την υγεία, τη διανομή ,τις μεταφορές και κατανάλωση ενέργειας κλπ. Στις μετακινήσεις σύντομα θα έχουμε συστήματα αυτόματης οδήγησης και οργάνωσης των μεταφορικών μέσων για περισσότερη ασφάλεια και οικονομία με την καλύτερη λειτουργία του GPS. Στον τομέα της υγείας προβλέπονται μία σειρά από καινοτομίες, από τη ρομποτική παρακολούθηση των ασθενών, μέχρι την ρομποτική χειρουργική. Στόχος είναι η αυτοματοποίηση της διαχείρισης των φυσικών πόρων όπως για παράδειγμα σε αυτό που ονομάζεται smart grids, αποτελεσματική χρήση εναλλακτικών μορφών ενέργειας. Όλες αυτές θα είναι μερικές εφαρμογές που θα αλλάξουν ριζικά το σημερινό τρόπο ζωής τις ερχόμενες δεκαετίες. Θα υπάρχει πρόσβαση σε όλες τις υπηρεσίες, αποτελεσματικός έλεγχος πόρων, διαδραστικότητα και συνεργασία για την επίτευξη ολοκληρωμένων στόχων. Μεγάλες εταιρίες δεν επικεντρώνονται πλέον στη πώληση

λογισμικού ή υπολογιστών αλλά ερευνούν ολοκληρωμένες λύσεις όπου τα πληροφορικά συστήματα χρησιμοποιούνται για τη βέλτιστη διαχείριση φυσικών πόρων και την ανάπτυξη έξυπνων υπηρεσιών για τη δημιουργία ενός «έξυπνου πλανήτη».[1]

1.1.1.ΑΝΑΦΟΡΑ ΣΤΟ ΜΕΛΛΟΝ

Στο μέλλον, δε θα είναι διόλου απίθανο να έχει ο καταναλωτής τη δυνατότητα να λαμβάνει ειδήσεις και πληροφορίες μέσω οικιακών συσκευών, ή να κατεβάζει σε πραγματικό χρόνο βίντεο σε αυτό-οδηγούμενα αυτοκίνητα. Είναι προφανές ότι οι επερχόμενες τεχνολογικές εξελίξεις θα επαναπροσδιορίσουν ριζικά τις προσδοκίες των καταναλωτών στο κοντινό μέλλον. Ένα από τα σημαντικότερα αναμενόμενα οφέλη του IoT για τους ανθρώπους του marketing, σύμφωνα με την έρευνα, είναι η δυνατότητα συλλογής και ανάλυσης πληροφοριών σχετικών με τις συνήθειες και τις προτιμήσεις του καταναλωτή και, το σημαντικότερο, το εκάστοτε πλαίσιο στο οποίο καταναλώνεται η ενημέρωση ή η ψυχαγωγία.

1.1.2. ΙΔΙΟΤΗΚΟΤΗΤΑ ΚΑΙ ΑΣΦΑΛΕΙΑ

Οι μεγαλύτερες προκλήσεις σχετίζονται με τη διασφάλιση της ιδιωτικότητας και την ασφάλεια στον ιστοχώρο. Η διασφάλιση των προσωπικών δεδομένων είναι ένα θέμα με εκθετικά αυξανόμενο βαθμό δυσκολίας, καθώς το IoT συγκεντρώνει τεράστιες ποσότητες δεδομένων και συνδέει όλο και περισσότερες συσκευές, λογισμικό, μηχανήματα και ανθρώπους. Εάν οι έξυπνες συσκευές παρέχουν χρήσιμα δεδομένα στους παρόχους περιεχομένου, χωρίς να αισθάνεται ο χρήστης ότι παραβιάζεται η ιδιωτικότητα των προσωπικών του δεδομένων, το περιεχόμενο προσαρμόζεται άμεσα στην εκάστοτε στιγμή, ερμηνεύοντας τη διάθεση, τις ανάγκες και τις προθέσεις των χρηστών και εκπέμποντας, παράλληλα, στοχευμένες διαφημίσεις και επικοινωνία, τότε οι δυνατότητες για την ενίσχυση της εμπιστοσύνης των καταναλωτών προς τις διαφημιζόμενες εταιρείες που θα κάνουν χρήση αυτών των τεχνολογιών, έναντι αυτών που θα παραμείνουν στο παραδοσιακό διαφημιστικό μοντέλο, θα είναι τεράστιες.

1.2. M2M-MACHINE TO MACHINE

Η M2M(Machine To Machine) επικοινωνία χρησιμοποιεί τεχνολογίες για να βοηθήσει τόσο σε ασύρματα όσο και σε ενσύρματα συστήματα να συνδεθούν με συσκευές της ίδιας ικανότητας. Δίνει την δυνατότητα σε συσκευές, όπως υπολογιστές, αισθητήρες, ενσωματωμένα συστήματα, κινητά, να επικοινωνούν μεταξύ τους και να παίρνουν αποφάσεις με ελάχιστη ανθρώπινη παρέμβαση έως και καθόλου. Οι συσκευές διαθέτουν πλέον την ευφυΐα να αποφασίζουν αυτόνομα βάσει των δεδομένων που συλλέγουν οι ίδιες ή άλλες συσκευές. Η τεχνολογία αυτή χρησιμοποιεί μια συσκευή (όπως έναν αισθητήρα) για να καταγράψει ένα γεγονός αναλογικά ή ψηφιακά (όπως η θερμοκρασία κλπ.) το οποίο αναμεταδίδεται μέσω ενός δικτύου (ασύρματο, ενσύρματο ή και τα δύο) σε μια εφαρμογή (πρόγραμμα λογισμικού), η οποία αποκωδικοποιεί το καταγεγραμμένο γεγονός σε χρήσιμη πληροφορία . Βασικές εφαρμογές είναι:

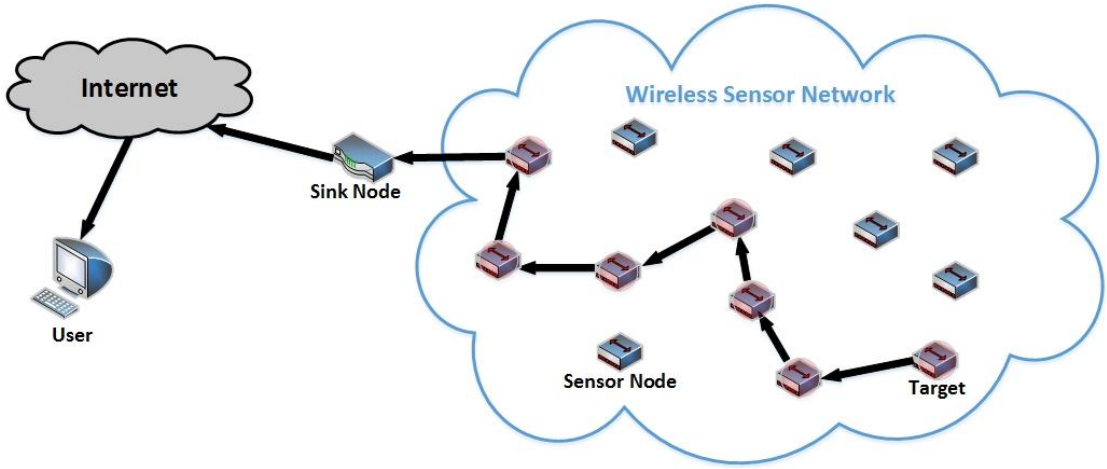
• Σύνδεση μηχανών/συσκευών με άλλες μηχανές, π.χ. απομακρυσμένα περιβάλλοντα παραγωγής (γεωργικός τομέας)

• Σύνδεση μηχανών με τα κέντρα υπηρεσιών(servers), π.χ. αυτοκίνητα που ενημερώνουν τα κέντρα εξυπηρέτησης για θέματα συντήρησης και βλαβών

• Σύνδεση κέντρων υπηρεσιών(servers) με τις μηχανές, π.χ. αυτόματοι πωλητές που αναφέρουν την κατάσταση των αποθεμάτων σε ένα κεντρικό σύστημα καταγραφής(βάση δεδομένων)

• Σύνδεση οχημάτων με μηχανές, π.χ. διαχείριση και τοποθεσία αεροπορίας [2]

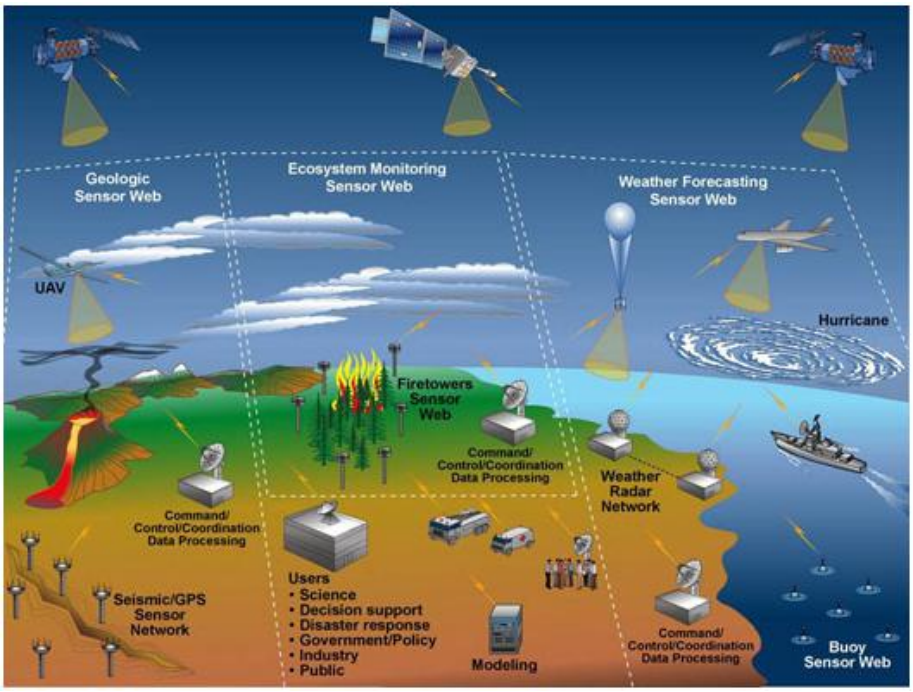
1.3. ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ



Εικόνα 2: Παράδειγμα ασύρματου δικτύου αισθητήρων

Τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks) είναι ενσωματωμένα συστήματα μέτρησης, μετάδοσης και επεξεργασίας πληροφοριών. Βασικό χαρακτηριστικό τους είναι ότι είναι ανεξάρτητοι ενεργειακά αυτόνομοι κόμβοι αισθητήρων, οι οποίοι τοποθετούνται σε μια γεωγραφική έκταση με στόχο να μεταδώσουν πληροφορίες προς ένα κεντρικό κόμβο(co-ordinator) με σκοπό την μελέτη και την έρευνα των αποτελεσμάτων. Έχει δύο βασικά χαρακτηριστικά ,τη δυνατότητα για λήψη δεδομένων(μετρήσεων) και τη δυνατότητα για έλεγχο αντίδρασης/αντιμετώπισης (actuators). Η σημαντικότητα των χαρακτηριστικών αυτών καθορίζεται από το είδος και τις ιδιαίτερες ανάγκες της εφαρμογής. Όταν η εφαρμογή απαιτεί την λήψη δεδομένων σε κάποιο συγκεκριμένο διάστημα χρόνου, πιθανές καθυστερήσεις στην αποστολή των μηνυμάτων δεν αλλάζουν τα δεδομένα, άρα το δίκτυο μπορεί να σχεδιαστεί σύμφωνα με την αύξηση του χρόνου ζωής . Ένα Ασύρματο Δίκτυο Αισθητήρων αποτελείται από πολλούς κόμβους, όπου σε αυτών συνδέονται ένας ή περισσότεροι αισθητήρες. Κάθε τέτοιος κόμβος του δικτύου έχει χαρακτηριστικά κάποια συγκεκριμένα κομμάτια: ένα ραδιο-πομποδέκτη με μια εσωτερική κεραία ή με μια εξωτερική κεραία(για καλύτερη κάλυψη) , ένα μικροελεγκτή(MCU), ένα ηλεκτρονικό κύκλωμα για τη διασύνδεση με τους αισθητήρες και μια πηγή ενέργειας, συνήθως μια μπαταρία. Οι περιορισμοί σε μέγεθος και κόστος έχουν ως αποτέλεσμα αντίστοιχους περιορισμούς σε πόρους όπως ενέργεια, μνήμη, υπολογιστική ταχύτητα και στο εύρος ζώνης των επικοινωνιών.[3]

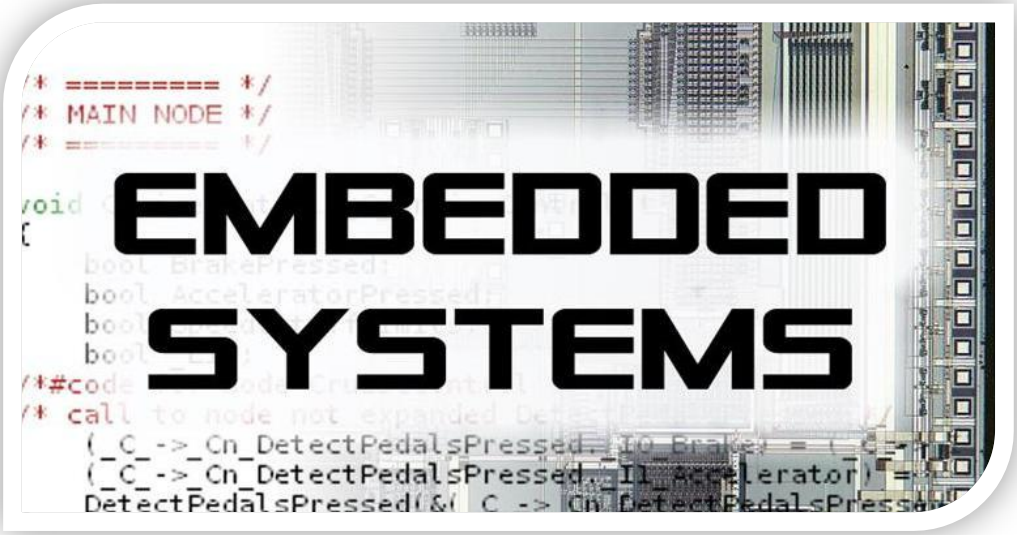
1.3.1. ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΠΡΑΞΗ



Εικόνα 3: Εφαρμογές ασύρματου δικτύου αισθητήρων

- Παρακολούθηση περιοχής
- Περιβαλλοντική / γεωσκόπηση
- Παρακολούθηση της ποιότητας του αέρα
- Παρακολούθηση της ρύπανσης του αέρα
- Ανίχνευση δασικών πυρκαγιών
- Ανίχνευση κατολισθήσεων
- Παρακολούθηση της ποιότητας των υδάτων
- Πρόληψη φυσικών καταστροφών
- Βιομηχανική παρακολούθηση
- Καταγραφή δεδομένων
- Βιομηχανική λογική και έλεγχος των αιτήσεων
- Παρακολούθηση νερού/αποβλήτων υδάτων
- Γεωργία
- Παρακολούθηση έξυπνου σπιτιού

1.4. ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ



Εικόνα 4: Ενσωματωμένα Συστήματα

Τι είναι τα ενσωματωμένα συστήματα;

Ενσωματωμένα συστήματα είναι αυτά στα οποία κάποιος επεξεργαστής λειτουργεί σαν μέρος μίας ολότητας, επιτελώντας συγκεκριμένο έργο, και στον οποίο εν γένει ο χρήστης δεν έχει πρόσβαση για να αλλάξει το πρόγραμμα ή την λειτουργικότητα του συστήματος. Με τον όρο «ενσωματωμένα συστήματα» θα εννοούμε την ενσωμάτωση κάποιου μικροεπεξεργαστή στην λειτουργία ενός ολόκληρου συστήματος με ηλεκτρονικά, μηχανολογικά, και άλλα μέρη, αλλά θα μας αφορούν μόνο οι λειτουργίες του μικροεπεξεργαστή και η διεπαφή αυτού με το εξωτερικό περιβάλλον.

Τι είναι ο μικροελεγκτής;

Ο μικροελεγκτής (αγγλικά, microcontroller) είναι ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει.

Τι είναι ο μικροεπεξεργαστής;

Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (πχ τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς.[4]

1.5. ΕΞΟΙΚΟΝΟΜΗΣΗ ΕΝΕΡΓΕΙΑΣ

Εικόνα 5: Εξοικονόμηση Ενέργειας

ΘΕΡΜΟΜΟΝΩΣΗ

- Κλείστε χαραμάδες σε πόρτες και παράθυρα με μονωτικό υλικό όπως αυτοκόλλητες ταινίες του εμπορίου ή ακόμη και σιλικόνη.
- Μην αερίζετε υπερβολικά τους χώρους που θέλετε να θερμάνετε.
- Προσθέσετε μόνωση στην οροφή του κτιρίου
- Προσθέσετε μόνωση στους τοίχους με φελιζόλ.

ΘΕΡΜΑΝΣΗ ΧΩΡΩΝ

- Αξιοποιείστε την ηλιακή ενέργεια για. Τις ηλιόλουστες χειμωνιάτικες μέρες να αφήνετε τον ήλιο να μπαίνει μέσα από τα νότια παράθυρα.
- Μην καλύπτετε τα θερμαντικά σώματα με οποιοδήποτε τρόπο, γιατί μειώνεται σημαντικά η απόδοση του δίχως λόγω.
- Φροντίστε για τη σωστή ρύθμιση και συντήρηση του καυστήρα και τον καθαρισμό του λέβητα κάθε καλοκαίρι από ειδικούς συντηρητές.

ΨΥΞΗ-ΚΛΙΜΑΤΙΣΜΟΣ ΧΩΡΩΝ

Πριν αποφασίσετε να αγοράσετε κλιματιστικό, καλύτερα να εξετάσετε τους εναλλακτικούς τρόπους με τους οποίους μπορείτε να έχετε δροσιά διότι οι καιροί είναι δύσκολοι. Τα κλιματιστικά καταναλώνουν μεγάλες ποσότητες ηλεκτρικής ενέργειας. Επί πλέον ρυπαίνουν αλλά και θερμαίνουν το περιβάλλον.

- Επισκιάστε όλα σας τα παράθυρα.
- Φυτέψετε δέντρα, κατά προτίμηση φυλλοβόλα για να σκιάσετε το κτίριο σας, αλλά και για να δημιουργήσετε καλύτερο ευνοϊκό ‘’μικροκλίμα’’, όπου αυτό είναι δυνατό.
- Επιλέξετε ανοιχτά και φωτεινά χρώματα στους εξωτερικούς τοίχους και στις οροφές αλλά και στις τέντες.
- Τις ζεστές μέρες να αερίζετε το κτίριο σας.
- Αν οι εξωτερικές συνθήκες και τα ανοίγματα του κτιρίου σας δεν εξασφαλίζουν τον απαραίτητο αερισμό, τοποθετείστε ανεμιστήρες προσαγωγής και απαγωγής του αέρα.

ΦΩΤΙΣΜΟΣ

- Αφήνετε το φυσικό φως του ήλιου να περνάει από όσο το δυνατόν περισσότερες πλευρές των χώρων. Έτσι επιτυγχάνεται μεγαλύτερη επάρκεια και καλύτερη κατανομή της ζέστης.
- Για την καλύτερη ρύθμιση του φωτισμού, προτιμείστε κινητά στόρια, παρά κουρτίνες στα παράθυρα.
- Προτιμήστε ένα χαμηλό φωτισμό και πρόσθετο τοπικό φωτισμό στα σημεία όπου το χρειάζεστε χρησιμοποιώντας λαμπτήρες χαμηλής ενεργειακής κατανάλωσης(lED).
- Να καθαρίζετε τακτικά τα φωτιστικά σώματα και τους λαμπτήρες για σκόνη.

