ΤΕΙ ΗΠΕΙΡΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΣΥΡΜΑΤΟ ΣΥΣΤΗΜΑ ΠΕΡΙΒΑΝΤΟΛΟΓΙΚΟΥ ΕΛΕΓΧΟΥ ΚΤΗΡΙΩΝ

Λισγάρας Βασίλειος (10812)

Επιβλέπων καθηγητής: Γρηγόρης Δουμένης

Άρτα 2016

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τους συμφοιτητές μου Μασκλαβάνο Γιάννη και Πανταζή Γρηγόρη που συμβάλαμε όλοι μαζί για τη δημιουργία του κοινού projectπου μας ανατέθηκε και για την πολύ καλή συνεργασία που υπήρξε μεταξύ μας και μας οδήγησε στην επίλυση των προβλημάτων που τελικά αντιμετωπίσαμε επιτυχώς. Επίσης τον καθηγητή μας Κύριο Δουμένη Γρηγόριο που μας παρείχε υποστήριξη και καθοδήγηση βοηθώντας μας να ξεπεράσουμε τις οποιεσδήποτε δυσκολίες παρουσιάστηκαν κατά τη διάρκεια εκπόνησης της παρούσας εργασίας.

Περίληψη

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι να μελετηθεί και να αναπτυχθεί ένας ασύρματος αισθητήριος κόμβος, μέσω του οποίου θα ελέγχεται η λειτουργικότητα κλιματιστικών μονάδων εξ αποστάσεως, από το διαδίκτυο ή από το τοπικό ασύρματο δίκτυο. Με αυτόν τον τρόπο επιδιώκεται η εξοικονόμηση ενέργειας και ο περιορισμός των εκπομπών διοξειδίου του άνθρακα, καθώς περιορίζεται η σπατάλη πόρων διότι ο έλεγχος των συσκευών γίνεται από απόσταση και περιορίζεται η χρήση τους στο απαραίτητο.

Αρχικά, γίνεται παρουσίαση ενός συνδεδεμένου κόμβου με το server και στη συνέχεια αναλύονται τα υλικά από τα οποία αποτελείται ο αισθητήριος κόμβος, όπως οι αισθητήρες θερμοκρασίας και υγρασίας, ο ηλεκτρομαγνητικός διακόπτης για τη διαχείριση της κλιματιστικής μονάδας και οι επιμέρους πλακέτες, καθώς και η διαδικασία προγραμματισμού τους. Στη συνέχεια, αναλύεται εκτενώς ο πηγαίος κώδικας του προγράμματος διαχείρισης του αισθητήριου κόμβου και της επικοινωνίας του με το server. Τέλος παρουσιάζεται το εναλλακτικό υλικό και λογισμικό όπως και προτάσεις για μελλοντικά project.

Abstract

The aim of the present dissertation is to study and develop a wireless sensor node to remotely control the functionality of HVAC systems over the internet. In this way, it is aimed to achieve energy efficiency in CO2 emission, since the waste of resources is reduced by controlling remotely the HVAC systems and it is used only when necessary.

First, presented the architecture design with the microcontroller/node and a server. After that the Bill of Materials (BOM) of the sensor node are presented, such as the temperature and humidity sensors, the controller relay and the microcontroller LaunchPads, as also the process of updating their firmware. Additionally, the source code of the sensor node and the communication with the server is analyzed in depth. Last, other suggested hardware is proposed, as also future project applications.

Κεφάλαιο 1: Εισαγωγή

Στο πρώτο κεφάλαιο κάνουμε εισαγωγή στις έννοιες του Internet of Things, Machine to Machine και των Wireless sensor networks.