ΘΕΡΜΟΣΙΦΩΝΑΣ – ΖΕΣΤΟ ΝΕΡΟ

- Ανάψτε το θερμοσίφωνα σας όταν χρειάζεστε ανάλογα με τις ανάγκες σας για ζεστό νερό και μην τον αφήνετε αναμμένο άσκοπα για πολλές ώρες.
- Μην αφήνετε τις βρύσες σας να στάζουν αδιάκοπα και μην αφήνετε το ζεστό νερό να τρέχει άσκοπα.

ΟΙΚΙΑΚΕΣ ΣΥΣΚΕΥΕΣ

Ηλεκτρική κουζίνα

- Όταν βράζετε νερό, να σκεπάζετε την κατσαρόλα με το καπάκι της. Θα βράσει γρηγορότερα και με μικρότερη κατανάλωση ενέργειας.
- Αποφεύγετε τις άσκοπες προθερμάνσεις και το συχνό άνοιγμα/κλείσιμο του φούρνου. Κάθε φορά που ανοίγετε την πόρτα του φούρνου χάνεται το 20% της εσωτερικής θερμότητας. Καθαρίζετε τακτικά τα ηλεκτρικά μάτια και το φούρνο για καλύτερη απόδοση .
- Για το ζέσταμα μικρών ποσοτήτων φαγητού προτιμήστε, εφόσον διαθέτετε, το φούρνο μικροκυμάτων γιατί εξοικονομεί ηλεκτρική ενέργεια και χρόνο, αλλά με μέτρο.

Ψυγεία – Καταψύκτες

- Το ψυγείο καταναλώνει αρκετή ενέργεια διότι λειτουργεί όλο το 24ωρο. Η ενεργειακή κατανάλωση που διαθέτουν όλες οι σύγχρονες ηλεκτρικές συσκευές μας δίνει πληροφορίες για την ενεργειακή απόδοση του. Επιλέξετε μια συσκευή με χαμηλή ενεργειακή κατανάλωση.
- Ρυθμίστε το θερμοστάτη του ψυγείου ώστε η θερμοκρασία να είναι 7oC και του καταψύκτη στους -18oC. Έτσι εξοικονομείτε μέχρι και 15% ρεύμα.
- Μη βάζετε ζεστά φαγητά μέσα στο ψυγείο. Καλύτερα να περιμένετε να κρυώσουν πρώτα.

Πλυντήριο ρούχων

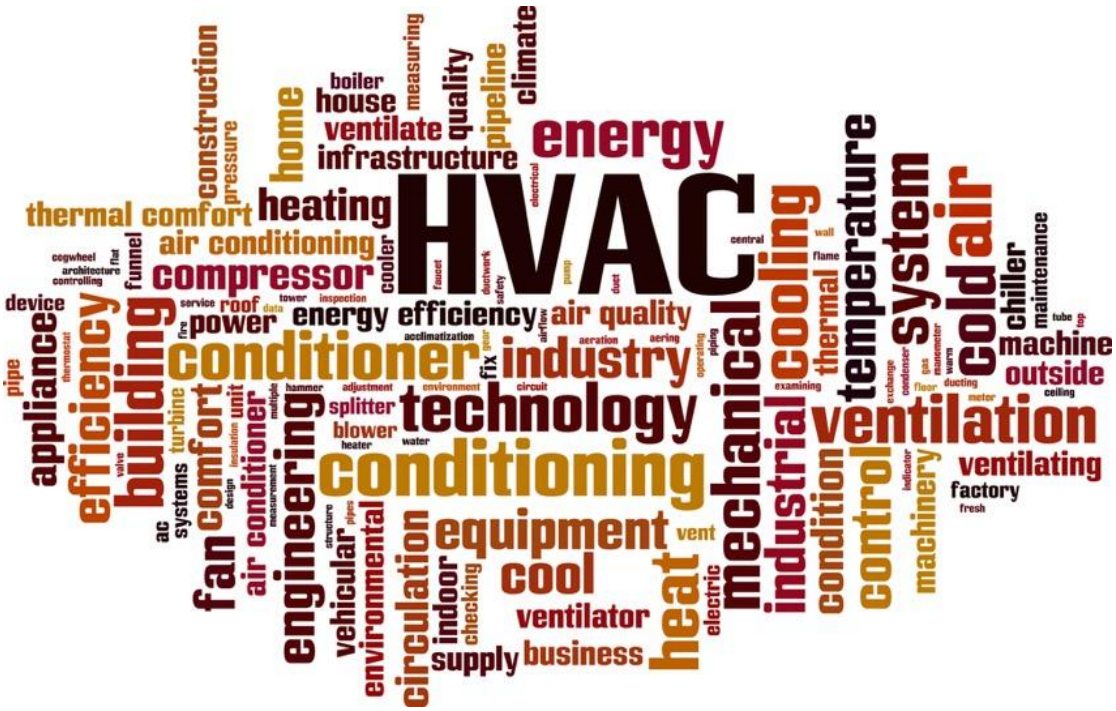
- Επιλέξτε ένα πλυντήριο με μικρή κατανάλωση νερού και ρεύματος. Υπάρχουν αρκετές διαφορές από συσκευή σε συσκευή όπως τα ενεργειακά.
- Φροντίστε να γεμίζετε το πλυντήριο και προτιμάτε να βάζετε όλη τη ποσότητα που χωράει (συνήθως 5-6 κιλά ρούχα).

- Όταν απουσιάζετε, κλείνετε τη παροχή νερού του πλυντηρίου διότι με μια παγωνιά μπορεί να πλημμυρήσετε.

Μικροσυσκευές

- Πριν αγοράσετε μια συσκευή ενημερωθείτε για τη κατανάλωση της σε ρεύμα(watt/h), δεδομένου ότι υπάρχουν διαφορές από συσκευή σε συσκευή (ακόμη και στη λειτουργία αναμονής).
- Μην αφήνετε αναμμένη την τηλεόραση, την καφετιέρα, το σίδερο κ.λ.π.. όταν δεν τα χρειάζεστε, ούτε ακόμα σε κατάσταση αναμονής. [5]

1.5.1. HVAC ΣΥΣΤΗΜΑΤΑ



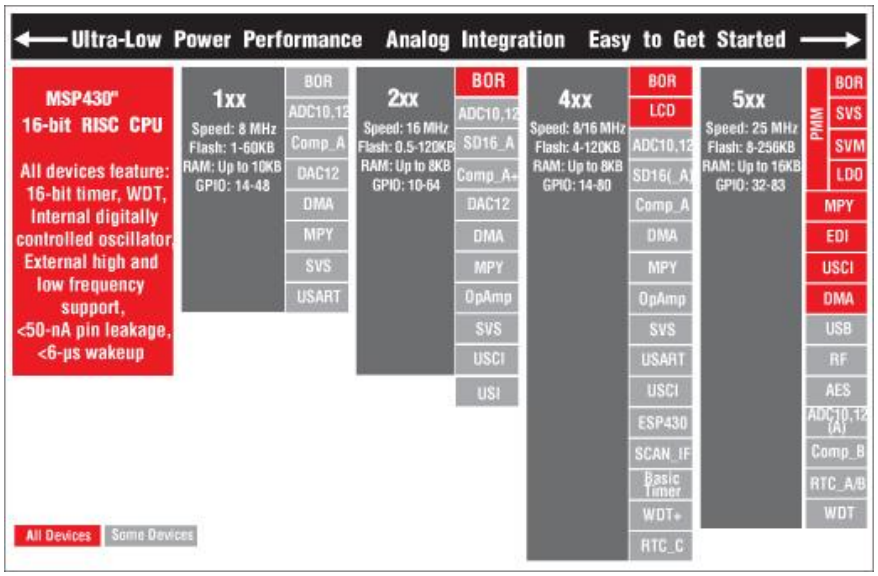
Εικόνα 6: HVAC system

Τα συστήματα HVAC(Heating, Ventilation and Air Conditioning) παρέχουν έλεγχο για τη ρύθμιση της λειτουργίας του συστήματος θέρμανσης ή / και κλιματισμού. Συνήθως μια αισθητήρια διάταξη χρησιμοποιείται για να συγκρίνει την πραγματική κατάσταση (π.χ. θερμοκρασία) με μια επιθυμητή . Στη συνέχεια, το σύστημα ελέγχου αντλεί ένα συμπέρασμα ποια μέτρα πρέπει να ληφθούν (π.χ. ξεκινήσει τον ανεμιστήρα). Τέτοια συστήματα συναντάμε συνήθως σε επαγγελματικούς χώρους διότι κοστίζουν.[6]

ΚΕΦΑΛΑΙΟ 2

Στο συγκεκριμένο κεφάλαιο γίνεται αναφορά στο αναπτυξιακό σύστημα για την υλοποίηση του project . Γίνεται περιγραφή στην οικογένεια του μικροελεγκτή ακόμα και σε αυτόν , στο wifi shield και στα υλικά που χρειάστηκαν. Επίσης γίνεται κοστολόγηση των υλικών και ησυνδεσμολογία μεταξύ τους.

2.1. ΟΙΚΟΓΕΝΕΙΑ ΜΙΚΡΟΕΛΕΓΚΤΩΝ MSP430



Εικόνα 7 : MSP430 Family

Η οικογένεια του MSP430 είναι:

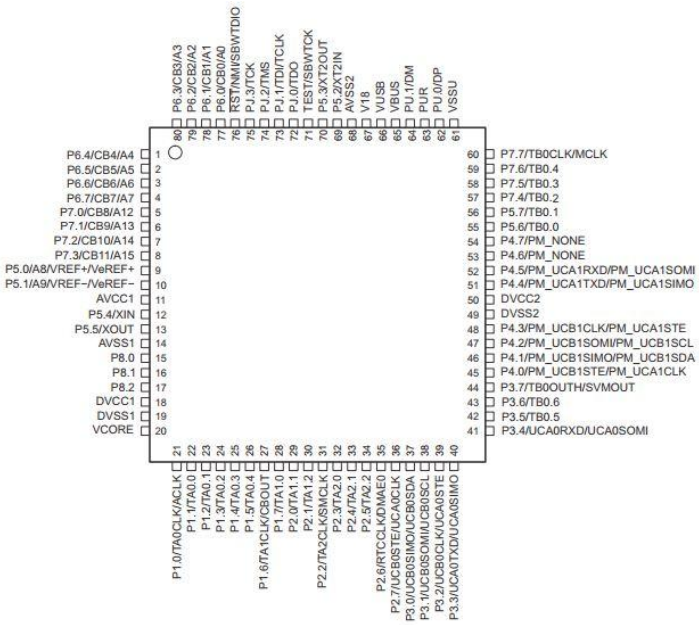
- MSP430x1xx series
- MSP430F2xx series
- MSP430G2xx series
- MSP430x3xx series
- MSP430x4xx series
- MSP430x5xx series
- MSP430x6xx series

2.1.1. MSP430F5529 LAUNCHPAD

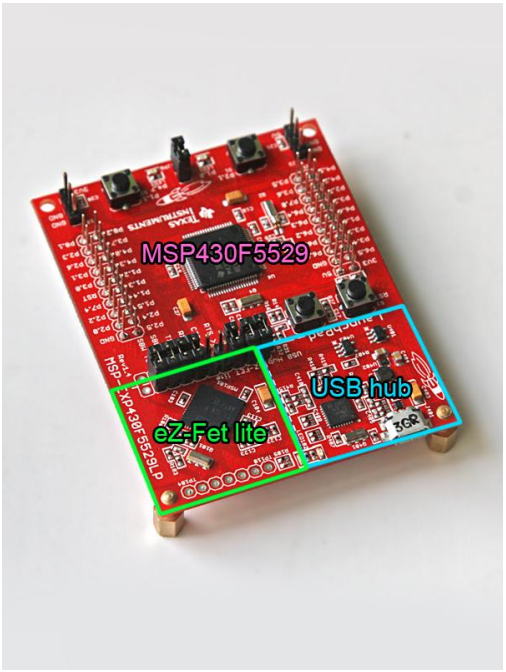
Εικόνα 8 : MSP430F5529

Ο MSP430F5529 περιέχει:

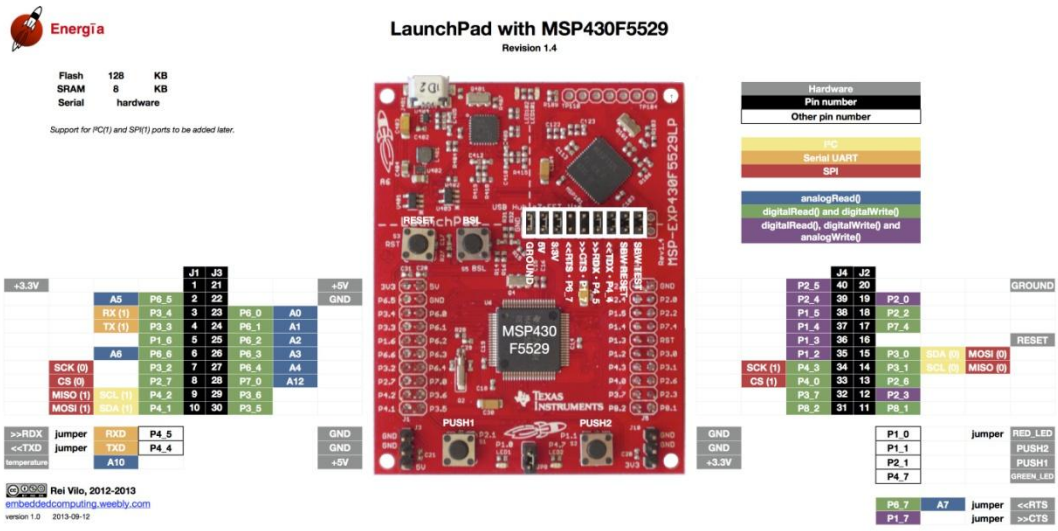
- 16-bit MCU
- 128KB Flash Memory
- 8KB RAM
- 25MHz CPU speed
- integrated USB 2.0 PHY
- 12-bit analog-to-digital converter (ADC)
- timers
- serial communication (UART, I2C, SPI) [7],[8]



Εικόνα 9 : MSP430F5529 Pinout

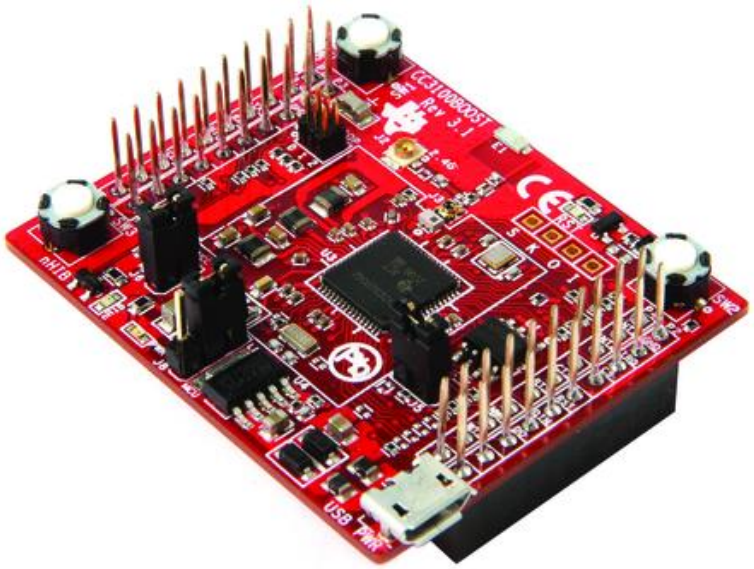


Εικόνα 10 : MSP430F5529 Board

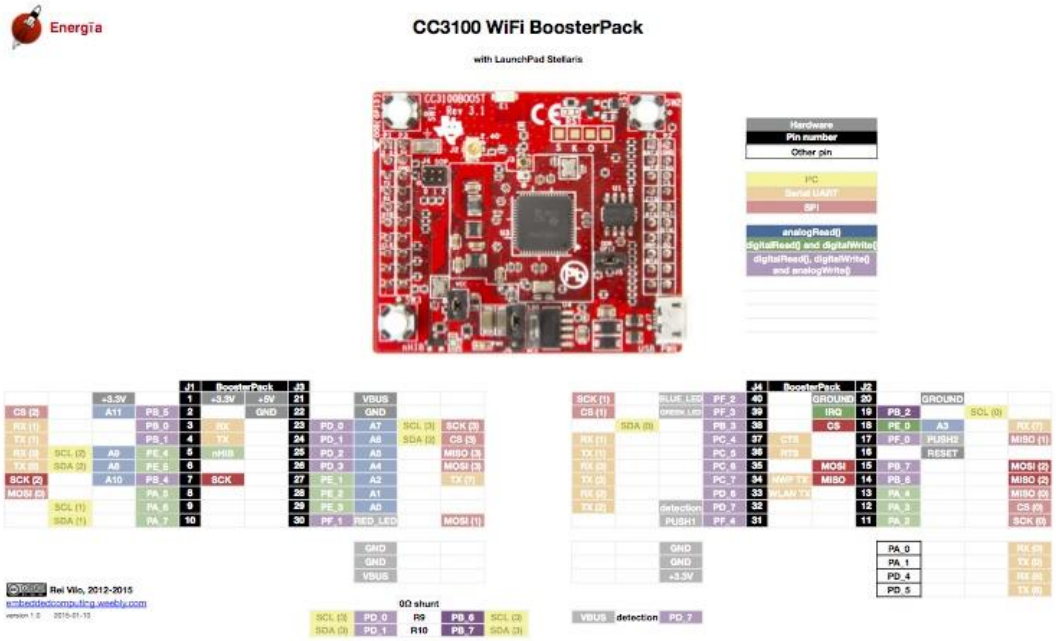


Εικόνα 11 : MSP430F5529 LaunchPad Pinout

2.2. WIFI CC3100 BOOSTERPACK



Εικόνα 12 : WIFI CC3100 BoosterPack



Εικόνα 13 : WIFI CC3100 BoosterPack Pinout

WLAN Receiver Characteristics

T_A = +25°C, V_{BAT} = 2.1 to 3.6 V. Parameters measured at SoC pin on channel 7 (2442 MHz)

Parameter	Condition (Mbps)	Min	Typ	Max	Units
Sensitivity (8% PER for 11b rates, 10% PER for 11g/11n rates)(10% PER)(1)	1 DSSS		–95.7		dBm
	2 DSSS		–93.6		
	11 CCK		–88.0		
	6 OFDM		–90.0		
	9 OFDM		–89.0		
	18 OFDM		–86.0		
	36 OFDM		–80.5		
	54 OFDM		–74.0		
	MCS0 (GF)(2)		–89.0		
	MCS7 (GF)(2)		–71.0		
Maximum input level (10% PER)	802.11b		–4.0		
	802.11g		–10.0		

(1) Sensitivity is 1-dB worse on channel 13 (2472 MHz).
(2) Sensitivity for mixed mode is 1-dB worse.

WLAN Transmitter Characteristics

T_A = +25°C, V_{BAT} = 2.1 to 3.6 V. Parameters measured at SoC pin on channel 7 (2442 MHz).(1)

Parameter	Condition(2)	Min	Typ	Max	Units
Maximum RMS output power measured at 1 dB from IEEE spectral mask or EVM	1 DSSS		18.0		dBm
	2 DSSS		18.0		
	11 CCK		18.3		
	6 OFDM		17.3		
	9 OFDM		17.3		
	18 OFDM		17.0		
	36 OFDM		16.0		
	54 OFDM		14.5		
	MCS7 (MM)		13.0		
Transmit center frequency accuracy		–25		25	ppm

(1) Channel-to-channel variation is up to 2 dB. The edge channels (2412 and 2472 MHz) have reduced TX power to meet FCC emission limits.
(2) In preregulated 1.85-V mode, maximum TX power is 0.25 to 0.75 dB lower for modulations higher than 18 OFDM.