Κεφάλαιο 2: Εφαρμογές και πράσινη διαχείριση

Στο δεύτερο κεφάλαιο γίνεται αναφορά στις έρευνες που έχουν γίνει για την εξοικονόμηση πόρων και ενέργειας λόγω των έξυπνων κλιματιστικών μονάδων και πως συμβάλει στην πράσινη διαχείριση

Κεφάλαιο 3: Προγραμματισμός μικροελεγκτή

Στο τρίτο κεφάλαιο θα δημιουργήσουμε από την αρχή βήμα προς βήμα έναν ασύρματο κόμβο για τη διαχείριση κλιματιστικών μονάδων.

Κεφάλαιο 4: Παρουσίαση της λειτουργίας του μικροελεγκτή

Στο τέταρτο κεφάλαιο γίνεται ανάλυση του πηγαίου κώδικα και παρουσίαση της λειτουργίας του μικροελεγκτή.

Κεφάλαιο 5: Προτάσεις και ιδέες

Στο πέμπτο κεφάλαιο προτείνονται ιδέες για μελλοντικά project που είναι βασισμένα στο συγκεκριμένο μικροελεγκτή και παρουσιάζονται και άλλα LaunchPad και μικροελεγκτές άλλων εταιριών που κατασκευάστηκαν με στόχο τη συνδεσιμότητα και το Internet of Things.

Περιεχόμενα

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ 6

Κεφάλαιο 1 – Εισαγωγή 7

1. Εισαγωγή 8

1.1. IoT – Internet of Things 8

1.2. M2M 11

1.3. Wireless Sensor Networks 12

Κεφάλαιο 2 – Εφαρμογές και πράσινη διαχείριση 15

2. Εφαρμογές και πράσινη διαχείριση 16

2.1. Smart HVAC Controls 16

Κεφάλαιο 3 – BOM και συνδεσμολογία 19

3. Απαιτούμενο Υλικό 20

3.1. Συνδεσμολογία 20

3.2. Διαδικασία flash image 23

Κεφάλαιο 4 – Προγραμματισμός μικροελεγκτή 24

4. Περιβάλλον ανάπτυξης Energia 25

4.1. Project 26

4.2. Project Code 29

4.2.1. Header Files, Libraries, Global Variables 29

4.2.2. Setup function 29

4.2.3. Loop function 31

Κεφάλαιο 5 – Προτάσεις και ιδέες 38

5. Αποτέλεσμα 39

5.1 Εναλλακτικό Hardware 39

5.2 Ιδέες για μελλοντικά project 40

Βιβλιογραφία 42

Βιβλιογραφία Εικόνων 43

Πηγαίος Κώδικας 44

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εικόνα 1.1 Internet of Things ……..…. σελ 09

Εικόνα 2.1 Συνδεδεμένος, πλήρως λειτουργικός κόμβος ……. σελ 18

Εικόνα 3.1 Αδιάβροχος αισθητήρας θερμοκρασίας DS18B20 ……. σελ 20

Εικόνα 3.2 Ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας DHT11 …….. σελ 21

Εικόνα 3.3 Ηλεκτρομαγνητικός διακόπτης 2 καναλιών 5 volt……. σελ 22

Εικόνα 3.4 Texas Instruments MSP4430F5529 LaunchPad pin numbers diagram …….. σελ 23

Εικόνα 4.1 Περιβάλλοντα ανάπτυξης κώδικα ……..… σελ 25

Εικόνα 4.2 Διάγραμμα ροής ……..… σελ 28

Εικόνα 4.3 Περιβάλλον ανάπτυξης Energia 14 ……..…..… σελ 32

Κεφάλαιο 1 – Εισαγωγή

1. Εισαγωγή

Στην εργασία αυτή θα ασχοληθούμε με το περιβάλλον ανάπτυξης του Energia 14 για την ανάπτυξη κώδικα για τον μικροελεγκτή της Texas Instruments – MSP430 και πιο συγκεκριμένα τον MSP430F5529 του περιβάλλοντος ανάπτυξης MSP-EXP430F5529 LaunchPad Evaluation Kit. Ο μικροελεγκτής θα προγραμματιστεί και θα συνδεθεί πιλοτικά σε τρεις κλιματιστικές μονάδες της σχολής Μηχανικών Πληροφορικής του ΤΕΙ Ηπείρου από όπου και θα συνδέεται στο ασύρματο δίκτυο, θα επικοινωνεί με ένα κέντρο ελέγχου, ένα server στη συγκεκριμένη περίπτωση, θα στέλνει τα δεδομένα που συλλέγει από τους αισθητήρες και θα λαμβάνει εντολές που αφορούν στη λειτουργία της μονάδας.