2.3. ΤΑ ΥΛΙΚΑ ΠΟΥ ΧΡΕΙΑΣΤΗΚΑΝ

Εικόνα 14 : Τα υλικά της κατασκευής

Υλικά Κατασκευής:

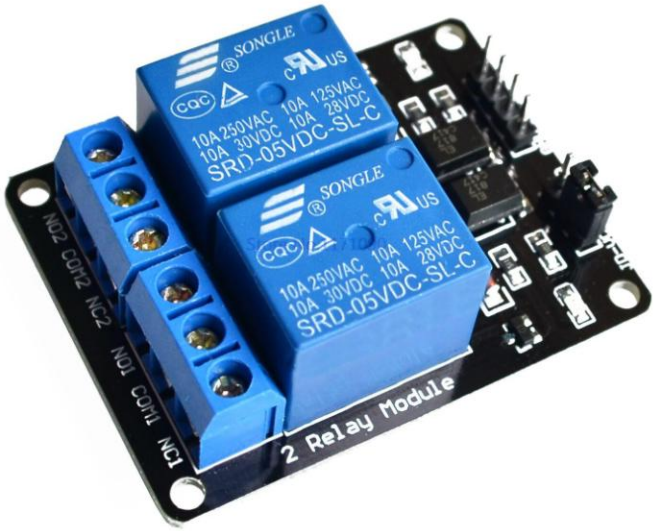
1. MSP430F5529LP LaunchPad
2. CC3100 BoosterPack WIFI
3. Relay 2 channel
4. DHT11 temperature & humidity sensor
5. DS18B20 waterproof temperature sensor
6. Power Supply 5V 2A USB plug
7. Ηλεκτρολογικό Κουτί
8. Καλώδια για συνδεσμολογία
9. joiners

DHT11 temperature & humidity sensor

Εικόνα 15 : DHT11 sensor

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Relay 2 channel



Εικόνα 16 : Relay 2 channel

Specifications:

- 5V SONGLE relay 250V 10A
- Input low level effective.
- Input sign.
- GND power supply.
- VCC signal power supply.
- IN1, IN2 line
- JD-VCC relay power supply.

Dimensions: 51x38.5x18mm

DS18B20 waterproof temperature sensor

Εικόνα 17 : DS18B20 waterproof temperature sensor

Ψηφιακός αισθητήρας θερμοκρασίας DS18B20 1-Wire απο την Maxim-IC. Υπολογίζει βαθμούς Κελσίου (°C) απο -55°C έως 125°C (+/-0.5°C) με ακρίβεια 9 - 12bit. Κάθε αισθητήρας χαρακτηρίζεται απο μία διεύθυνση 16-Bit. Λόγω των διευθύνσεων που έχουν οι αισθητήρες αυτοί μπορούν να συνδεθούν πολλοί μαζί σε ένα μόνο καλώδιο (1-wire protocol).

2.3.1. ΚΟΣΤΟΛΟΓΗΣΗ ΥΛΙΚΩΝ

1. MSP430F5529 LanchPad + CC3100 BoosterPack WIFI = 29.60 €

https://store.ti.com/cc3100boost-msp-exp430f5529lp

2. Relay 2 channel = 1.30 €

http://www.ebay.com/itm/5V-2-Two-Channel-Relay-Module-With-optocoupler-for-Arduino-PIC-ARM-DSP-AVR-/140764956257?hash=item20c63ec661:g:eBoAAOSw-KFXfha9

3. DHT11 temperature & humidity sensor = 0.95 €

http://www.ebay.com/itm/New-DHT11-Temperature-and-Relative-Humidity-Sensor-Module-for-arduino-/310670497097?hash=item4855679549:g:BqQAAOSwqfNXiI37

4. DS18B20 waterproof temperature sensor = 1.38 €

http://www.ebay.com/itm/Waterproof-Digital-Thermal-Probe-or-Sensor-DS18B20-Length-1M-/201544352532?hash=item2eecfac314:g:Q88AAOSwXeJXehjS

5. Power Supply 5V 2A USB plug = 1.89€

http://www.ebay.com/itm/USB-Switching-Power-Supply-Adapter-Charger-AC-100-240V-DC-5V-2A-10W-US-EU-Plug-/361704993157?var=&hash=item54374c3185:m:mWQPBDxSif1tWMTvpj060AA

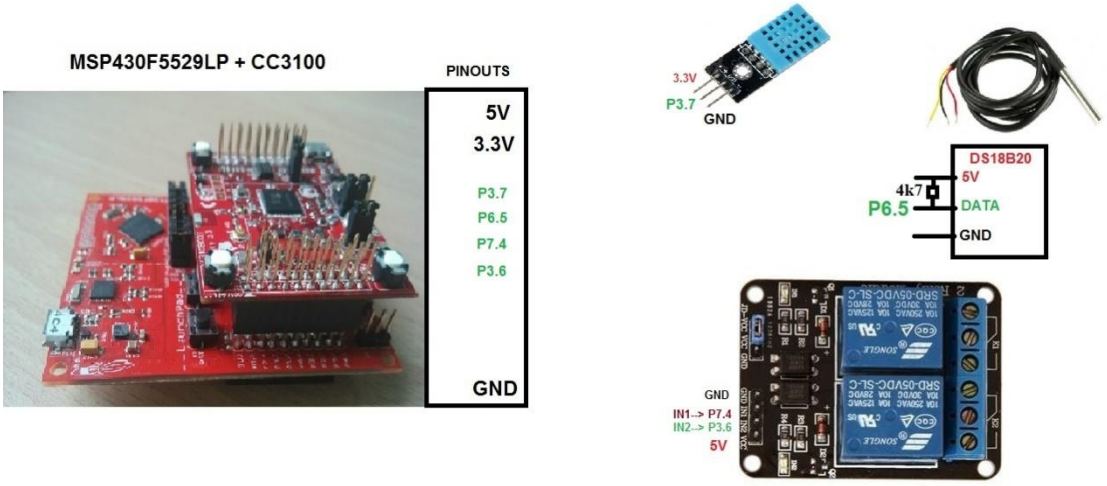
6. Ηλεκτρολογικό Κουτί = 2.5 €

http://www.ebay.com/itm/Waterproof-Junction-Box-Plastic-Electric-Enclosure-Case-85x85x50mm-/162296540606?hash=item25c9a0adbe:g:Iy4AAOSwj85YPmxn

Συνολικό κόστος ανά κόμβο : 37.62 €

Αν αγοράσεις τα υλικά από την Ελλάδα το συνολικό κόστος ανά κόμβο ίσως είναι και διπλάσιο σε τιμή.

2.4. Η ΣΥΝΔΕΣΜΟΛΟΓΙΑ



Εικόνα 18 : Η συνδεσμολογία

Αρχικά θα συνδέσουμε το CC3100 booster pack στον MSP430F5529 Launchpad και στη συνέχεια θα συνδέσουμε του αισθητήρες και τα relay. Ο μικροελεγκτής μας παρέχει τάσεις τροφοδοσίας 3.3volt, 5volt, γείωση και μερικές ψηφιακές εισόδους/εξόδους για δεδομένα.

Με την χρήση των jumper wires θα συνδέσουμε το pin Vcc του αισθητήρα DHT11 με το pin 3,3V του μικροελεγκτή για παροχή σταθερού ρεύματος τάσης 3.3volt και το pin της γείωσης με το pin GND της γείωσης του μικροελεγκτή. Τα πράσινα pins όπως φαίνονται στην παραπάνω εικόνα είναι είσοδοι και έξοδοι ψηφιακών δεδομένων. Το pin που θα συνδέσουμε τον DHT11 είναι το p3.7 . Θα συνδέσουμε τον αισθητήρα DS18B20 για τη μέτρηση της θερμοκρασίας του ψυκτικού υγρού στο pin 5V παροχής σταθερής τάσης 5volt και στο pin GND της γείωσης. Τα δεδομένα θα τα διαβάζουμε από το pin p6.5 . Η αντίσταση 4.7kohm θα συνδεθεί αντίστοιχα στο pin 5V και p6.5.

Η σύνδεση των relay αντίστοιχα στα pins 5V και GND. Η είσοδος in1(HiLow) θα συνδεθεί στο pin p7.4 και η είσοδος in2(OnOff) θα συνδεθεί στο pin p3.6 .

Αφού αποκτήσαμε μία ιδέα για τον μικροελεγκτή και τα υλικά και πως αυτά συνδέονται μεταξύ τους, πάμε να δούμε παρακάτω πως θα προγραμματίσουμε τον μικροελεγκτή.

ΚΕΦΑΛΑΙΟ 3

Στο συγκεκριμένο κεφάλαιο γίνεται ο προγραμματισμός του μικροελεγκτή με το αναπτυξιακό περιβάλλον Energia IDE , γίνεται αναφορά στις βιβλιοθήκες, τα διαγράμματα ροής και τις συναρτήσεις. Ακόμα στο πρωτόκολλο επικοινωνίας MQTT και πως αυτό βοηθάει στην υλοποίηση του project.

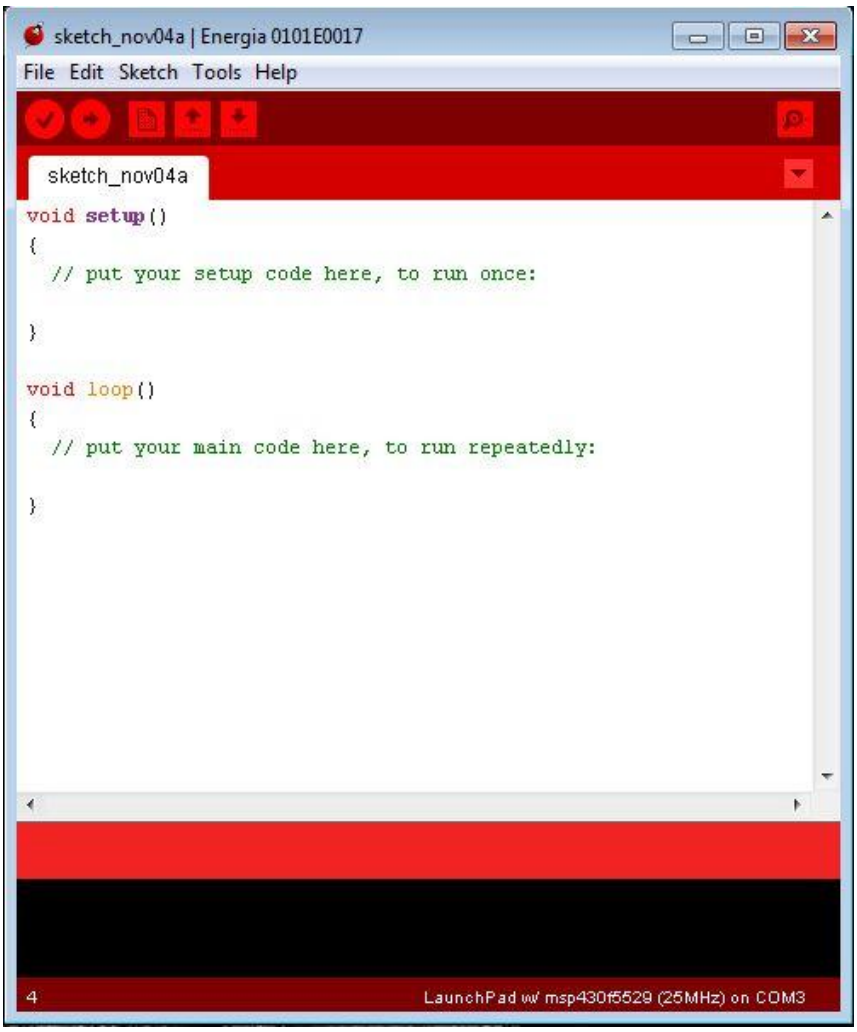
3.1. ENERGIA IDE

Εικόνα 19 : Energia IDE logo

Το Energia IDE είναι μια πλατφόρμα ανοιχτού λογισμικού η οποία ξεκίνησε τον Ιανουάριο του 2012 από τον Robert Wessels, ο οποίος έφερε την Wiring και το Arduino στα δεδομένα της Texas Instruments βασισμένο στον MSP430 LaunchPad. Το Energia IDE υποστηρίζεται σε MAC OS, Windows και Linux. Χρησιμοποιεί τον mspgcc μεταγλωτιστή ο οποίος δημιουργήθηκε από τον Peter Bigot. Υποστηρίζει πολλές βιβλιοθήκες για την εύκολη υλοποίηση του κώδικα.

Ξεκινώντας πρέπει να κατεβάσουμε από τον διαδικτυακό ιστότοπο το Energia IDE http://energia.nu/download/ . Αφού έχει κατέβει πρέπει να αποσυμπιέσουμε το αρχείο energia-0101EXXXX-windows.zip και είμαστε έτοιμοι να γνωρίσουμε την πλατφόρμα . Επίσης πρέπει να κατεβάσουμε τους drivers για τον MSP430F5529 LaunchPad από εδώ http://energia.nu/files/ezFET-Lite.zip. Αυτό μας δίνει την δυνατότητα ο Η/Υ να "δει" από μία COM θύρα το LaunchPad όταν έχει συνδεθεί μέσω της θύρας USB .

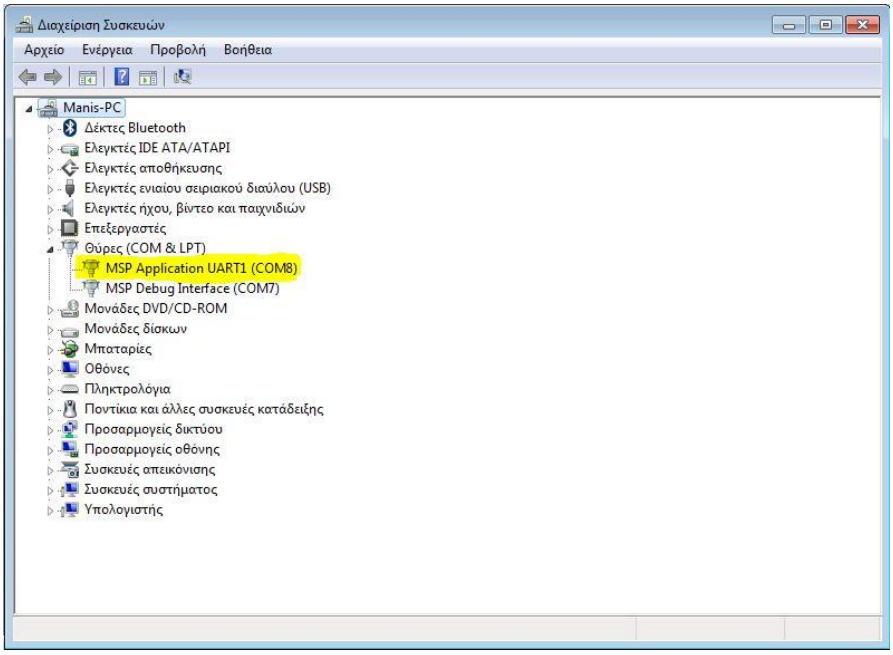
Ξεκινώντας το Energia IDE(Windows χρήστες) κάνοντας διπλό κλικ στο Energia.exe και θα εμφανιστεί ένα άδειο sketch παράθυρο, όπως το παρακάτω. [9]



Εικόνα 20 : Energia IDE

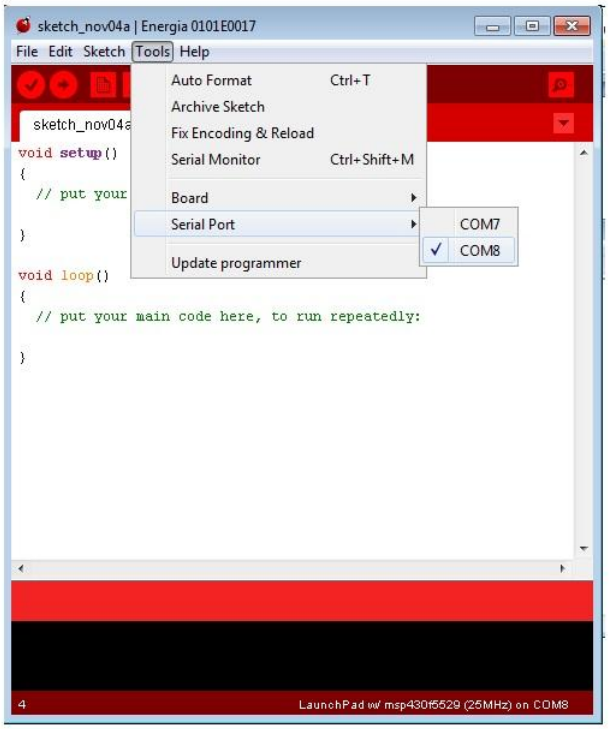
Οι αρχικές ρυθμίσεις που πρέπει να γίνουν ώστε να επικοινωνεί ο MSP430F5529 LaunchPad είναι, αφού ήδη έχουμε συνδέσει στην θύρα τον MSP430F5529 LaunchPad, να εντοπίσει ποια COM θύρα "δόθηκε" για επικοινωνία. Αυτό γίνετε πατώντας δεξί κλικ και ιδιότητες στον " Υπολογιστής" , έπειτα "Διαχείριση Συσκευών" και επιλέγουμε τις "Θύρες (COM & LPT) και

"κρατάμε" την MSP Application UART1 (COMx).

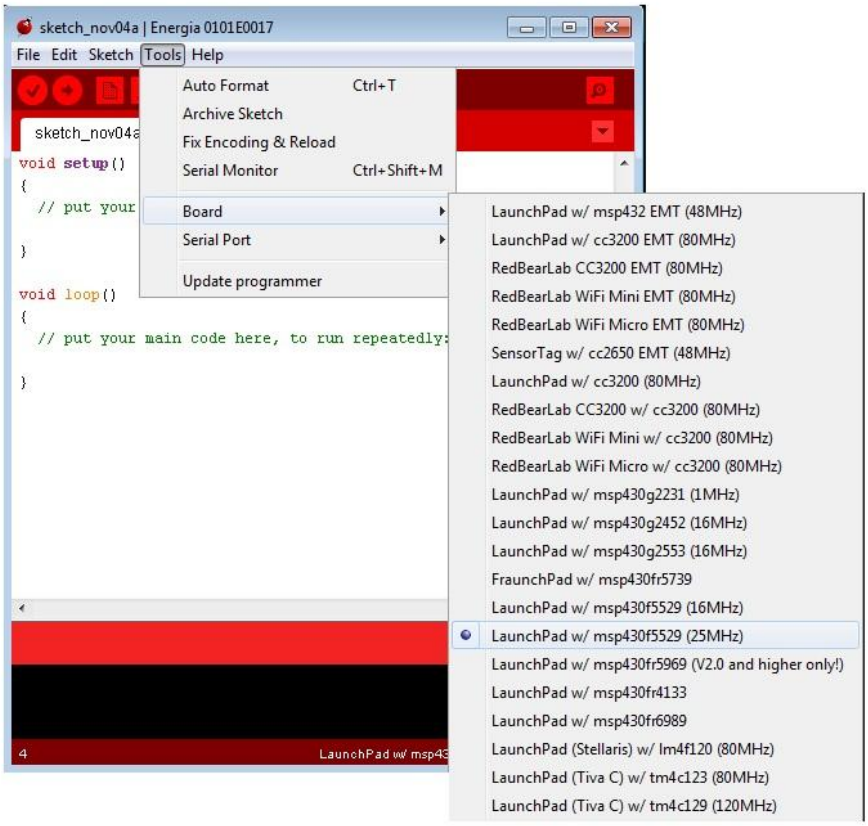


Εικόνα 21 : Θύρα COM

Επιστρέφοντας στο Energia IDE αφού εντοπίσαμε την θύρα COM είμαστε έτοιμοι για τις ρυθμίσεις.