1.1. IoT – Internet of Things

Το ίντερνετ των πραγμάτων είναι ένα δίκτυο που αποτελείται από συνδεδεμένες συσκευές, δηλαδή ενσωματωμένες συσκευές που αποτελούνται από μικροεπεξεργαστές, διάφορους αισθητήρες, τρέχουν λογισμικό και συνδέονται στο ίντερνετ. Με αυτόν τον τρόπο όλες οι συνδεδεμένες συσκευές αποτελούν ένα σύνολο το οποίο ανταλλάσσει πληροφορίες είτε με το διαχειριστή είτε μεταξύ τους. Η κάθε μια συσκευή είναι σχεδιασμένη και προγραμματισμένη να εκτελεί συγκεκριμένες λειτουργίες, όπως να ανταλλάζει δεδομένα με άλλες συσκευές στο τοπικό δίκτυο ή ακόμη και στο ίντερνετ και να εκτελεί τις προγραμματισμένες εντολές ή να δέχεται καινούριες από κάποιο συγκεκριμένο εξυπηρετητή ή άλλη συσκευή.

Το Internet of Things (IoT), αναμένεται να προσφέρει εξελιγμένη συνδεσιμότητα συσκευών, συστημάτων και υπηρεσιών η οποία ξεπερνά την απλή επικοινωνία μηχανής προς μηχανή (M2M), και καλύπτει μια ποικιλία πρωτοκόλλων, τομέων και εφαρμογών. Η διασύνδεση αυτών των ενσωματωμένων συσκευών, συμπεριλαμβανομένων και των έξυπνων αντικειμένων, αναμένεται να εισαχθεί αυτόνομα σε σχεδόν όλα τα πεδία, καθώς επίσης και να εφαρμόσει εξελιγμένες εφαρμογές, όπως το Smart Grid.

Κάνοντας αναφορά στα πράγματα (Things) εννοούμε μια ευρεία κατηγορία συσκευών όπως εμφυτεύματα παρακολούθησης καρδιάς, biochip μετάδοσης σε οικόσιτα ζώα, ηλεκτρικές συσκευές σε παράκτια ύδατα, αυτοκίνητα με built-in αισθητήρες ή συσκευές λειτουργίας πεδίου που βοηθούν τους πυροσβέστες κατά την έρευνα και τη διάσωση. Τα υπάρχοντα παραδείγματα από την αγορά περιλαμβάνουν έξυπνα θερμοστατικά συστήματα και πλυντήρια-στεγνωτήρια που χρησιμοποιούν ασύρματο δίκτυο (wifi) για παρακολούθηση εξ' αποστάσεως.

Εικόνα 1 1 Internet of Things Πηγή: http://www.3g.co.uk

Πέρα από το μεγάλο αριθμό νέων περιοχών εφαρμογών της αυτόματης σύνδεσης του Ίντερνετ στις οποίες μπορεί να επεκταθεί, το IoT αναμένεται να παράγει μεγάλα ποσά δεδομένων από διαφορετικές τοποθεσίες, συγκεντρωτικά και με πολύ μεγάλη ταχύτητα. Επομένως, δημιουργείται η ανάγκη ενός καλύτερου τρόπου ή μέσου εισαγωγής, αποθήκευσης και επεξεργασίας αυτών των δεδομένων.