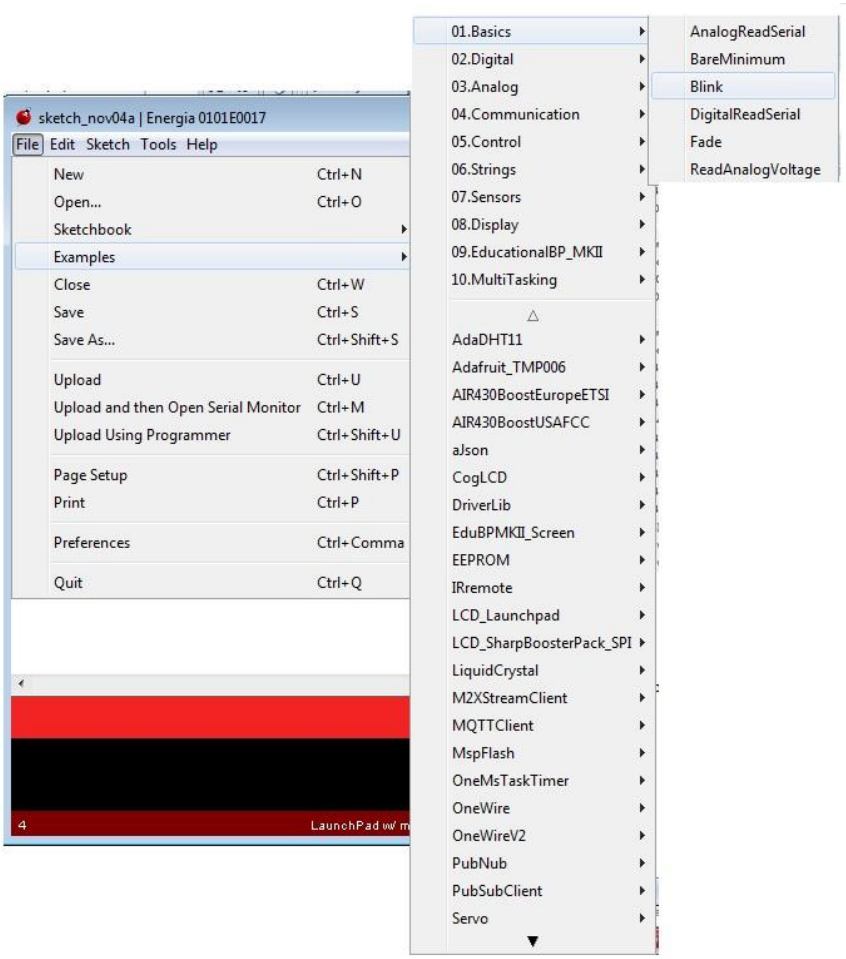


Εικόνα 22 : Energia IDE Θύρα COM



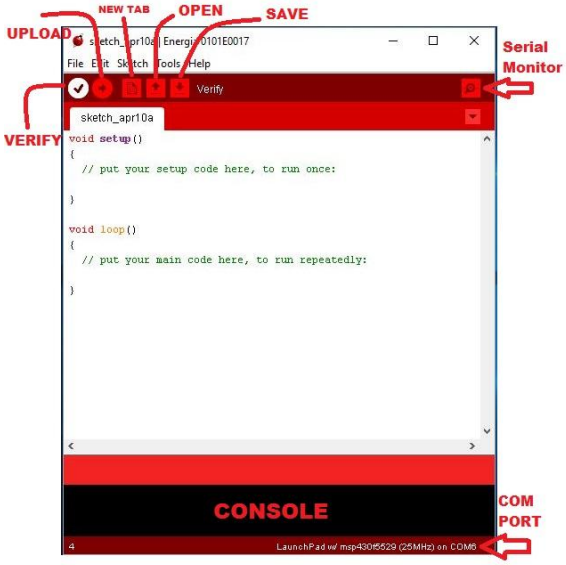
Εικόνα 23 : Επιλογή "πλακέτας"

Και τώρα είμαστε έτοιμοι για την ανάπτυξη του κώδικα μας. Επίσης υπάρχουν και έτοιμοι κώδικες για τις αρχικές μας δοκιμές όπως φαίνεται παρακάτω.



Εικόνα 24 : Enegia IDE Examples

Κάποιες βασικές λειτουργίες του Energia IDE είναι:



Εικόνα 25 : Energia IDE λειτουργίες

3.2. ΟΙ ΒΙΒΛΙΟΘΗΚΕΣ

Οι βιβλιοθήκες για την υλοποίηση του κώδικα ειναι:

- SPI.h

Η συγκεκριμένη βιβλιοθήκη μας επιτρέπει να επικοινωνούμε με SPI (Serial Peripheral Interface) συσκευές με κύρια συσκευή το MSP430F5529 LaunchPad. Είναι ένα σύγχρονο σειριακής μετάδοσης πρωτόκολλο το οποίο χρησιμοποιείτε για την επικοινωνία ανάμεσα σε μικροελεγκτές

- Wifi.h

Η συγκεκριμένη βιβλιοθήκη χρησιμοποιείτε για την ασύρματη επικοινωνία μέσω του CC3100 Booster Pack.

WiFi class

The WiFi class initializes the ethernet library and network settings.

- begin()
- disconnect()
- config()
- setDNS()
- SSID()
- BSSID()
- RSSI()
- encryptionType()
- scanNetworks()
- status()
- getSocket()
- macAddress()

IPAddress class

The IPAddress class provides information about the network configuration.

- localIP()
- subnetMask()

- gatewayIP()

Server class

The Server class creates servers which can send data to and receive data from connected clients (programs running on other computers or devices).

- Server
- WiFiServer()
- begin()
- available()
- write()
- print()
- println()

Client class

The client class creates clients that can connect to servers and send and receive data.

- Client
- WiFiClient()
- connected()
- connect()
- write()
- print()
- println()
- available()
- read()
- flush()
- stop()

UDP class

The UDP class enables UDP message to be sent and received.

- WiFiUDP
- begin()
- available()

- beginPacket()
- endPacket()
- write()
- parsePacket()
- peek()
- read()
- flush()
- stop()
- remoteIP()
- remotePort()

- PubSubClient.h

Η συγκεκριμένη βιβλιοθήκη μας βοηθά να αποστέλλουμε μηνύματα μέσω του πρωτοκόλλου MQTT .

Constructors

- **PubSubClient** ()
- **PubSubClient** (client)
- **PubSubClient** (server, port, [callback], client, [stream])

Functions

- boolean **connect** (clientID)
- boolean **connect** (clientID, willTopic, willQoS, willRetain, willMessage)
- boolean **connect** (clientID, username, password)
- boolean **connect** (clientID, username, password, willTopic, willQoS, willRetain, willMessage)
- void **disconnect** ()
- int **publish** (topic, payload)
- int **publish** (topic, payload, retained)
- int **publish** (topic, payload, length)
- int **publish** (topic, payload, length, retained)
- int **publish_P** (topic, payload, length, retained)
- boolean **subscribe** (topic, [qos])
- boolean **unsubscribe** (topic)

- boolean **loop** ()
- int **connected** ()
- int **state** ()
- PubSubClient **setServer** (server, port)
- PubSubClient **setCallback** (callback)
- PubSubClient **setClient** (client)
- PubSubClient **setStream** (stream) [10]

- DHT.h

  Η συγκεκριμένη βιβλιοθήκη είναι υπεύθυνη για την μέτρηση της θερμοκρασίας και υγρασίας του DHTxx σένσορα.

- DHT(uint8_t pin, uint8_t type)
- readTemperature()
- readHumidity()

- aJSON.h

  Η συγκεκριμένη βιβλιοθήκη μας δίνει την δυνατότητα να δίνουμε "μορφή" στα μηνύματα τα οποία θα αποστέλουμε.

  Parsing JSON

```
aJsonObject* jsonObject = aJson.parse(json_string);
```

  retrieve the value for name:

```
aJsonObject* name = aJson.getObjectItem(root, "name");
```

  The value of name can be retrieved via:

```
Serial.println(name->valuestring);
```

  Finished? Delete the root (this takes care of everything else). This deletes the objects and all values referenced by it.

```
aJson.deleteItem(root);
```

Creating JSON Objects from code

```
aJsonObject *root,*fmt;
   root=aJson.createObject();
   aJson.addItemToObject(root, "name", aJson.createItem("Jack (\"Bee\") Nimble"));
   aJson.addItemToObject(root, "format", fmt = aJson.createObject());
   aJson.addStringToObject(fmt,"type",     "rect");
   aJson.addNumberToObject(fmt,"width",        1920);
   aJson.addNumberToObject(fmt,"height",       1080);
   aJson.addFalseToObject (fmt,"interlace");
   aJson.addNumberToObject(fmt,"frame rate",   24);
```

- OneWire.h

Η συγκεκριμένη βιβλιοθήκη είναι υπεύθυνη για τον θερμοκρασιακό σένσορα DS18B20 .

ds.search(addr) ;

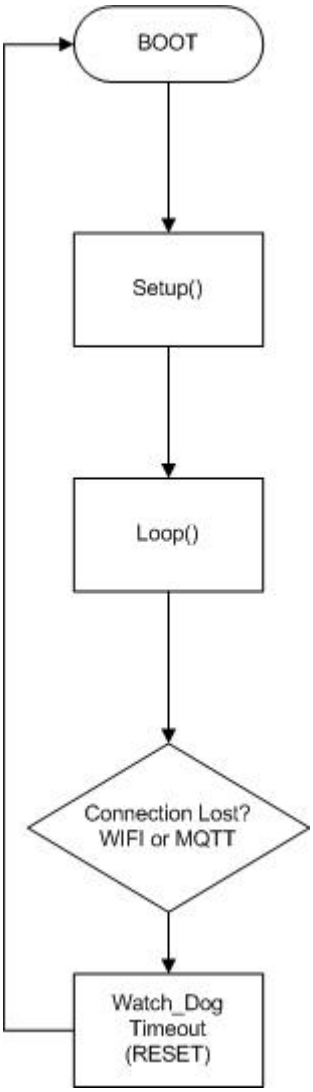
ds.reset();

ds.select(addr);

ds.write();

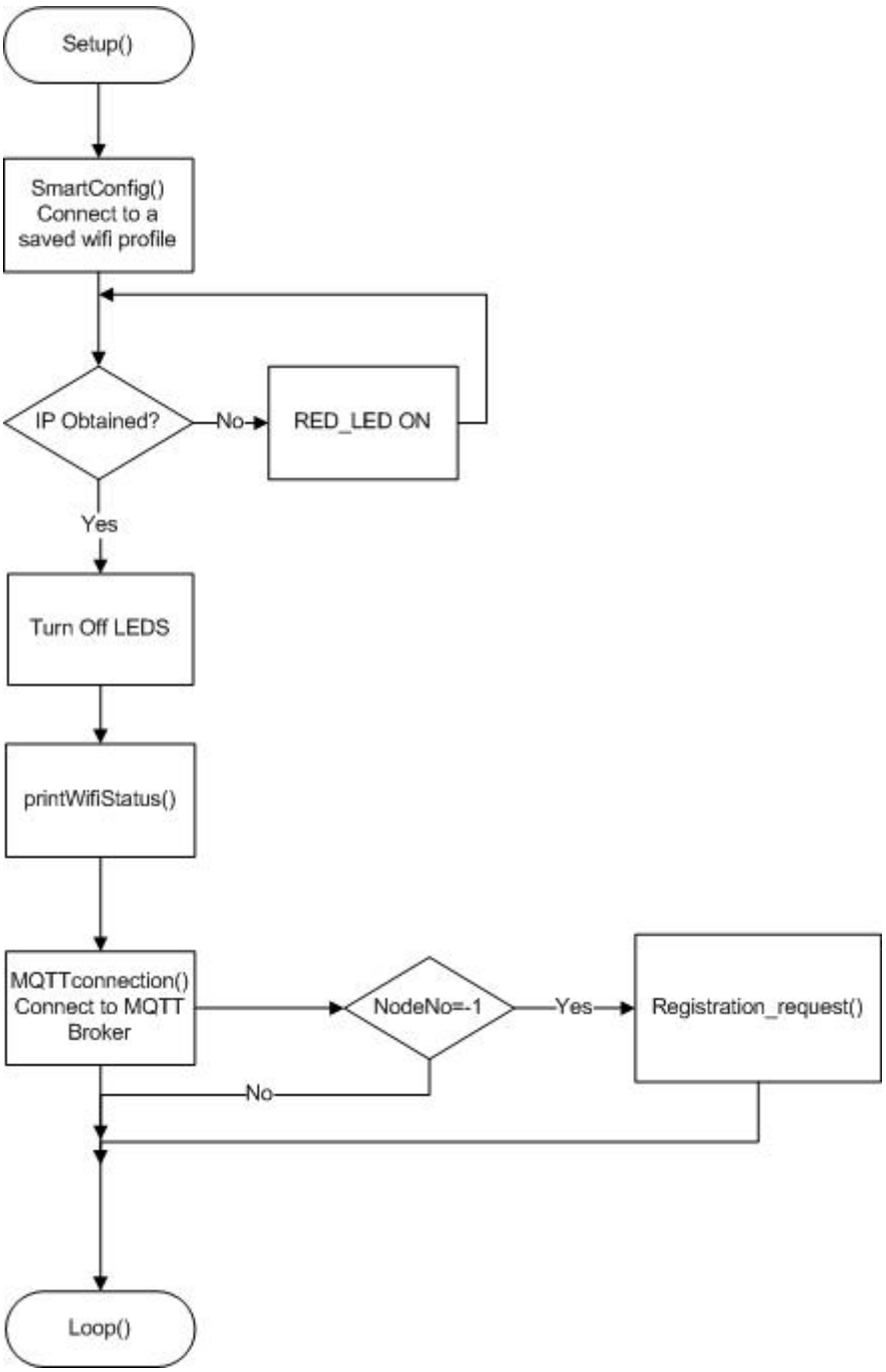
ds.read(); [11]

3.3 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ



Εικόνα 26: main flowchart

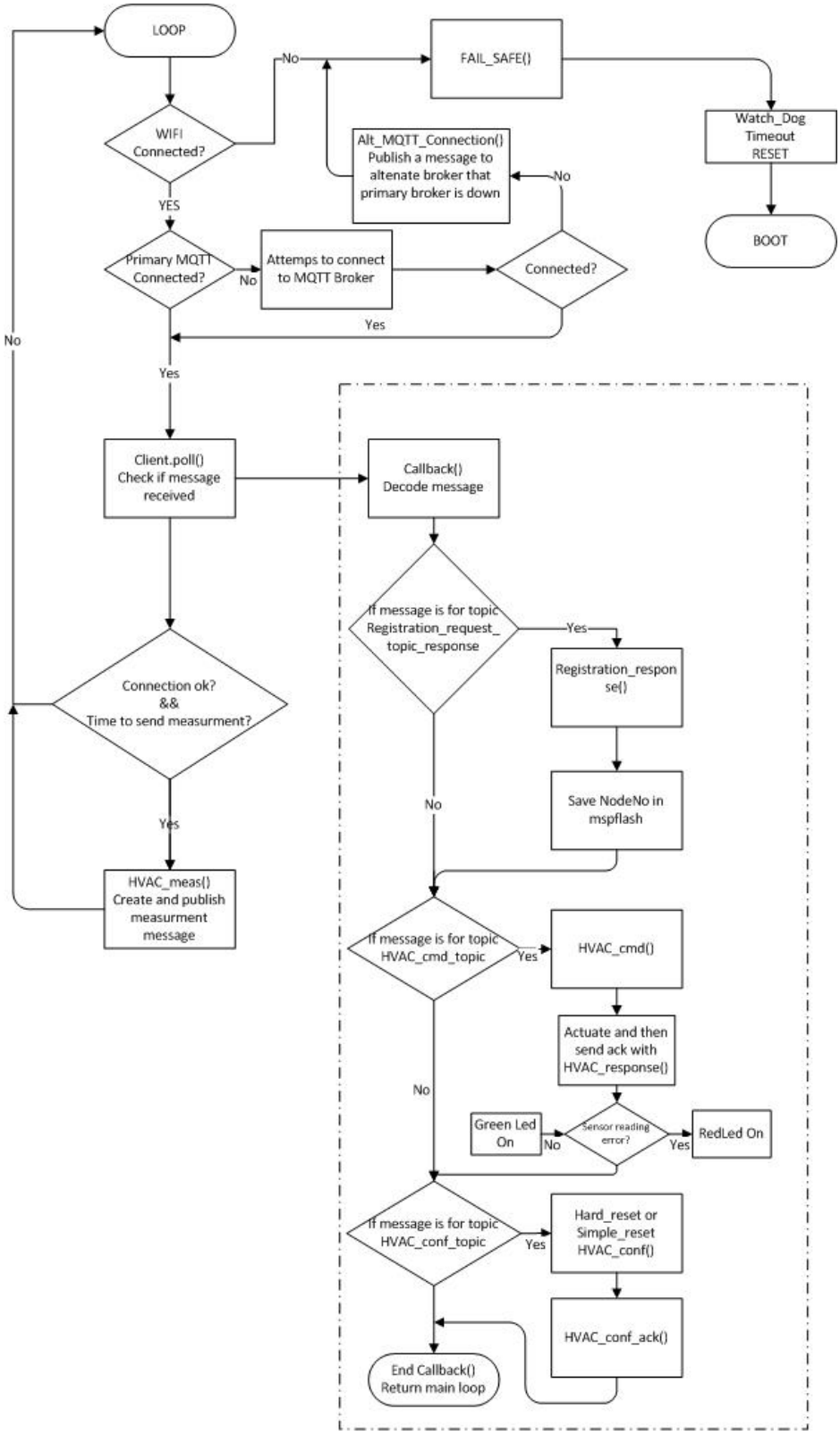
Στην παραπάνω εικόνα διακρίνουμε την κεντρική λειτουργία του κώδικα μέσω του διαγράμματος ροής. Όταν είναι σε BOOT τότε ξεκινάει να τροφοδοτείτε το σύστημα. Στη setup() συνάρτηση γίνεται η δήλωση των pins εισόδου/εξόδου, οι αρχικοποιήσεις των μεταβλητών και των αισθητήρων . Η setup() εκτελείτε μία μόνο φορά εκτός και αν κάνει reset το σύστημα. Η Loop() συνάρτηση είναι η κύρια συνάρτηση η οποία εκτελείται συνέχεια και περιέχει το μεγαλύτερο μέρος του κώδικα. Σε κάθε loop ελέγχουμε αν υπάρχει σύνδεση με το Wifi και με τον Mqtt Broker, αν δεν υπάρχει τότε μηδενίζουμε τον watch dog χρονιστή και κάνει reset το σύστημα. Ας δούμε περεταίρω πληροφορίες για την setup() και την loop() .



Εικόνα 27: setup flowchart

Όταν είμαστε στην setup() μπαίνει σε λειτουργία smartconfig mode η οποία ψάχνει να βρει αποθηκευμένο προφίλ δικτύου, αν δεν βρει τότε περιμένει εκεί μέχρι ο χρήστης να εγκαταστήσει ένα όπως θα δούμε παρακάτω. Αν βρει τότε περιμένει να πάρει IP διεύθυνση και εμφανίζει τα στοιχεία του wifi , όπως το όνομα του δικτύου SSID , την IP και την ισχύς του σήματος RSSI. Έπειτα καλείται η MQTTconnection με την οποία συνδεόμαστε στον primary mqtt broker : mqtt.m2m.ce.teiep.gr , γίνεται subscribe των topics και κάνουμε ανάγνωση της msp flash memory αν έχει πάρει NodeNo ο κόμβος, αν όχι τότε καλείται η Registration_request και αποστέλλει το

μήνυμα για εγγραφή και συνεχίζει στην loop() περιμένοντας Registration_response. Αν έχει ήδη πάρει NodeNo τότε συνεχίζει στην loop() .



Εικόνα 28: loop flowchart

Στην loop() ελέγχουμε κάθε φορά αν υπάρχει σύνδεση με το Wifi αρχικά, αν όχι τότε καλείται η FAIL_SAFE() στην οποία ανάβουμε το κόκκινο led και απενεργοποιεί τα ρελέ για ασφάλεια.