Πρώιμη Ιστορία

Από το 2014, η οπτική ως προς το Internet of Things έχει εξελιχθεί χάρη στην κάλυψη πολλαπλών τεχνολογιών, οι οποίες εκτείνονται από τις ασύρματες επικοινωνίες μέχρι το διαδίκτυο και από τα ενσωματωμένα συστήματα, στα μικρο-ηλεκτρομηχανικά συστήματα (MEMS). Αυτό σημαίνει ότι τα παραδοσιακά πεδία ενσωματωμένων συστημάτων, τα ασύρματα δίκτυα αισθητήρων, τα συστήματα ελέγχου, η αυτοματοποίηση, συμπεριλαμβανομένης και την αυτοματοποίηση οικιών και κτηρίων, και άλλα, συμβάλλουν στην ενεργοποίηση του Internet of Things (IoT).

Η ιδέα ενός συστήματος έξυπνων συσκευών, είχε συζητηθεί από το 1982, με μια τροποποιημένη μηχανή αναψυκτικών στο Carnegie Mellon University και έγινε η πρώτη συνδεδεμένη στο διαδίκτυο συσκευή ικανή να κάνει απογραφή και να κρίνει αν το προϊόν είναι παγωμένο. Όπως αναφέρει ο Mark Weiser (1991) στο άρθρο του για την πανταχού υπολογιστική τεχνολογία «The Computer of the 21st Century», όπως και άλλοι ακαδημαϊκή χώροι, όπως το

UbiComp και το PerCom παρήγαγαν τη σύγχρονη όψη του IoT. Το 1994, ο Reza Raji περιέγραψε την ιδέα του στο περιοδικό του ινστιτούτου IEEE spectrum ως «μεταφορά μικρών πακέτων δεδομένων σε ένα μεγάλο σύνολο κόμβων, ώστε να ενσωματώσει και να αυτονομίσει τα πάντα, από οικιακή συσκευή μέχρι και ολόκληρα εργοστάσια». Παρόλα αυτά, μόλις από το 1999, ο κλάδος ξεκίνησε να αποκτά κινητικότητα. Ο Bill Joy οραματίστηκε την επικοινωνία από Συσκευή προς Συσκευή (Device to Device – D2D), ως μέρος του πλαισίου "Six Webs", που παρουσιάστηκε το 1999 στο Παγκόσμιο Οικονομικό Φόρουμ. Ο Kevin Ashton πρότεινε τον όρο «Internet of Things», την ίδια χρονιά.

Η ιδέα του Internet of Things πρωτοέγινε γνωστή το 1999, μέσω του κέντρου AutoID στο MIT και σε σχετικές δημοσιεύσεις ανάλυσης αγοράς. Η ανίχνευση των ραδιοσυχνοτήτων (RFID) υπήρξε προαπαιτούμενο για το IoT τον πρώτο καιρό. Αν κάθε αντικείμενο και κάθε άνθρωπος στην καθημερινή ζωή, ήταν εξοπλισμένοι με ανιχνευτές, οι υπολογιστές θα μπορούσαν να του κατευθύνουν και να τους καταγράφουν. Από το να χρησιμοποιούνται RFID, η σήμανση των αντικειμένων μπορεί να γίνει με τεχνολογίες όπως επικοινωνία κοντινού πεδίου, barcodes, QR κωδικοί και ψηφιακά υδατογραφήματα.

Στην αρχική επεξήγησή του, μια από τις πρώτες επιπτώσεις της εφαρμογής του IoT με το να εξοπλίζονται όλα τα αντικείμενα στον κόσμο με μικροσκοπικές συσκευές ανίχνευσης ή μηχανές-αναγνώσιμοι ανιχνευτές, ήταν η μετατροπή της καθημερινής ζωής με διάφορους θετικούς τρόπους. Για παράδειγμα, ο στιγμιαίος και ακατάπαυστος έλεγχος απογραφής, θα ήταν δεδομένος. Η ικανότητα ενός ατόμου να αλληλοεπιδρά με τα αντικείμενα, θα μπορούσε να τροποποιηθεί εξ αποστάσεως βάσει με τις υπάρχουσες ανάγκες, σε συμφωνία με τις υπάρχουσες συμφωνίες του τελικού χρήστη. Μια τέτοια τεχνολογία, θα μπορούσε να εγγυηθεί μεγαλύτερο έλεγχο στους παραγωγούς εικόνας και κίνησης ως προς τους τελικούς χρήστες, με το να ενισχύει εξ' αποστάσεως τους περιορισμούς των πνευματικών δικαιωμάτων και τη διαχείριση των ψηφιακών περιορισμών.

Εφαρμογές

Σύμφωνα με το Gartner Inc., οργανισμός τεχνολογικής έρευνας και συμβουλευτικής, θα υπάρξουν σχεδόν 26 δισεκατομμύρια συσκευές στο IoT μέχρι το 2020. Η ABI Research εκτιμά ότι πάνω από 30 δισεκατομμύρια συσκευές θα είναι ασύρματα συνδεδεμένες στο IoT μέχρι το 2020 και το ονομάζει το Ίντερνετ των πάντων, Internet of Everything. Βάσει μιας σύγχρονης έρευνας και μελέτης του Pew Research Internet Project, μια μεγάλη πλειοψηφία των ειδικών τεχνολογίας και των συχνών χρηστών του ίντερνετ που απάντησαν, το 83% συμφώνησε με τον όρο ότι το IoT, η ενσωματωμένη και wearableυπολογιστική θα έχουν διευρυμένα και ευεργετικά αποτελέσματα μέχρι το 2025. Είναι πλέον ξεκάθαρο ότι το Internet of Things θα αποτελείται από ένα μεγάλο αριθμό συσκευών συνδεδεμένες στο διαδίκτυο. Πιο συγκεκριμένα με βάση τον τομέα εφαρμογής της κάθε συνδεδεμένης συσκευής χωρίζονται σε πέντε κατηγορίες: τις έξυπνες συσκευές που μπορούν να φορεθούν, τα έξυπνα σπίτια, τις έξυπνες πόλεις, τις έξυπνες συσκευές για τη διαχείριση και καταγραφή περιβαλλοντολογικών καταστάσεων και τις έξυπνες επιχειρήσεις.

1.2. M2M

Με τον όρο Machine to Machine (M2M) αναφερόμαστε σε τεχνολογίες που μας επιτρέπουν την επικοινωνία ασύρματων αλλά και ενσύρματων συσκευών με άλλες συσκευές του ίδιου τύπου. Γενικότερα είναι ένας ευρύς όρος καθώς δεν προσδιορίζει συγκεκριμένα την ασύρματη ή την ενσύρματη τεχνολογία δικτύωσης αλλά χρησιμοποιείται κυρίως για τη μεταφορά δεδομένων και την επικοινωνία εντός επιχειρήσεων. Είναι ένα αναπόσπαστο κομμάτι του Internet of Things που προσφέρει πολλά πλεονεκτήματα κυρίως στον εργοστασιακό τομέα και στον τομέα των επιχειρήσεων, όμως έχει επίσης μεγάλο εύρος εφαρμογών στην εργοστασιακή αυτοματοποίηση, στα δίκτυα παροχής ηλεκτρικής ενέργειας, στις έξυπνες πόλεις, στην υγεία και σε πολλούς άλλους τομείς της καθημερινότητάς μας κυρίως για την επίβλεψη αλλά και για τη διαχείριση.

Ιστορία

Οι ενσύρματες συσκευές επικοινωνίας χρησιμοποιούσαν κωδικοποίηση για την ανταλλαγή πληροφοριών, μέχρι τον 20^{ο} αιώνα. Στη συνέχεια, από την άνοδο των αυτοματοποιημένων δικτύων υπολογιστών και .την προγενέστερη τηλεφωνική επικοινωνία, η τεχνολογία Machine to Machine, η μετάδοση πληροφοριών, έλαβε πιο εξελιγμένες μορφές. Πιο συγκεκριμένα, χρησιμοποιείται σε εφαρμογές όπως η τηλεμετρία, ο βιομηχανικός αυτοματισμός και το SCADA (Supervisory Control and Data Acquisition), ένα σύστημα που επιτρέπει τον απομακρυσμένο έλεγχο συσκευών.