Έπειτα ελέγχουμε αν υπάρχει σύνδεση με τον primary mqtt broker , αν όχι τότε κάνει 5 προσπάθειες κάθε 3 δευτερόλεπτα για επανασύνδεση και αν δεν καταφέρει να συνδεθεί τότε συνδέεται στον alternate mqtt broker : iot.eclipse.org και στέλνει ένα μήνυμα για Alarm ότι η επικοινωνία με τον mqtt broker χάθηκε ,καλείται η FAIL_SAFE() στην οποία ανάβουμε το κόκκινο led και απενεργοποιεί τα ρελέ για ασφάλεια. Σε κάθε loop ελέγχουμε αν έχει έρθει κάποιο μήνυμα στα topics τα οποία έχει κάνει subscribe ,μέσω της client.poll() και αν ήρθε τότε ελέγχουμε το topic στο οποίο ήρθε το μήνυμα . Αν είναι για το Registration_request_topic_response τότε ελέγχει την MAC address αν είναι για το συγκεκριμένο κόμβο το μήνυμα και αποθηκεύει το NodeNo στην msp flash memory. Αν είναι για το HVAC_cmd_topic τότε καλείτε η HVAC_cmd(), ελέγχει το NodeNo για ποιον κόμβο απευθύνεται το μήνυμα και ενεργοποιεί ή απενεργοποιεί τα ρελέ και καλεί την HVAC_response() η οποία στέλνει acknowlegde μήνυμα με την κατάσταση των ρελέ και των leds τα οποία ελέγχουν αν οι αισθητήρες μετράνε σωστά . Αν είναι για το HVAC_conf_topic τότε καλείτε η HVAC_conf(), ελέγχει το NodeNo για ποιον κόμβο απευθύνεται το μήνυμα και κάνει HardReset (σβήνει την mspflash memory-διαγράφει το NodeNo) ή Reset (watch dog timer = 0) και καλεί την HVAC_conf_ack() η οποία στέλνει acknowlegde μήνυμα αν έκανε HardReset ή Reset . Αφού έχει ελέγξει τα topic τότε επιστρέφει στη ροή της κυρίως loop ελέγχοντας αν υπάρχει σύνδεση και αν είναι ώρα για να στείλει μήνυμα μέτρησης(κάθε 1 λεπτό), αν είναι τότε καλεί την HVAC_meas() η οποία στέλνει μήνυμα το οποίο περιέχει το NodeNo , την υγρασία , την θερμοκρασία χώρου, την θερμοκρασία του ψυκτικού υγρού, την κατάσταση των ρελέ, την θερμοκρασία του msp430, το RSSI και το message count .Έπειτα επαναλαμβάνονται τα παραπάνω σε κάθε loop.

3.3.1. ΟΙ ΣΥΝΑΡΤΗΣΕΙΣ

- **Συνάρτηση SmartConfig()**

Εικόνα 29: SimpleLink WIFI

Κάνοντας κλήση στον κώδικα μας τη συνάρτηση smartconfig() μας δίνει την δυνατότητα να συνδέσουμε τον κόμβο μας σε ένα ασύρματο δίκτυο χρησιμοποιώντας την βιβλιοθήκη WiFi.h .Αρχικά τυπώνει στη σειριακή οθόνη το μήνυμα *"Started SmartConfig Mode"*.Σαν προκαθορισμένη λειτουργία έχει να αναμένει από το χρήστη να του δώσει ένα προφίλ δικτύου μέσω της android εφαρμογής *SimpleLink WiFi Starter* όπως θα δούμε παρακάτω. Επειδή όμως αν κάνουμε επανεκκίνηση ή αν βγάλουμε από το ρεύμα ή ακόμα αν κοπεί το ρεύμα του δικτύου τότε ο κατασκευαστής έχει θέσει την συνάρτηση να διαγράφει όλα τα αποθηκευμένα προφίλ δικτύου άρα θα μας ξαναζητήσει να του δώσουμε πάλι ένα προφίλ το οποίο εμείς θέλουμε να το κάνουμε μία φορά και έπειτα να μπαίνει αυτόματα σε αυτό. Αυτό γίνεται αν σχολιάσουμε κάποιον κώδικα από την βιβλιοθήκη WiFi.h όπως φαίνεται παρακάτω:

Στα Windows μεταβαίνουμε στο φάκελο C:\Program Files\energia-0101E0017\hardware\msp430\libraries\WiFi και ανοίγουμε προς επεξεργασία το αρχείο WiFi.cpp. Για την επεξεργασία του αρχείου χρησιμοποιήθηκε το πρόγραμμα *Notepad++* . Πατώντας CTRL+F κάνουμε αναζήτηση της λέξης *"smartconfig"* και βρίσκουμε την συνάρτηση *int WiFiClass::startSmartConfig()* . Εκεί βάζουμε σε σχόλια των κώδικα ο οποίος διαγράφει τα αποθηκευμένα προφίλ δικτύων όπως φαίνεται παρακάτω:

```
// if(sl_WlanProfileDel(WLAN_DEL_ALL_PROFILES) < 0)

// return -1;
```

Επίσης σχολιάζουμε τον κώδικα ο οποίος ενεργοποιεί το smartconfig mode και περιμένει από το χρήστη να δώσει ένα προφίλ δικτύου μέσω της εφαρμογής *SimpleLink WiFi Starter* . Ο κώδικας που θα σχολιαστεί είναι ο παρακάτω:

```
/*
  sl_WlanSmartConfigStart(0,  //groupIdBitmask
      SMART_CONFIG_CIPHER_NONE, //cipher
      0,                  //publicKeyLen
      0,                  //group1KeyLen
      0,                  //group2KeyLen
      NULL,               //publicKey
      NULL,               //group1Key
      NULL);              //group2Key
*/
```

Τέλος σχολιάζουμε των κώδικα ο οποίος είναι προκαθορισμένος από τον κατασκευαστή , η πολιτική σύνδεσης να λειτουργεί σε Auto Mode και SmartConfig Mode ,ενώ εμείς θέλουμε σε Auto Mode μόνο και αυτό γίνεται παρακάτω:

```
//if(sl_WlanPolicySet(SL_POLICY_CONNECTION,
  //SL_CONNECTION_POLICY(1,0,0,0,0),
  //&policyVal,
  //1 /*PolicyValLen*/) < 0) return -1;
```

Η πολιτική ασύρματης σύνδεσης ορίζει 4 επιλογές για να συνδεθείς σε ένα ασύρματο δίκτυο.

- Auto : Η συσκευή προσπαθεί να συνδεθεί σε ένα αποθηκευμένο προφίλ δικτύου βάση προτεραιότητας. Μπορούμε να αποθηκεύσουμε μέχρι 7 προφίλ δικτύου με βάση την προτεραιότητα. Αν έχουμε θέσει την ίδια προτεραιότητα σε διαφορετικά προφίλ τότε η επιλογή του δικτύου γίνεται βάση της ασφάλειας (WPA2 -> WPA -> OPEN) . Αν η ασφάλεια των δικτύων είναι ίδια τότε γίνεται επιλογή βάση του RSSI strength.

- Fast : Η συσκευή συνδέεται στο τελευταίο αποθηκευμένο προφίλ που είχα ξανασυνδεθεί.
- Auto SmartConfig : Το CC3100 ξεκινάει αυτόματα σε SmartConfig λειτουργία
- Any P2P : Χρησιμοποιείται για την σύνδεση σε ένα ορατό WiFi-Direct device, το CC3100 ανιχνεύει και συνδέεται στο πρώτο δίκτυο το οποίο έχει ασφάλεια δικτύου 'SL_SEC_TYPE_P2P_PBC'

Η προκαθορισμένη πολιτική σύνδεσης από τον κατασκευαστή είναι σε Auto και SmartConfig . [12]

Διαδικασία εγκατάστασης προφίλ δικτύου στο CC3100

Εικόνα 30: CC3100 profile setup

Απαραίτητη προϋπόθεση να έχουμε αποθηκεύσει ένα προφίλ δικτύου με την υψηλότερη προτεραιότητα(7=highest για debug) σύνδεσης σε αυτό ώστε όταν ο κόμβος "μπει" σε smartconfig mode, μέσω του κινητού να δημιουργήσουμε ένα σημείο πρόσβασης (hotspot) με το ίδιο SSID και PASSWORD και να συνδεθεί σε αυτό αρχικά. Έπειτα ο χρήστης μπορεί να θέσει ένα καινούριο προφίλ δικτύου με προτεραιότητα σύνδεσης 6 όπως θα δούμε παρακάτω. Αυτό είναι απαραίτητο διότι αν ο κόμβος εγκατασταθεί στα fan coils θα είναι δύσκολο να επέμβουμε κάθε φορά για να αλλάξουμε ένα προφίλ δικτύου.

- Αποθήκευση προφίλ bebug δικτύου:

Για να γίνει αυτό θα πρέπει κρατώντας πατημένο το SW1(switch1) στο CC3100 και πατώντας μια φορά το SW3(switch3) (σβήνει το πράσινο led του CC3100 για ένα δευτερόλεπτο) και αφήνοντας

το SW1 έπειτα από ένα δευτερόλεπτο, το CC3100 μπαίνει σε λειτουργία Access Point Mode. Ανοίγουμε το WIFI του smartphone μας συνδεόμαστε στο δίκτυο mysimplelink-XXXXXX . Ανοίγουμε τον browser και πληκτρολογώντας την διεύθυνση http://192.168.1.1/ μας εμφανίζει το interface του access point cc3100. Πηγαίνουμε στην καρτέλα του Setup και θέτουμε SSID , Security Type, Security Key και Profile Priority = 7 και πατάμε Add .

- Αποθήκευση επιθυμητού προφίλ δικτύου:

Αφού έχουμε αποθηκεύσει το debug προφίλ δικτύου, δημιουργούμε ένα σημείο πρόσβασης (hotspot) μέσω του smartphone μας με το SSID και PASSWORD του debug δικτύου έτσι ώστε ο κόμβος να συνδεθεί σε αυτό. Έπειτα από το smartphone μας στο σημείο πρόσβασης(hotspot) στην λίστα των συνδεδεμένων συσκευών θα παρατηρήσουμε 1 συνδεδεμένη συσκευή. Από εκεί θα δούμε την IP την οποία έχει πάρει ο κόμβος μας π.χ.192.168.43.182 και ανοίγοντας τον browser θα συνδεθούμε σε αυτήν http://192.168.43.182/ . Θα εισέλθουμε στο interface του cc3100 και πηγαίνοντας στην καρτέλα του Setup και θέσουμε το καινούριο SSID , Security Type, Security Key και Profile Priority = 6 και πατάμε Add .

- **Συνάρτηση Registration_request()**

Η συνάρτηση αυτή στέλνει ένα μήνυμα στον MQTT Broker με την MAC address ,το Node firmware,τον τύπο του Node,το Node Hardware version ώστε όταν για πρώτη φορά "ξυπνήσει" το Node να ενημερώσει τον MQTT Broker.Το μήνυμα αποστέλλεται σε JSON μορφή.

```
{
        "NodeID": <NodeID>,   //Node's ID in the form  "00:00:00:00:00:00",
        "Ver": "<Node_FW>",   //Node's firmware
        "Type": "<Node_type>",
//Type of the device (hvac_meas,hvac_act,hvac_comb,hvac_fc,hvac_IR,other)
        "HW Ver" : <Node_HW>, // Node's Hardware version
}
```

- **Συνάρτηση Registration_response()**

Η συνάρτηση αυτή λαμβάνει ένα μήνυμα από τον MQTT Broker , απάντηση στο μήνυμα Registration_request, με την mac address και τον αριθμό του Node.

```
{
    "NodeID": <NodeID>,   //Node's ID in the form "00:00:00:00:00:00",
    "NodeNo": <NodeNo>  //Node's Number
}
```

- **Συνάρτηση HVAC_meas()**

Η συνάρτηση αυτή στέλνει μηνύματα κάθε 60 δευτερόλεπτα στον MQTT Broker τον αριθμό του Node, την υγρασία, την θερμοκρασία, την θερμοκρασία ψυκτικού υγρού, την κατάσταση του OnOff ρελέ, την ταχύτητα του fan coil ,την θερμοκρασία του msp430, την ισχύς σήματος του wifi και το message count.

```
{
    "NodeNo": <Node_No> ,    //integer
    "Humidity": <HVAC_hum>,  //integer
    "Temperature": <HVAC_temp>,  //integer
    "CoolantTemp": <HVAC_fluid_temp>, //integer
    "OnOff": <HVAC_Unit_OnOff>,  //boolean
    "HiLow":  < HVAC_Unit_fan>,   //boolean
    "Msp_Temp":<internaltemp>, //integer
    "RSSI": <rssi>, // long
     "msgCount": 1  //integer
}
```

- **Συνάρτηση HVAC_cmd()**

Η συνάρτηση αυτή λαμβάνει ένα μήνυμα από τον MQTT Broker κάποια τυχαία χρονική στιγμή, δηλαδή όταν ο χρήστης θέλει να αλλάξει την κατάσταση του ρελέ π.χ.

{

"NodeNo": <Node_No>, //integer

"OnOff": <HVAC_Unit_OnOff>, //boolean

"HiLow": < HVAC_Unit_fan> //boolean

"msgCount": 1, //integer

}

- **Συνάρτηση HVAC_cmd_response()**

Η συνάρτηση αυτή στέλνει μήνυμα απάντησης του HVAC_cmd μηνύματος στον MQTT Broker.

{

"NodeNo": <Node_No>, //integer

"OnOff": <HVAC_Unit_OnOff>, //boolean

"HiLow": < HVAC_Unit_fan> //boolean

"LEDR" : true, //boolean

"LEDG" : true, //boolean

"msgCount": 1 //integer

}

- **Συνάρτηση callback()**

Η συνάρτηση αυτή καλείτε όταν ο κόμβος δεχθεί μήνυμα από τον broker στα topic τα οποία έχουμε κάνει subscribe, τα οποία είναι :

- Topic = "m2mce/hvac/registration_response"
- Topic = "m2mce/hvac/ce/act"
- Topic = "m2mce/hvac/conf"

- **Συνάρτηση Alt_MQTT_Connection()**

Η συνάρτηση αυτή μας δίνει την δυνατότητα στη περίπτωση που χαθεί η επικοινωνία με τον κόμβο να συνδεθεί σε έναν εναλλακτικό broker και να αποστείλει ένα μήνυμα στο topic = "m2mce/hvac/alarm" ότι η επικοινωνία με τον πρωτεύον broker χάθηκε.

{

"NodeNo": <Node_No>, //integer

"ALARM" : "Connection with Primary MQTT Broker Lost"

}

- **Συνάρτηση FAIL_SAFE()**

Η συνάρτηση αυτή μας παρέχει ασφάλεια στο σύστημα όταν χαθεί η επικοινωνία με το wifi ή με τον mqtt broker , έτσι ανάβει το κόκκινο led και κλείνει τους ρελέδες.

- **Συνάρτηση printWifiStatus()**

Η συνάρτηση αυτή εμφανίζει στην σειριακή οθόνη λεπτομέρειες του ασύρματου δικτύου στο οποίο συνδεθήκαμε, όπως το όνομα του(SSID) , την IP διεύθυνση και την ένταση του σήματος(RSSI).

- **Συναρτήσεις για την MSPFLASH memory**
  - Συνάρτηση doRead()
    Είναι υπεύθυνη να διαβάσει από την flash memory το Node number το οποίο αποθηκεύτηκε.
  - Συνάρτηση doWrite()
    Είναι υπεύθυνη να κάνει εγγραφή στην flash memory το Node number
  - Συνάρτηση doErase()
    Είναι υπεύθυνη να καθαρίσει την flash memory

- **Συνάρτηση HVAC_conf()**

Η συνάρτηση αυτή οποιαδήποτε χρονική στιγμή αν δεχθεί μήνυμα από τον broker ότι χρειάζεται να κάνει ο κόμβος "ολική επανεκκίνηση" η οποία σβήνει από την μνήμη το nodeno ή απλή επανεκκίνηση.

```
{
        "NodeNo":2, // integer
        "HardReset":true, // boolean  (true make hard reset)
        "Reset":true // boolean (true make simple reset)
}
```

- **Συνάρτηση HVAC_conf_ack()**

Η συνάρτηση αυτή στέλνει μήνυμα acknowledge στο παραπάνω μήνυμα.

```
{
        "NodeNo":2, // integer
        "HardReset":true, // boolean  (true make hard reset)
        "Reset":true // boolean (true make simple reset)
}
```

- **Συνάρτηση GetTempX10(int)**

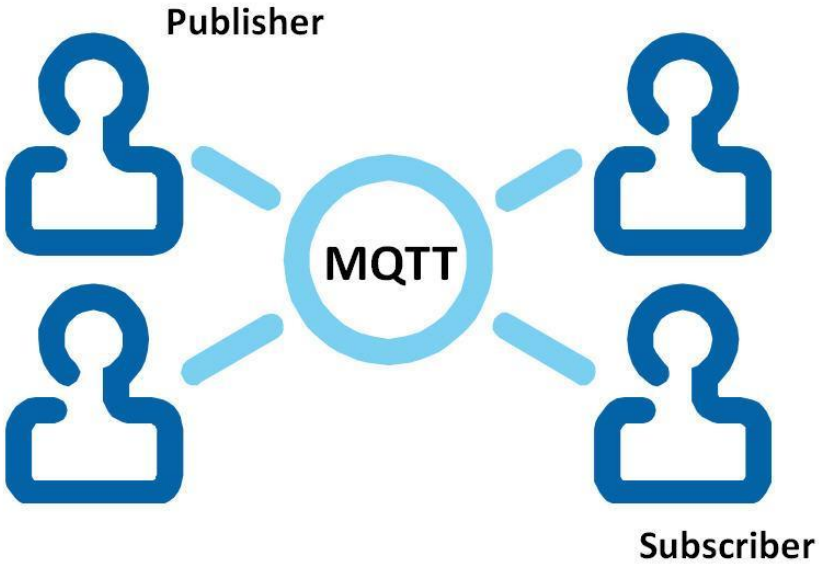
Η συνάρτηση αυτή παίρνει σαν όρισμα έναν ακέραιο αριθμό, ο οποίος για GetTempX10(0) θα κάνει 1 ανάγνωση της εσωτερικής θερμοκρασίας του msp430f5529 , αν GetTempX10(3) θα κάνει 8 αναγνώσματα της εσωτερικής θερμοκρασίας, έτσι υπολογίζοντας τον μέσο όρο θα είναι πιο σωστή η ένδειξη της εσωτερικής θερμοκρασίας.

- **Συνάρτηση readOWsensor()**

Η συνάρτηση αυτή κάνει ανάγνωση της θερμοκρασίας του DS18B20 αισθητήρα. Σαν προκαθορισμένη ανάλυση του αισθητήρα είναι σε 0.0625°C και χρόνος μέτρησης 750ms .

http://www.homautomation.org/2015/11/17/ds18b20-how-to-change-resolution-9101112-bits/

3.4. MQTT ΠΡΩΤΟΚΟΛΛΟ



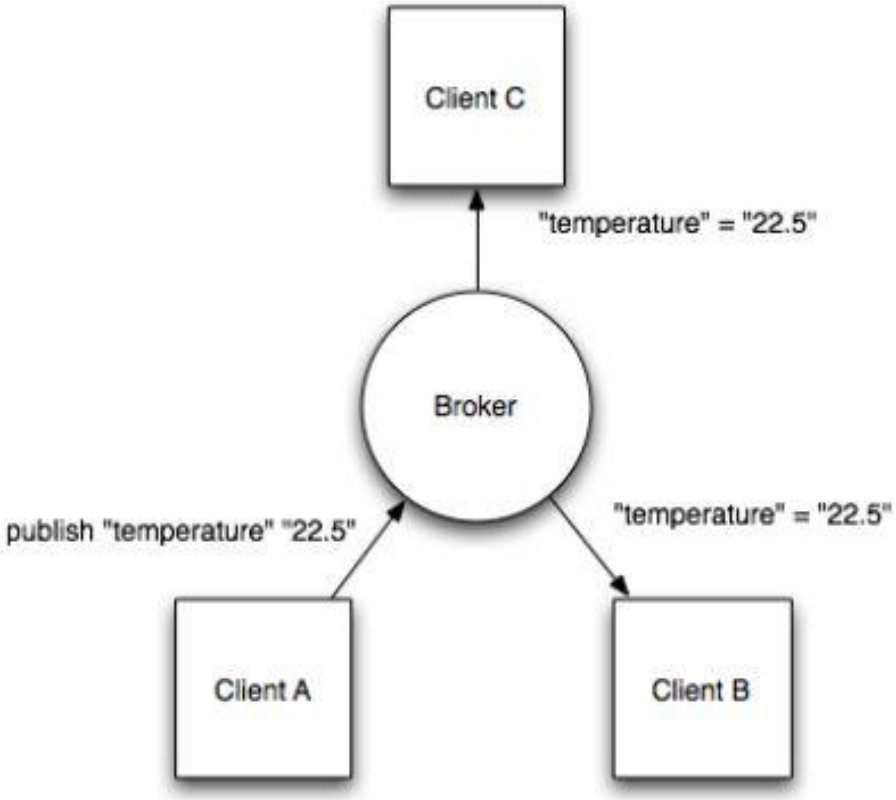
Εικόνα 31: MQTT

Τι είναι το MQTT πρωτόκολλο;

Το MQTT είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων πάνω από το TCP/IP, με προεπιλεγμένη θύρα τη 1883. Είναι ελαφρύ, απλό και εύχρηστο στην εφαρμογή του, χαρακτηριστικά που το κάνουν ιδανικό για M2M επικοινωνίες και IoT εφαρμογές όπου απαιτείται περιορισμένος όγκος κώδικα και το εύρος ζώνης του δικτύου είναι περιορισμένο. Ακόμη η χαμηλή κατανάλωση ενέργειας είναι άλλος ένας λόγος που συχνά επιλέγεται. Παράδειγμα χρήσης του το Facebook/messenger.

Το μοντέλο του Publish/Subscribe:

Το publish/subscribe (pub/sub) μοντέλο έρχεται ως εναλλακτική στο παραδοσιακό client-server μοντέλο όπου δύο συσκευές επικοινωνούν απευθείας. Εδώ ο client που στέλνει ένα μήνυμα (pub) δεν ξέρει την ύπαρξη του παραλήπτη client (sub). Υπάρχει ο broker, ο οποίος είναι γνωστός τόσο στον αποστολέα (publisher) όσο και στον παραλήπτη (subscriber). Η δουλειά του broker είναι να λαμβάνει όλα τα εισερχόμενα μηνύματα και να τα προωθεί κατάλληλα ώστε συγκεκριμένα μηνύματα να φτάνουν σε συγκεκριμένους παραλήπτες.



Εικόνα 32 : Publish/Subscribe μοντέλο

3.4.1. Η ΣΥΝΔΕΣΗ CLIENT ΜΕ BROKER

Τι είναι ο Client;

Ο όρος MQTT client συμπεριλαμβάνει τόσο τον publisher όσο και τον subscriber. Στην γενική περίπτωση ένας MQTT client μπορεί να είναι και τα δύο. Ένας client μπορεί να είναι από ένας μικροελεγκτής μέχρι ένα πλήρες υπολογιστικό σύστημα το οποίο έχει μια MQTT βιβλιοθήκη και είναι συνδεδεμένος σε ένα MQTT broker πάνω από οποιοδήποτε δίκτυο.

Τι είναι ο Broker;

Η καρδιά οποιουδήποτε pub/sub μοντέλου είναι ο broker. Ανάλογα με την υλοποίηση ο broker μπορεί να διαχειριστεί ταυτόχρονα χιλιάδες συνδέσεις. Η πρωταρχική του δουλειά είναι να λαμβάνει, να φιλτράρει και να προωθεί σωστά τα μηνύματα.

Η σύνδεση του Client με τον Broker:

Αρχικά τόσο ο client όσο και ο broker για να τρέξουν χρειάζονται μια στοίβα TCP/IP. Η MQTT σύνδεση γίνεται πάντα από ένα client και το broker, ποτέ client με client. Η σύνδεση αρχικοποιείται με το client να στέλνει ένα CONNECT μήνυμα και ο broker να απαντά με ένα CONNACK και τη κατάσταση σύνδεσης. Όταν η σύνδεση εγκατασταθεί ο broker θα τη διατηρήσει μέχρι ο client να στείλει εντολή αποσύνδεσης ή να χάσει τη σύνδεση του (από το δίκτυο). Ένα CONNECT μήνυμα μεταξύ άλλων περιλαμβάνει:

• clientId - Ένα μοναδικό αναγνωριστικό για κάθε MQTT client. Αν δεν οριστεί από το client ο broker θα του δώσει ένα τυχαίο αριθμό.

•Clean Session - Αυτή η μεταβλητή δηλώνει κατά πόσο ο client θέλει μόνιμη σύνδεση (persistent session) ή όχι. Τέτοια σύνδεση (Clean Session=false) σημαίνει ότι ο broker θα αποθηκεύσει όλα τα subscriptions και όλα τα χαμένα μηνύματα (με QoS>0) που αφορούσαν αυτό τον client ενώ ήταν αποσυνδεδεμένος.

• Username/Password - Αν ο broker ρυθμιστεί κατάλληλα ο client πρέπει να πιστοποιήσει την αυθεντικότητα του.

• Will Message - Έχουμε την δυνατότητα κατά την σύνδεση να ρυθμίσουμε ένα μήνυμα και όταν ο client αποσυνδεθεί αναπάντεχα τότε ο broker θα αναλάβει να το προωθήσει κατάλληλα στην θέση του client σαν τελευταίο μήνυμα.

• KeepAlive - Είναι το χρονικό περιθώριο στο οποίο δεσμεύεται ο client να στέλνει ένα PING μήνυμα και ο broker να στέλνει μια PING απάντηση ώστε και οι 2 πλευρές να ξέρουν ότι η σύνδεση είναι ενεργή.

3.4.2. PUBLISH/SUNSCRIBE

- **Publish**

  Ένας client στέλνει μηνύματα σε συγκεκριμένα topics. Άρα μια εντολή publish πρέπει να περιέχει το topic και το μήνυμα. Το MQTT όμως δεν ασχολείται με το περιεχόμενο και τη δομή του μηνύματος, είναι δουλειά του χρήστη να αποφασίσει αν θέλει δυαδικά δεδομένα, απλό κείμενο ή σε κάποιο τύπο όπως JSON και XML. Πιο συγκεκριμένα μια εντολή publish περιέχει τουλάχιστον τα παρακάτω:

  • Topic Name - Μια συμβολοσειρά ιεραρχικά δομημένη με το χαρακτήρα ’/’.

  • Quality of Service (QoS) - Μπορεί να είναι 0,1 ή 2. Κάθε επίπεδο παρέχει διαφορετική εγγύηση για την αποστολή του μηνύματος. Λεπτομέρειες πιο κάτω.

• Retain Flag - Αν η τιμή αυτής της μεταβλητής είναι αληθής το μήνυμα θα αποθηκευτεί από το broker ως τελευταία γνωστή τιμή και νέοι subscribers θα λαμβάνουν αυτό το μήνυμα αμέσως μετά τη σύνδεσή τους.

• Payload - Εδώ αποθηκεύεται το μήνυμα σε όποια δομή επιθυμεί ο χρή στης.

• Packet Identifier - Μοναδικό αναγνωριστικό μεταξύ broker και client ώστε να ξέρουν την πορεία του μηνύματος. Έχει νόημα για QoS>0. Επίσης καθορίζεται αυτόματα από την αντίστοιχη MQTT βιβλιοθήκη.

• DUP flag - Καθορίζει αν αυτό το μήνυμα είναι αντίγραφο κάποιου άλλου και ξαναστέλνεται επειδή το προηγούμενο δεν έφτασε στον παραλήπτη. Έχει νόημα για QoS>0. Είναι μέρος ενός εσωτερικού μηχανισμού για αξιόπιστη αποστολή-παραλαβή των μηνυμάτων.

- **Subscribe**

  Η αποστολή μηνυμάτων δεν έχει νόημα αν δεν υπάρχει κάποιος να τα παρα λάβει. Ένας client χρειάζεται να εκτελέσει την εντολή subscribe για να λάβει τα μηνύματα. Η εντολή subscribe είναι απλή και περιλαμβάνει τα ακόλουθα:

  • Packet Identifier - Μοναδικό αναγνωριστικό μεταξύ broker και client ώστε να ξέρουν την πορεία του μηνύματος. Αντίστοιχο με αυτό στην publish εντολή.

  • Λίστα από Subscriptions - Κάθε subcription είναι ένα ζεύγος από topic και επίπεδο QoS. Αν υπάρχουν πολλαπλά subscriptions για το ίδιο topic λαμβάνεται υπόψη αυτό με το μεγαλύτερο QoS.

  Κάθε subscription θα επικυρωθεί από το broker στέλνοντας πίσω ένα SUBACK μήνυμα.

3.4.3. ΕΠΙΠΕΔΑ QOS

Το επίπεδο QoS(Quality of Service) είναι μια συμφωνία μεταξύ αποστολέα και παραλήπτη για το πόσο εγγυημένη είναι η παράδοση ενός μηνύματος. Όσο αφορά το QoS η διαδρομή του μηνύματος από τον publisher στο subscriber μπορεί να εξεταστεί από 2 πλευρές. Κατά πρώτον από το Qos που αφορά την αποστολή μηνύματος από τον Publisher στον broker, όπως έχει καθοριστεί από την εντολή publish. Κατά δεύτερο από το Qos για την αποστολή μηνύματος από τον broker στο subscriber, όπως έχει καθοριστεί από την εντολή subscribe. Συνεπώς, το QoS μπορεί να υποβιβαστεί για κάποιο παραλήπτη ο οποίος έκανε subscribe με μικρότερο QoS. Στο MQTT υπάρχουν 3 QoS επίπεδα:

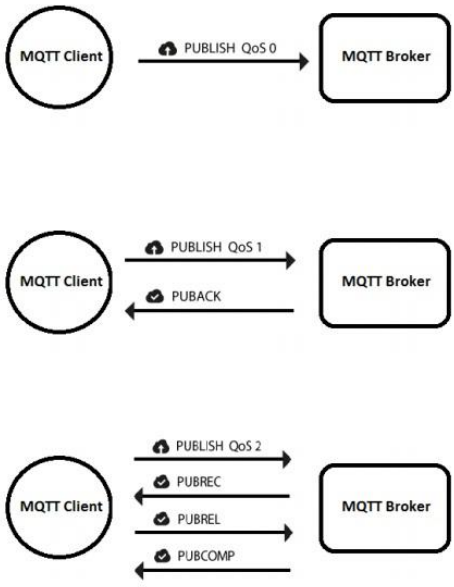
• **QoS 0** - Το πολύ μια φορά. Το ελάχιστο επίπεδο QoS, το μήνυμα αποστέλλεται μια φορά και δεν λαμβάνεται επιβεβαίωση παράδοσης ούτε αποθηκεύεται. Συνήθως είναι όσο αξιόπιστο είναι

και το TCP πρωτόκολλο πάνω από το οποίο γίνεται η αποστολή. Αυτό το επίπεδο επιλέγεται όταν υπάρχει σταθερή σύνδεση broker και client (συνήθως ενσύρματη) ή όταν δεν υπάρχει πρόβλημα να χάσουμε λίγη πληροφορία.

• **QoS 1** - Τουλάχιστον, μια φορά. Ο αποστολέας αποθηκεύει το μήνυμα μετά την αποστολή του μέχρι να πάρει το PUBACK μήνυμα. Αν περάσει συγκεκριμένο χρονικό διάστημα και δεν το λάβει προσπαθεί να αποστείλει το μήνυμα ξανά. Με αυτό το τρόπο το μήνυμα μπορεί να φτάσει περισσότερες φορές. Αυτό το επίπεδο επιλέγεται όταν στόχος είναι να φτάσει σίγουρα η πληροφορία και εναπόκειται στο χρήστη αν θα χειριστεί την επιπρόσθετη πληροφορία (τα αντίγραφα).

• **QoS 2** - Σίγουρα μια φορά. Είναι ο ασφαλέστερος, αλλά και ο πιο αργός τρόπος να σταλεί ένα μήνυμα. Για να συμβεί αυτό γίνονται 2 ανταλλαγές μηνυμάτων. Αν ο client στείλει δεύτερη φορά ένα μήνυμα, γιατί καθυστέρησε η απάντηση ο broker θα προωθήσει μόνο το ένα. Αυτό το επίπεδο επιλέγεται όταν είναι πολύ σημαντικό για την εφαρμογή να λαμβάνονται όλα τα μηνύματα ακριβώς μια φορά και η επιβάρυνση από το διπλό έλεγχο δεν μας είναι τόσο μειονέκτημα.

Το QoS είναι σημαντικό χαρακτηριστικό του πρωτοκόλλου MQTT. Με τον έλεγχο της παράδοσης του μηνύματος να είναι στα χέρια του MQTT μπορεί να κάνει την επικοινωνία σε αναξιόπιστα δίκτυα ευκολότερη. Επιπλέον δίνει την ευχέρεια στο χρήστη να διαλέξει το QoS επίπεδο ανάλογα με το είδος της εφαρμογής του. [13]



Εικόνα 33 : QoS(Quality of Service)

3.4.4. ΤΑ TOPICS

- Root topic
  "m2mce"
- Climate control
  "m2mce/hvac"
- Per main AC Unit
  "m2mce/hvac/ce" or "m2mce/lib" (for library building)
- Registration topics:

Registration_request_topic = "m2mce/hvac/registration"

Registration_request_topic_response = "m2mce/hvac/registration_response"

- Measurment topic:

HVAC_meas_topic = "m2mce/hvac/ce/meas"

- Command topics:

HVAC_cmd_topic = "m2mce/hvac/ce/act"

HVAC_cmd_ack_topic = "m2mce/hvac/ce/act_ack"

- Configuration/Maintenance topics:

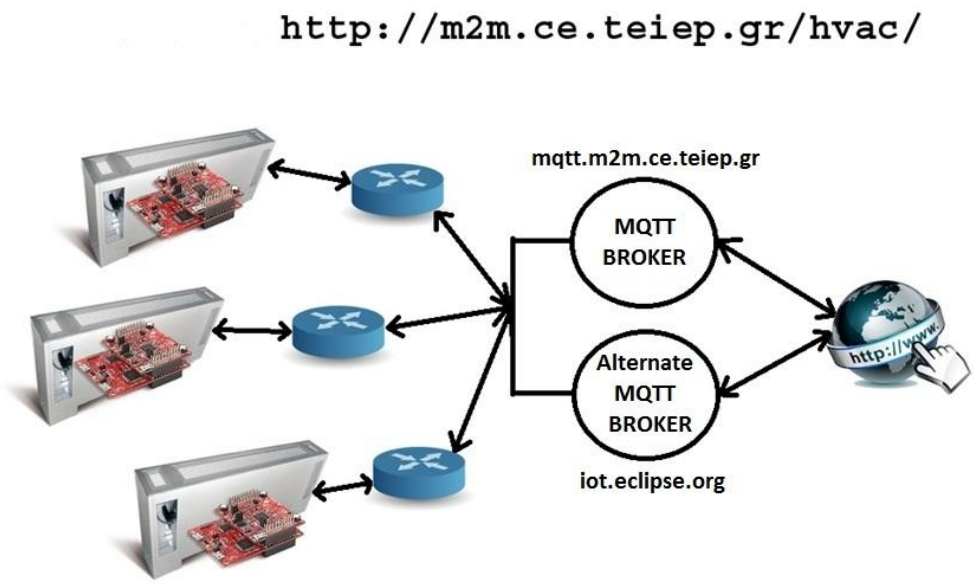
HVAC_conf_topic = "m2mce/hvac/conf"

HVAC_conf_ack_topic = "m2mce/hvac/conf/ack"

- ALARM topic

HVAC_alt_broker_topic = "m2mce/hvac/alarm"

Άρα η γενική ιδέα αυτού του κεφαλαίο ήταν ο προγραμματισμός του μικροελεγκτή και η επικοινωνία με τον mqtt broker, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 34 :Αρχιτεκτονική του project

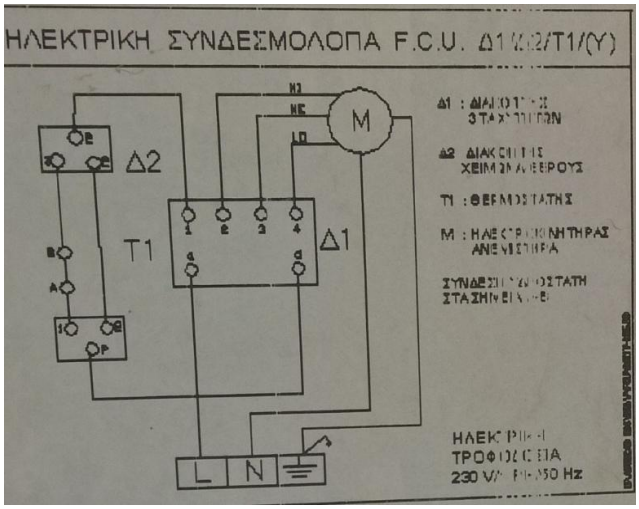
Η ιδέα του συστήματος είναι να εγκατασταθούν "έξυπνοι" κόμβοι στα fan coils του ΤΕΙ ΗΠΕΙΡΟΥ , να συνδέονται με το τοπικό wifi , να επικοινωνούν με τον mqtt broker και να ελέγχονται από την ιστοσελίδα. Ας πάμε να δούμε την εγκατάσταση των "έξυπνων" κόμβων στα fan coils.

ΚΕΦΑΛΑΙΟ 4

Σε αυτό το κεφάλαιο θα δούμε την εγκατάσταση των "έξυπνων" κόμβων στα fan coils. Θα μελετήσουμε την συνδεσμολογία του κατασκευαστή των fan coils ώστε να μπορέσουμε να συνδέσουμε τα relays και τους αισθητήρες ορθά.