Οι συσκευές Machine to Machine οι οποίες συνδυάζουν την τηλεφωνία και τη χρήση υπολογιστή, υλοποιήθηκαν για πρώτη φορά από τον Θεόδωρο Γ. Παρασκευάκο το 1968 και το κατοχύρωσε ως πατέντα στις ΗΠΑ το 1973. Η τεχνολογία που υλοποίησε είναι γνωστή στο κοινό ως ταυτότητα χρήστη.

Τον 21ο αιώνα πλέον οι συσκευές για την επικοινωνία των συσκευών έχουν πιο νέα χαρακτηριστικά και πιο εξελιγμένες δυνατότητες όπως για παράδειγμα ενσωματωμένη τεχνολογία GPS, ενσωματωμένες έξυπνες κάρτες ειδικά βελτιωμένες για τη λειτουργία machine to machine, όπως για παράδειγμα οι κάρτες των κινητών τηλεφώνων, γνωστές ως MIMs δηλαδή Machine to machine identification modules, καθώς και ενσωματωμένη Java, μια πολύ σημαντική τεχνολογία για την εξέλιξη του Internet of Things.

Τα εξαρτήματα του hardware ενός δικτύου Machine to Machine, κατασκευάζονται από λίγους και σημαντικούς κατασκευαστές της αγοράς. Το 1998, η Quake Global, ξεκίνησε να κατασκευάζει M2M δορυφορικά και επίγεια modem. Η εταιρία αρχικά βασιζόταν πολύ στο δίκτυο ORBCOMMγια την παροχή υπηρεσιών επικοινωνίας μέσω δορυφόρου, όμως στη συνέχεια επέκτεινε την προσφορά των υπηρεσιών επικοινωνίας, συνδυάζοντας τόσο δορυφορικά όσο και επίγεια δίκτυα, γεγονός που της πρόσφερε πλεονέκτημα στην προσφορά προϊόντων επικοινωνίας machine to machinesε σχέση με τον ανταγωνισμό.

1.3. Wireless Sensor Networks

Τα ασύρματα δίκτυα αισθητήρων – Wireless Sensor Networks (WSN), αποτελούνται από πολλούς αισθητήρες κατανεμημένους σε κάποιο χώρο για την επίβλεψη των φυσικών ή περιβαλλοντικών συνθηκών, όπως για παράδειγμα της θερμοκρασίας του χώρου στον οποίο βρίσκονται, την υγρασία που επικρατεί, την ατμοσφαιρική πίεση κ.ά. και όλα αυτά τα δεδομένα μεταφέρονται είτε από κόμβο σε κόμβο και καταλήγουν σε ένα κεντρικό εξυπηρετητή είτε απευθείας στον server που θα επεξεργαστεί τα δεδομένα που θα λάβει από τους αισθητήρες και ανάλογα θα ειδοποιήσει για την επιθυμητή λειτουργία ή για κάποιο ενδεχόμενο σφάλμα που θα προκύψει και στη συνέχεια θα στείλει τις απαραίτητες εντολές ελέγχου στους κόμβους. Η ανάπτυξη των ασύρματων αισθητήρων ξεκίνησε και εφαρμόστηκε κυρίως από το στρατό για την παρακολούθηση στο πεδίο της μάχης αλλά σήμερα τέτοιου είδους δίκτυα έχουν πλέον εφαρμογή τόσο σε βιομηχανικό επίπεδο όσο και σε καταναλωτικό επίπεδο για παράδειγμα επιβλέποντας και κάνοντας διαχείριση βιομηχανικές διαδικασίες αλλά και παρακολούθηση της υγείας.