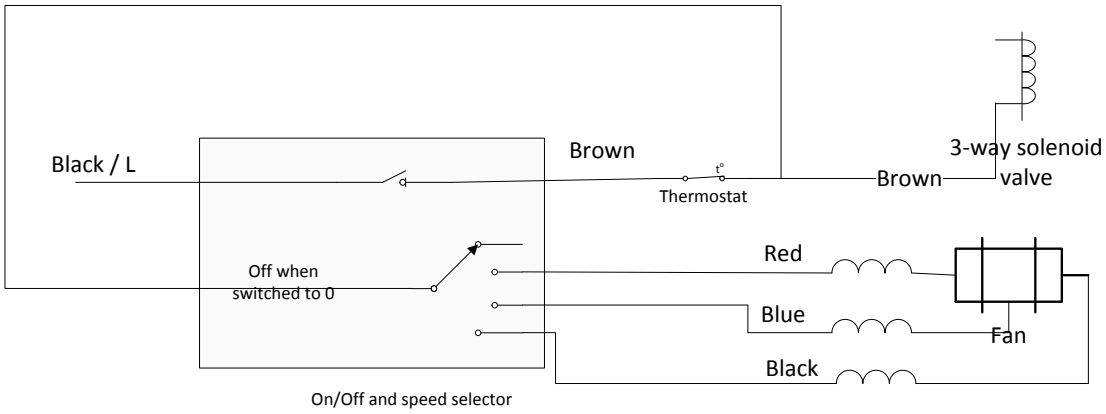
4.1 Η ΣΥΝΔΕΣΜΟΛΟΓΙΑ

Εργοστασιακή συνδεσμολογία fan coil unit



Εικόνα 35 : Εργοστασιακή συνδεσμολογία F.C.U

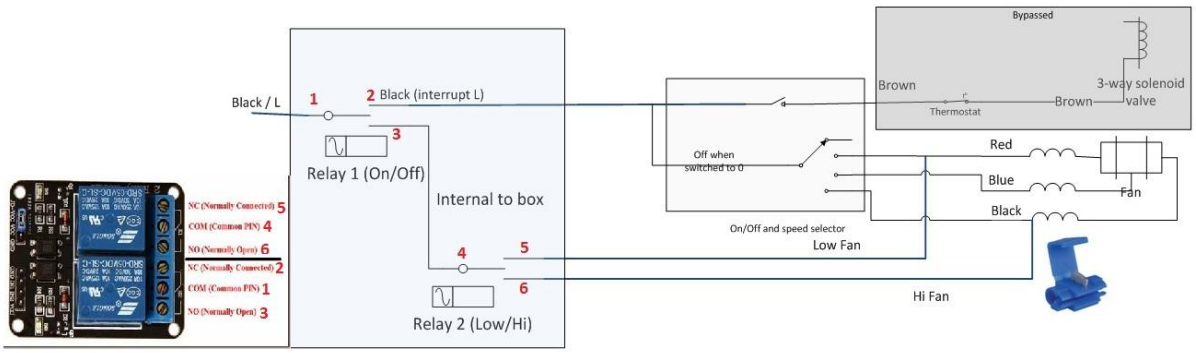
Εικόνα 36 : F.C.U εξαρτήματα



Εικόνα 37 : συνδεσμολογία F.C.U

Επιθυμητή συνδεσμολογία:

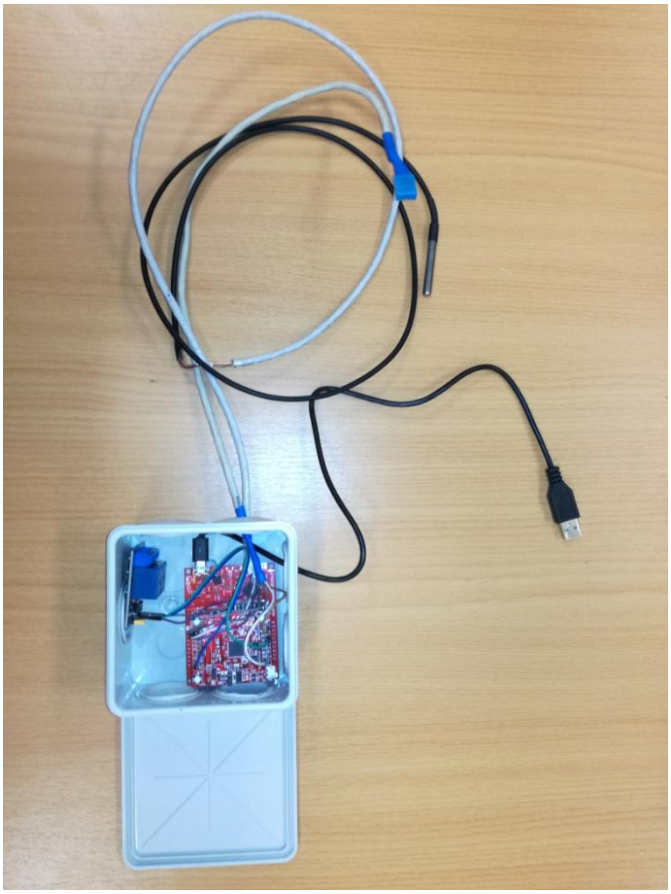
ΠΡΟΣΟΧΗ: Για να αποφευχθεί η πιθανότητα βραχυκυκλώματος των τυλιγμάτων του ανεμιστήρα από τυχών ταυτόχρονο τοπικό και απομακρυσμένο χειρισμό (μέσω του διακόπτη Δ1 και των relays αντίστοιχα), θα πρέπει η συνδεσμολογία να αποτρέπει την ταυτόχρονη ενεργοποίηση μέσω τοπικού χειρισμού. Αυτό επιτυγχάνεται με την κάτωθι συνδεσμολογία, που επιτρέπει χειροκίνητο χειρισμό μόνον όταν δεν είναι ενεργοποιημένη η συσκευή μέσω απομακρυσμένου ελέγχου.

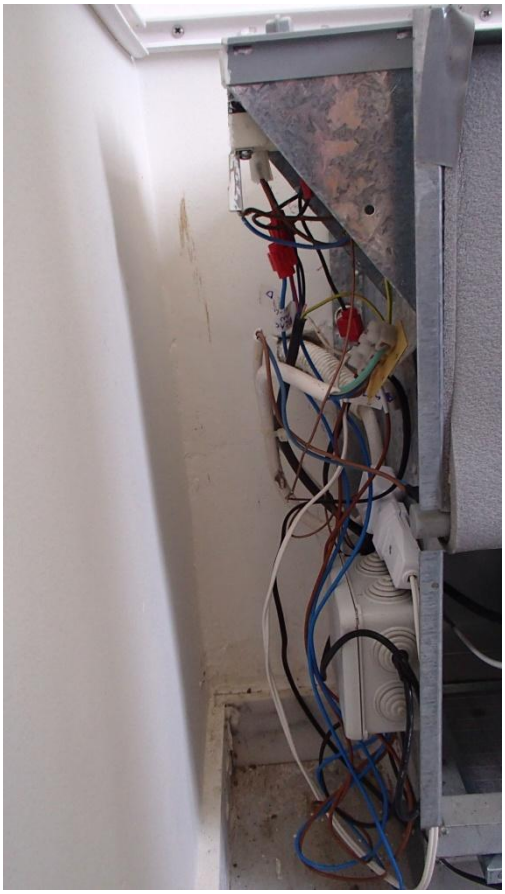


Εικόνα 38 : Επιθυμητή συνδεσμολογία F.C.U

Το κύκλωμα του υπάρχον θερμοστάτη και της ηλεκτροβάνας έχει καταργηθεί από τους ηλεκτρολόγους όπως φαίνεται στο σχήμα. Άρα ο K2 relay είναι για να ελέγχει το OnOff, ενώ ο K1 relay ελέγχει το HiLow του fan coil. Η επιθυμητή συνδεσμολογία φαίνεται παραπάνω. Επίσης χρησιμοποιήσαμε "κλέφτες" και κλέμες για να αποφύγουμε να κόψουμε τα υπάρχον καλώδια του fan coil.

4.2. ΠΑΡΑΘΕΣΗ ΦΩΤΟΓΡΑΦΙΩΝ





[1] https://en.wikipedia.org/wiki/Internet_of_things

[2] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, D. Boyle:
From Machine-to-Machine to the Internet of Things: Introduction to a
New Age of Intelligence. Elsevier, 2014, ISBN 978-0-12-407684-6

[3] https://el.wikipedia.org/wiki/Ασύρματο_δίκτυο_αισθητήρων

[4] https://en.wikipedia.org/wiki/Embedded_system

[5] https://www.eac.com.cy/EL/CustomerService/SavingEnergy/Pages/SavingEnergy.aspx

[6]https://en.wikipedia.org/wiki/HVAC

[7]John H. Davies: MSP430 Microcontroller Basics , ISBN-13: 978-0750682763

[8]Cem Unsalan : Programmable Microcontrollers with Applications: MSP430 LaunchPad with CCS and Grace , McGraw-Hill Education , ISBN-13: 978-0071830034

[9]http://energia.nu/

[10]http://pubsubclient.knolleary.net/api.html

[11]http://energia.nu/reference/libraries/

[12] http://processors.wiki.ti.com/index.php/CC31xx_Connection_Policy

[13]https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

A MSP430F5529 - Energia 17

Ακολουθεί ο πηγαίος κώδικας που χρησιμοποιήθηκε στον MSP430F5529

```
///// LIBRARIES /////
#include <SPI.h>          // This library allows you to communicate with
SPI devices, with the msp430 as the master device
#include <WiFi.h>         // Wifi connection library
#include <PubSubClient.h> // Mqtt protocol library
http://pubsubclient.knolleary.net/api.html
#include "DHT.h"          // DHT sensor library
#include <aJSON.h>        // header file for creating and parsing JSON
#include <OneWire.h>      // DS18B20 temperature sensor library
#include "MspFlash.h"     // Manipulate mspflash memory


///// Wifi & MQTT Connection /////
char MQTT_srv[] = "mqtt.m2m.ce.teiep.gr";  // Mqtt Broker URL
char MQTT_srv_alt[] = "iot.eclipse.org"; // Mqtt alternate Broker URL
int port = 1883;     // Port number
char username[] = "******";  // Username to connect to broker
char password[] = "******";   // Password to connect to broker

bool connected = false;        // true if connection is active (can be
false due to MQTT problems)
int  mqtt_failed_attempts=1;   // "0" indicates connected mqtt
#define max_mqtt_attempts_reboot 5      // After been connected, number
of retries before reset/reboot
WiFiClient wifiClient;   // the TCP client


//TOPICS//
char Registration_request_topic[] = "m2mce/hvac/registration";
char Registration_request_topic_response[] =
"m2mce/hvac/registration_response";
char HVAC_meas_topic[] = "m2mce/hvac/ce/meas";
char HVAC_cmd_topic[] = "m2mce/hvac/ce/act";
char HVAC_cmd_ack_topic[] = "m2mce/hvac/ce/act_ack";
char HVAC_conf_topic[] = "m2mce/hvac/conf";
char HVAC_conf_ack_topic[] = "m2mce/hvac/conf/ack";
char HVAC_alt_broker_topic[] = "m2mce/hvac/alarm";

///// declare callback() & mqtt client/////
void callback(char* topic, byte* payload, unsigned int length);   //
callback function protype - otherwise next statement generates "out of
scope" error
PubSubClient client(MQTT_srv, port, callback, wifiClient);        //
PubSubClient (server, port, [callback], client, [stream])
PubSubClient client_alt(MQTT_srv_alt, port, callback, wifiClient);    //
PubSubClient (alternate server, port, [callback], client, [stream])
```

```
long rssi = 0; // WIFI rssi(Received signal strength indication)
               // Attention!!! In order for rssi to work, it needs the
commit from here:
https://github.com/energia/Energia/commit/24b1824f7f3da7dec01dbedacff81b
79876c880e
               // to the int32_t WiFiClass::RSSI() in the WiFi.cpp in
the WiFi library!!! (copy/paste works perfect!!)


///// MSPFlash /////
//Use SEGMENT_B, SEGMENT_C or SEGMENT_D (each 64 bytes, 192 bytes in
total)
#define flash SEGMENT_D
int pos = 0; // position of segment in memory


///// MAC Address & Node_ID & Node_No/////
byte mac[10];                          // byte array for storing the mac
address
char Node_ID[10];                      // global variable for storing the
mac address
int Node_No = -1;                      // global variable for storing the
Node Number instead of mac address NodeNo=-1 --> Unregistered Node


///// JSON message object place holder/////
aJsonObject* sendJson;               // global variable for the root
aJSON object to send to the server
aJsonObject* recvJson;               // global variable for the root
aJSON object received from server
char* sendstring;                    // the string to hold the JSON
object to send to server
char* recvstring;                    // the string to hold the JSON
object to received from server


///// DS18B20 liquid temp sensor variables (according to GFDS18B20
library) /////
OneWire  ds(2);  // on pin P6_5 or 2  ds18b20 sensor (a 4.7K resistor is
necessary)
float HVAC_fluid_temp;          // global variable to store the
temperature aquired from DS18B20
byte addr[8];


///// DHT22 sensor variables (according to DHT library) /////
// Uncomment whatever type you're using!
#define DHTTYPE DHT11           // DHT 11
//#define DHTTYPE DHT22         // DHT 22  (AM2302)
//#define DHTTYPE DHT21         // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +3.3V or +5V
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the
sensor

#define DHTPIN P3_7             // the data pin of the DHT connected to
```

```
DHT dht(DHTPIN, DHTTYPE);      // Initialize the DHT driver, currently
on DHTPIN and DHTTYPE


///// RELAY control variables/////
#define HVAC_Unit_OnOff_PIN P3_6             // the pin controling the
On/Off state of the fancoil
#define HVAC_Unit_fan_PIN P7_4               // the pin controling the
Hi/Low state of the fancoil
boolean HVAC_Unit_OnOff = true;              // global variable to store
the state of the pin  (true -> OFF , false -> ON)
boolean HVAC_Unit_fan = true;                // global variable to store
the state of the pin  (true -> LOW , false -> HIGH)


///// GLOBAL VARIABLES /////
int msgCount=0;                  // to count the messages we send
#define LEDR RED_LED             // define red led as LEDR
#define LEDG GREEN_LED           // define green led as LEDG
bool time4Meas = true;           // global boolean variable changing
according to Minutes4Update
#define Minutes4Update 60000     // 120000 milliseconds = 2 minutes,
60000=60 sec to wait before sending the measurment message to the MQTT
broker
unsigned long elapsedtime;       // holds the last time a message send
to the MQTT broker
boolean Hard_Reset=true;         //true -> make hard_reset(erase flash
memory)
boolean Reset_simple=true;       //true -> make simple_reset(watchdog=0)
int internaltemp=0;             // save the msp430 internal temperature
static const uint16_t *ADC15TEMP30 = (uint16_t *)0x1A1A; //the
calibration values using the 1.5V reference are located at 0x1A1A for
the 30c
static const uint16_t *ADC15TEMP85 = (uint16_t *)0x1A1C; //the
calibration values using the 1.5V reference are located at 0x1A1C for
the 85c



///// INIT SETUP /////
void setup()
{


Serial.begin(9600); // Baud Rate for serial monitor
pinMode(LEDR, OUTPUT);         // configure Red Led as output
pinMode(LEDG, OUTPUT);         // configure Green Led as output
digitalWrite(LEDR, HIGH);      // TurnOn Red Led to indicate
configuration process
digitalWrite(LEDG, HIGH);      // TurnOn Green Led to indicate
configuration process
pinMode(HVAC_Unit_OnOff_PIN, OUTPUT);        // configure
HVAC_Unit_OnOff_PIN as output
pinMode(HVAC_Unit_fan_PIN, OUTPUT);          // configure
HVAC_Unit_fan_PIN as output
digitalWrite(HVAC_Unit_OnOff_PIN, HIGH);     // safe mode set to HIGH
digitalWrite(HVAC_Unit_fan_PIN, HIGH);       // safe mode set to HIGH

        Serial.println("Starting WiFi SmartConfig"); // Print in serial
monitor
    WiFi.startSmartConfig(); //Enter SmartConfigMode
```

```
      while (WiFi.localIP() == INADDR_NONE) {
    // print dots while we wait for an ip addresss
    Serial.print(".");
    digitalWrite(LEDR, HIGH);            // turn on red led
    delay(300);
  }

    digitalWrite(LEDR, LOW);            // configuration done turn off
led
    digitalWrite(LEDG, LOW);            // configuration done turn off
led

     printWifiStatus();                        // we're connected now, so
print in serial console the connection status

     WiFi.macAddress(mac);             // read the mac address of the
device ! Needed here to run every time or else somehow overwritten
     sprintf(Node_ID, "%02X:%02X:%02X:%02X:%02X:%02X", mac[5], mac[4],
mac[3], mac[2], mac[1], mac[0]); // properly store in char array the mac
as string

           MQTTconnection();                                  // call
MQTTconnection function for connection to Mqtt Broker


    dht.begin();                              // initialize the DHT
temperature sensor
        ds.search(addr);                          // Read ds18b20 sensor
address
    elapsedtime = millis();                 // initialize the variable
with current miliseconds passed

}//END of setup() function



///// MAIN LOOP /////
void loop()
{

    if ( WiFi.status() == WL_CONNECTED ) {           // check if we have
connection with the AP
              if ( client.connected() ){
                mqtt_failed_attempts=0;                    //Resets the
mqtt fail counts. "0" indicates connected
        connected=true;
              }   // when mqtt connection lost
              else{
              delay(3000);                                  //slow down
the loop before attempting recconect
        mqtt_failed_attempts++;                     //increase the mqtt
fail count
          MQTTconnection();                        // call
MQTTconnection function for connection to Mqtt Broker

        if (mqtt_failed_attempts > max_mqtt_attempts_reboot) {  //Decide
to abort and reboot
                FAIL_SAFE();  // Turn off fan coil for safety
                  Alt_MQTT_Connection(); // Publish message to
alternate_Broker that primary MQTT_Broker is down
```

```
                Serial.println("Reseting...");
              WDTCTL = 0;                                          // zero
out the WatchDog Timer so the device restarts
                } // END - if (mqtt_failed_attempts >
max_mqtt_attempts_reboot)
              } // END - if ( client.connected() )

    }
    else{                                                  // when
connection with AP lost
             FAIL_SAFE();  // Turn off fan coil for safety
        Serial.println("Reseting...");
             WDTCTL = 0;                                   // zero out
the WatchDog Timer so the device restarts

        }    // END - if ( WiFi.status() == WL_CONNECTED)


    if (time4Meas && connected && Node_No!=-1){
//Determine if it is time to measure/publish

           HVAC_meas(); // Call HVAC_meas() function to publish
measurments
           time4Meas = false;                          // make the
update variable false
           elapsedtime = millis();                      // store the
current time a message sent

         }// End of time4Meas IF
    if ((millis() - elapsedtime) > Minutes4Update) time4Meas = true; //
check if MINS passed and change variable to true to send measurment
message to the MQTT broker

        client.poll(); // Check if any message were received on the
topic we subscribed to
} // End of loop() function

///// callback function calls from client.poll() /////
void callback(char* topic, byte* payload, unsigned int length) {

 /*
  Serial.println("Received message for topic ");
  Serial.print(topic);
  Serial.print(" with length ");
  Serial.println(length);
  Serial.println("Message:");
  Serial.write(payload, length);   // Writes binary data to the serial
port
*/

     char json[512];                    // char array
     memcpy(json, payload, 512);         // void * memcpy ( void *
destination, const void * source, size_t num );    Copy block of memory
to convert byte array into char array


  if ( strcmp(topic, Registration_request_topic_response) == 0 )
//check if we received a message in a specific topic
  {
     Serial.println("Registration_response message Received");
```

```
      recvJson = aJson.parse(json);           // create a JSON object
from the received message
      Registration_response();                // call
Registration_response() function
   }                                          //End-if ( strcmp(topic,
Registration_request_topic_response) == 0 )

   if ( strcmp(topic, HVAC_cmd_topic) == 0 )     // checks if a command
type message has been received
   {
      Serial.println("Act message Received");
      recvJson = aJson.parse(json);            // create a JSON object
from the received message
      HVAC_cmd();                              //call HVAC_cmd() function
   }                                           //End-if ( strcmp(topic,
HVAC_cmd_topic) == 0 )

   if ( strcmp(topic, HVAC_conf_topic) == 0 )  // checks if a
configuration message has been received
   {
     Serial.println("Reset message received");
     recvJson = aJson.parse(json);               // create a JSON object
from the received message
     HVAC_conf();                              // call HVAC_conf()
function
   }                                           //End-if ( strcmp(topic,
HVAC_conf_topic) == 0 )

   memset(json,0,sizeof json); //memset(array,0,sizeof(array)); claer
json array

 } // End of callback() function

///// Function about Read NodeNo from mspflash memory /////
void doRead()
{

   int i=0;
   //Serial.println(Node_No);
   //Serial.println(Node_No, HEX);
   //Serial.println(sizeof(int));
   //Serial.println(".");
   //Serial.println("Read:");
   Flash.read(flash+(pos * sizeof(int)), (unsigned char*)&Node_No,
sizeof(int));
   //Serial.println(Node_No);
   //Serial.println(Node_No, HEX);
   //Serial.println(".");
} //End of doRead() function

///// Function about Write NodeNo to mspflash memory /////
void doWrite()
{
 Serial.println("Write");
 Flash.write(flash + (pos * sizeof(int)), (unsigned char*)&Node_No,
sizeof(int));
 Serial.println("Done.");
} //End of doWrite() function

///// Function to Erase mspflash memory /////
void doErase()
```

```
{
 Serial.println("Erase");
 Flash.erase(flash);       // Erase msp flash memory
 Serial.println("Done.");
} //End of doErase() function

///// Reset nodes function /////
void HVAC_conf()
{
   if (recvJson != NULL) {                                    // if the
message is valid JSON
    aJsonObject* NodeNo = aJson.getObjectItem(recvJson, "NodeNo"); //
read the NodeNo value

     if (NodeNo != NULL)    {              // if NodeNo is in the message
       if(NodeNo->valueint == Node_No){    // if NodeNo is for this node
         aJsonObject* HardReset = aJson.getObjectItem(recvJson,
"HardReset");     // Read HardReset value from server
            Hard_Reset=HardReset->valuebool;
         aJsonObject* Reset = aJson.getObjectItem(recvJson, "Reset");
// Read Reset value from server
            Reset_simple=Reset->valuebool;
         if(HardReset!=NULL && Hard_Reset)
         {
            HVAC_conf_ack(); // send ack message to broker
            doErase(); // Erase mspflash memory
            WDTCTL = 0; // zero out the WatchDog Timer so the device
restarts
         }  //End-if (HardReset!=NULL && Hard_Reset)

         if(Reset!=NULL && Reset_simple)
         {
            HVAC_conf_ack(); // send ack message to broker
            WDTCTL = 0; // zero out the WatchDog Timer so the device
restarts
         }  //End-if (Reset!=NULL && Reset_simple)

      }                                       //End-if (NodeNo->valueint ==
Node_No)
     }                                        //End-if (NodeNo != NULL)
  }                                           //End-if (recvJson != NULL)
   free(recvJson);                                          // free
the JSON string
  aJson.deleteItem(recvJson);                               // delete
the JSON Object

}// End of HVAC_conf() function

///// Reset nodes function ack/////
void HVAC_conf_ack()
{
  sendJson = aJson.createObject();                // Creating the root
JSON Object
  if (sendJson == NULL)
    {                          // if object = NULL
    Serial.print("JSON Object not created.");     // print in serial
console JSON Object not created.
    }                                               // END - if
(sendJson == NULL)
  aJson.addNumberToObject(sendJson, "NodeNo", Node_No); // Add the node
number
```

```
  aJson.addBooleanToObject(sendJson, "HardReset", Hard_Reset); // Add
the hardreset
  aJson.addBooleanToObject(sendJson, "Reset", Reset_simple); // Add the
reset

   ////publish Registration_request message /////
  sendstring = aJson.print(sendJson);             // save the created
JSON object as a string
  client.publish(HVAC_conf_ack_topic, sendstring);           // int
publish (topic, payload)
        Serial.print("Sent: ");                          // print in
serial console the message Sent:
        Serial.println(sendstring);                      // print in
serial console the JSON string we sent
        free(sendstring);                                // release the
variable for JSON string
        aJson.deleteItem(sendJson);                      // delete the
JSON object
}// End of HVAC_conf_ack() function


///// MQTTconnection() function /////
void MQTTconnection() // Function about connecting to Mqtt Broker
{

    if (!client.connected()) // Reconnect if the connection was lost
    {
      if(!client.connect(Node_ID, username, password)) //  boolean
connect (clientID, username, password)
      {
        Serial.println("Connection to MQTT Broker failed ! Check
ClientID , Name or Password");
      }
      else
      {
      Serial.println("Connection success to MQTT Broker");

         doRead(); // Read NodeNo from mspflash memory
         // NodeNo = -1  Unregistered Node
      if (Node_No == -1){
         Registration_request();          // call Registration_request
function to create json message
     Serial.println("Waiting for response to registration message");

      }

       if(client.subscribe(Registration_request_topic_response))  //
Check if someone subscribe in Topic
      {
        //Serial.println("Subscription successfull");
      } // End of subscribe() function
       if(client.subscribe(HVAC_cmd_topic))  // Check if someone
subscribe in Topic
      {
        //Serial.println("Subscription successfull");
      } // End of subscribe() function
      if(client.subscribe(HVAC_conf_topic))  // Check if someone
subscribe in Topic
      {
        //Serial.println("Subscription successfull");
      } // End of subscribe() function
```

```
     } // End of client.connect() function
    } // End of client.connected() function
} // End of MQTTconnection() function


///// Alternate MQTT_connection function /////
void Alt_MQTT_Connection(){
          if (!client_alt.connected()) // Reconnect if the connection
was lost
             {
               if(!client_alt.connect(Node_ID, username, password)) //
boolean connect (clientID, username, password)
               {
                 Serial.println("Connection to altenate MQTT Broker
failed ! Check ClientID , Name or Password");
               }
               else
               {
               Serial.println("Connection success to alternate MQTT
Broker");
               sendJson = aJson.createObject();                //
Creating the root JSON Object
               if (sendJson == NULL)
                 {                          // if object = NULL
                 Serial.print("JSON Object not created.");     // print
in serial console JSON Object not created.
                 }                                               // END
- if (sendJson == NULL)
                  doRead(); // Read NodeNo from mspflash memory
                  aJson.addNumberToObject(sendJson, "NodeNo", Node_No);
// Add the node number
                  aJson.addStringToObject(sendJson, "ALARM",
"Connection with Primary MQTT Broker Lost");
                  sendstring = aJson.print(sendJson);             //
save the created JSON object as a string
               client_alt.publish(HVAC_alt_broker_topic, sendstring);
// int publish (topic, payload)
               Serial.print("Sent: ");                          // print
in serial console the message Sent:
               Serial.println(sendstring);                     // print
in serial console the JSON string we sent
               free(sendstring);                               //
release the variable for JSON string
               aJson.deleteItem(sendJson);                     //
delete the JSON object
               }// END - if(!client.connect(clientID, username,
password))
             }// END - if (!client.connected())
}// END of Alt_MQTT_Connection() function


///// Registration_request() function /////
void Registration_request(){
sendJson = aJson.createObject();                // Creating the root
JSON Object
  if (sendJson == NULL)
    {                          // if object = NULL
    Serial.print("JSON Object not created.");     // print in serial
console JSON Object not created.
    }                                               // END - if
(sendJson == NULL)
```

```
  aJson.addStringToObject(sendJson, "NodeID", Node_ID); // Add the mac
address
  aJson.addStringToObject(sendJson, "Ver", WiFi.firmwareVersion()); //
Add the Version of CC3100 wifi BOOSTERPACK
  aJson.addStringToObject(sendJson, "Type", "hvac_comb"); // Add the
Type of the device (hvac_meas,hvac_act,hvac_comb,hvac_fc,hvac_IR,other)
  aJson.addStringToObject(sendJson, "HW_Ver", "MSP430F5529 REV1.6 +
CC3100 REV4.0"); // Add the Harware Version we use

  ////publish Registration_request message /////
  sendstring = aJson.print(sendJson);             // save the created
JSON object as a string
  client.publish(Registration_request_topic, sendstring);           //
int publish (topic, payload)
        Serial.print("Sent: ");                          // print in
serial console the message Sent:
        Serial.println(sendstring);                      // print in
serial console the JSON string we sent
        free(sendstring);                                // release the
variable for JSON string
        aJson.deleteItem(sendJson);                      // delete the
JSON object
}//End of Registration_request() function


///// Registration_response() function /////
void Registration_response(){
  if (recvJson != NULL) {                                  // if the
message is valid JSON
    aJsonObject* NodeID = aJson.getObjectItem(recvJson, "NodeID"); //
read the NodeID value

     if (NodeID != NULL)    {              // if NodeID is in the message
     char* MAC=NodeID->valuestring;        // store mac value
      if ( strcmp(MAC, Node_ID) == 0 ) {   // compare mac from message
with device mac
         aJsonObject* NodeNo = aJson.getObjectItem(recvJson, "NodeNo");
// Read NodeNo value from server
         Node_No=NodeNo->valueint;         // store NodeNumber value
            doErase(); // Erase mspflash memory
            doWrite(); // Write NodeNo to mspflash memory
            WDTCTL = 0; // zero out the WatchDog Timer so the device
restarts

      }                                     //End-if ( strcmp(MAC,
Node_ID == 0 )
     }                                      //End-if (NodeID != NULL)
  }                                         //End-if (recvJson != NULL)
   free(recvJson);                                          // free
the JSON string
  aJson.deleteItem(recvJson);                                // delete
the JSON Object

}//END of Registration_response() function


///// HVAC_meas() function /////
void HVAC_meas()
{
sendJson = aJson.createObject();                // Creating the root
JSON Object to be filled with the measurements
```

```
  if (sendJson == NULL)
    {                           // if object = NULL
    Serial.print("JSON Object not created.");     // print in serial
console JSON Object not created.
    }                                       // END - if (sendJson ==
NULL)
  aJson.addNumberToObject(sendJson, "NodeNo", Node_No); // Add the
NodeNo (Node Number)
                digitalWrite(LEDG, LOW); // turn off green led
                digitalWrite(LEDR, LOW); // turn off red led
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
sensor)
  int HVAC_hum = dht.readHumidity();                              // read
and store the humidity from the sensor to variable - ATTENTION!! Should
it be float according to the example in the lib?
  int HVAC_temp = dht.readTemperature();                          // read
and store the temperature from the sensor to variable - ATTENTION!!
Should it be float according to the example in the lib?
  delay(10); // delay to take time for dht temperature reading

  if ((HVAC_temp == 0) & (HVAC_hum == 0) ) {  /* t =0 AND h = 0 out of
specs so error in reading (isnan(t) || isnan(h)) { // check if returns
are valid, if they are NaN (not a number) then something went wrong! */
    Serial.println("Failed to read from DHT");           // print in
serial console Failed to read from DHT
    aJson.addNumberToObject(sendJson, "Humidity", -100); // add to the
JSON object -100 to indicate error in reading
    aJson.addNumberToObject(sendJson, "Temperature", -100); // add to
the JSON object -100 to indicate error in reading
    digitalWrite(LEDR, HIGH);     // TurnOn Red Led
  } else {                                                // else if
returns are valid
    aJson.addNumberToObject(sendJson, "Humidity", HVAC_hum);    // add
to the JSON object the value from the sensor
    aJson.addNumberToObject(sendJson, "Temperature", HVAC_temp); // add
to the JSON object the value from the sensor
    digitalWrite(LEDG, HIGH);     // TurnOn Green Led
  }     // END - if ((HVAC_temp == 0) & (HVAC_hum == 0) )

 readOWsensor(); // Reading temperature from DS18B20 sensor function and
assign it to the global variable HVAC_fluid_temp
 if (HVAC_fluid_temp == 0) // if HVAC_fluid_temp == 0 error reading
 {
    aJson.addNumberToObject(sendJson, "CoolantTemp", -100);// add the
fluid temperature to the JSON object
    if (digitalRead(LEDG))   // if green led is on
    {
      digitalWrite(LEDG, LOW);  // Turn off green led
      digitalWrite(LEDR, HIGH); // Turn on red led
    }
     Serial.println("Failed to read from ds18b20");
 } else {
    aJson.addNumberToObject(sendJson, "CoolantTemp", HVAC_fluid_temp);//
add the fluid temperature to the JSON object

 } // END - if (HVAC_fluid_temp == 0)

  HVAC_Unit_OnOff = digitalRead(HVAC_Unit_OnOff_PIN); // check the state
of the pin and assign the value to HVAC_Unit_OnOff variable
```

```
  aJson.addBooleanToObject(sendJson, "OnOff", HVAC_Unit_OnOff);// add
the onoff status to the JSON object
  HVAC_Unit_fan = digitalRead(HVAC_Unit_fan_PIN); // check the state of
the pin and assign the value to HVAC_Unit_fan variable
  aJson.addBooleanToObject(sendJson, "HiLow", HVAC_Unit_fan);// add the
hilow status to the JSON object
  internaltemp=GetTempX10(2); // Now, the internaltemp is a 16 bit value
that maps i.e. 301 -> 30.1 Celcius, averaging 4 readings
  aJson.addNumberToObject(sendJson, "Msp_Temp", internaltemp); // add
the msp430 internal temperature to the JSON object
  rssi = WiFi.RSSI();                  // gets the received signal
strength
  aJson.addNumberToObject(sendJson, "RSSI", (int) rssi);// add the
receiver signal strength to the JSON object
  aJson.addNumberToObject(sendJson, "msgCount", msgCount++); // count
messages and add to the JSON object
  //publish HVAC_meas message
  sendstring = aJson.print(sendJson);              // save the created
JSON object as a string
  client.publish(HVAC_meas_topic, sendstring);           // int publish
(topic, payload)
        Serial.print("Sent: ");                          // print in
serial console the message Sent:
        Serial.println(sendstring);                      // print in
serial console the JSON string we sent
        free(sendstring);                                // release the
variable for JSON string
        aJson.deleteItem(sendJson);                      // delete the
JSON object
}// End of HVAC_meas() function


///// HVAC_cmd() function /////
void HVAC_cmd(){
    if (recvJson != NULL) {                                  // if the
message is valid JSON
    aJsonObject* NodeNo = aJson.getObjectItem(recvJson, "NodeNo"); //
check the NodeNo
     if (NodeNo != NULL)    {               // if NodeNo is in the message
      if (NodeNo->valueint == Node_No ){  // if NodeNo from message is
the same with device NodeNo
       aJsonObject* on_off = aJson.getObjectItem(recvJson, "OnOff");  //
Read OnOff value from server
       aJsonObject* hi_lo = aJson.getObjectItem(recvJson, "HiLow"); //
Read HiLow value from server
       if (on_off != NULL) {                                    // if
on_off value exists in message
        //Serial.println(on_off->valuebool);                    //
print it to the serial console
        if (on_off->valuebool) {                                // if
on_off = true
          digitalWrite(HVAC_Unit_OnOff_PIN, HIGH);
// set pin to HIGH
        } else {                                              // else if
on_off = false
          digitalWrite(HVAC_Unit_OnOff_PIN, LOW);
// set pin to LOW
        }                                                     // END -
if (on_off->valuebool)
      }                                                       // END -
if (on_off != NULL)
```

```
      if (hi_lo != NULL) {                                          // if
hi_lo value exists in message
        //Serial.println(hi_lo->valuebool);                         // print
it to serial console
        if (hi_lo->valuebool) {                                     // if
hi_lo = true
          digitalWrite(HVAC_Unit_fan_PIN, HIGH);
// set pin to HIGH
        } else {                                                  // else if
hi_lo = false
          digitalWrite(HVAC_Unit_fan_PIN, LOW);
// set pin to LOW
        }                                                         // END -
if (hi_lo->valuebool)
      }      // END - if (hi_lo != NULL)
      aJsonObject* msgCnt = aJson.getObjectItem(recvJson, "msgCount");
// store to msgCnt the value received from server
       HVAC_cmd_response();// Call  HVAC_cmd_response() function

      }else{
        Serial.print("\n Act message for other device");
      }//End - if  MAC == Node_ID
     }// End - if NodeID != NULL
   }                                                               // END -
if (recvJson != NULL)
  free(recvJson);                                                 // free
the JSON string
  aJson.deleteItem(recvJson);                                     // delete
the JSON Object

}// End of HVAC_cmd() function


///// HVAC_cmd_response() function /////
void HVAC_cmd_response(){
  sendJson = aJson.createObject();                // Creating the root
JSON Object
  if (sendJson == NULL)
    {                             // if object = NULL
    Serial.print("JSON Object not created.");     // print in serial
console JSON Object not created.
    }                                               // END - if
(sendJson == NULL)
    aJson.addNumberToObject(sendJson, "NodeNo", Node_No); // Add the
NodeNo to the JSON object
    HVAC_Unit_OnOff = digitalRead(HVAC_Unit_OnOff_PIN); // check the
state of the pin and assign the value to HVAC_Unit_OnOff variable
    aJson.addBooleanToObject(sendJson, "OnOff", HVAC_Unit_OnOff);//  add
the OnOff to JSON object
       HVAC_Unit_fan = digitalRead(HVAC_Unit_fan_PIN); // check the
state of the pin and assign the value to HVAC_Unit_fan variable
       aJson.addBooleanToObject(sendJson, "HiLow", HVAC_Unit_fan);// add
the HiLow to JSON object
       aJson.addBooleanToObject(sendJson, "LEDR", digitalRead(LEDR));//
add the LEDR to JSON object
       aJson.addBooleanToObject(sendJson, "LEDG", digitalRead(LEDG));//
add the LEDG to JSON object
       aJson.addNumberToObject(sendJson, "msgCount", msgCount); // count
messages and add to the JSON object
       //publish HVAC_cmd_response message
```

```
        sendstring = aJson.print(sendJson);             // save the
created JSON object as a string
    client.publish(HVAC_cmd_ack_topic, sendstring);          // int
publish (topic, payload)
        Serial.print("Sent: ");                         // print in
serial console the message Sent:
        Serial.println(sendstring);                     // print in
serial console the JSON string we sent
        free(sendstring);                               // release the
variable for JSON string
        aJson.deleteItem(sendJson);                     // delete the
JSON object
}// End of HVAC_cmd_response function

///// Fail Safe function if connection lost then turn off fan coil /////
void FAIL_SAFE() {
  digitalWrite(LEDR, HIGH);            // turn on red led
  digitalWrite(HVAC_Unit_OnOff_PIN, HIGH);     // failsafe mode set to
HIGH
  digitalWrite(HVAC_Unit_fan_PIN, HIGH);       // failsafe mode set to
HIGH
  Serial.println("Enter Fail Safe Mode");
}// End of FAIL_SAFE function


void printWifiStatus()
{                   // function to print the SSID of the network we're
attached to
  Serial.print("Network Name: ");          // print the Network Name: in
serial console
  Serial.println(WiFi.SSID());             // print the SSID of the
network you're attached to

  IPAddress ip = WiFi.localIP();           // gets the WiFi shield's IP
address
  Serial.print("IP Address: ");            // print IP Address: in
serial monitor
  Serial.println(ip);                      // prints the shield's IP
address in serial monitor

  rssi = WiFi.RSSI();                  // gets the received signal
strength
  Serial.print("signal strength (RSSI):"); // print the signal strength
(RSSI): in serial monitor
  Serial.print(rssi);                      // print the received signal
strength in serial monitor
  Serial.println(" dBm");                  // print the dBm in serial
console
}                                          // END - void
printWifiStatus()

///// msp430 internal temperature /////
int16_t GetTempX10(int oversampling)
{
  uint32_t reading=0;

  analogReference(INTERNAL1V5); //use the 1.5V internal reference

  for(int i=0; i<(1<<oversampling); i++){
    reading+=analogRead(TEMPSENSOR); //do 2^oversampling reads
    delay(2);  //wait 2 ms between successive reads
```

```
    }
  reading>>=oversampling; //and divide by 2^oversampling, we are left
with the average from 2^oversampling reads.

  //Map converts a number from an input range (our calibration
constants) to an output range (our temperature)
  return map(reading,*ADC15TEMP30,*ADC15TEMP85,300,850); //convert our
reading to temp x 10 & return
}


///// This function reads ds18b20 fluid temperature /////
void readOWsensor()
{
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];

  /*
  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }
  */
  ds.search(addr);
  //Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    //Serial.write(' ');
    //Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      //Serial.println("CRC is not valid!");
      return;
  }
  Serial.println();

  // the first ROM byte indicates which chip
  if(addr[0]==0x28) {
      //Serial.println("  Chip = DS18B20");
      type_s = 0;
  }else{
      //Serial.println("Device is not DS18B20.");
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);        // start conversion, with parasite power on
at the end

  delay(1000);     // maybe 93.75 ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of
it.

  present = ds.reset();
  ds.select(addr);
```

```
  ds.write(0xBE);         // Read Scratchpad

  //Serial.print("  Data = ");
  //Serial.print(present, HEX);
  //Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    //Serial.print(data[i], HEX);
    //Serial.print(" ");
  }
  //Serial.print(" CRC=");
  //Serial.print(OneWire::crc8(data, 8), HEX);
  //Serial.println();

  // Convert the data to actual temperature
  // because the result is a 16 bit signed integer, it should
  // be stored to an "int16_t" type, which is always 16 bits
  // even when compiled on a 32 bit processor.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  HVAC_fluid_temp = (float)raw / 16.0;
  //Serial.print("  Temperature = ");
  //Serial.print(HVAC_fluid_temp);
  //Serial.print(" Celsius, ");


}// End of readOWsensor()
```