Τα ασύρματα δίκτυα αισθητήρων αποτελούνται από κόμβους που συνήθως αριθμούν από μερικές δεκάδες έως και αρκετές χιλιάδες από αυτούς. Κάθε αισθητήρας αποτελείται από διάφορα μέρη όπως είναι ο πομποδέκτης που είναι συνδεμένος σε εσωτερική ή εξωτερική κεραία, ο μικροελεγκτής, το κύκλωμα για την επικοινωνία με του αισθητήρες και την πηγή ενέργειας που συνήθως τροφοδοτείται από μπαταρίες. Το μέγεθος των κόμβων ποικίλει καθώς ξεκινάνε από το μέγεθος ενός παπουτσιού και θεωρητικά φτάνουν μέχρι και το μέγεθος ενός κόκκου σκόνης, αλλά ακόμα δεν έχει δημιουργηθεί λειτουργικός αισθητήρας σε τόσο μικρό μέγεθος. Το κόστος των ασύρματων αισθητήρων είναι εξίσου ανάλογο καθώς ξεκινάει από λίγα δολάρια και φτάνει μέχρι και εκατοντάδες, ανάλογα με το πόσο περίπλοκη είναι η σχεδίαση του κάθε κόμβου. Οι περιορισμοί στο μέγεθος και στο κόστος των αισθητήρων έχουν ανάλογο αποτέλεσμα στον περιορισμό των πόρων όπως για παράδειγμα της ενέργειας, της μνήμης και της επεξεργαστικής ισχύς. Η τοπολογία των ασύρματων δικτύων αισθητήρων μπορεί να είναι απλή,

όπως για παράδειγμα η τοπολογία δικτύου αστέρα και εκτείνεται σε προηγμένη τοπολογία πολλών κόμβων που αποτελούν ένα ή και περισσότερα δίκτυα πλέγματος.

Προς το παρόν τα ασύρματα δίκτυα αισθητήρων αποτελούν μέρος της ενεργούς έρευνας στον τομέα της επιστήμης των υπολογιστών και των τηλεπικοινωνιών σε αρκετά συνέδρια και διάφορα εργαστήρια κάθε χρόνο όπως είναι τα IPSN - International Conference on Information Processing in Sensor Networks, SenSys - Conference on Embedded Networked Sensor Systems και EWSN - European Conference on Wireless Sensor Networks.

Οι βασικές ως τώρα εφαρμογές είναι για την επίβλεψη κάποια περιοχής, όπου παρατηρείται κάποιο φαινόμενο και πρέπει να εποπτευθεί. Στο στρατό για παράδειγμα μια περιοχή παρατηρείται από αισθητήρες για κάποια ενδεχόμενη εισβολή ή αντίστοιχα παρατηρείται μια περιοχή με αγωγούς αερίου ή πετρελαίου.

Αντίστοιχα στον τομέα της υγείας οι συσκευές που χρησιμοποιούνται είναι δύο τύπων, οι συσκευές που φοριούνται ή βρίσκονται σε μικρή απόσταση από τον ασθενή και χρησιμοποιούνται κυρίως για την ασφάλειά του και οι εσωτερικές συσκευές που εμφυτεύονται στον ασθενή. Οι εφαρμογές τους είναι κυρίως η παρακολούθηση των άρρωστων ατόμων είτε αυτοί βρίσκονται σε κάποιο νοσοκομείο είτε βρίσκονται στο σπίτι τους.

Στη συνέχεια όσον αφορά στον τομέα της παρακολούθησης των περιβαλλοντολογικών συνθηκών κάποιες φορές η επικοινωνία αλλά και οι αισθητήρες οι ίδιοι αντιμετωπίζουν πιο δύσκολες συνθήκες λόγο του περιβάλλοντος που είναι τοποθετημένοι με σκοπό να το επιβλέπουν και να κάνουν διάφορες μετρήσεις αλλά και ως προς τη διαχείριση της ενέργειας που πρέπει να καταναλώνουν λόγο αδυναμίας σύνδεσής τους με κάποια πηγή συνεχούς ενέργειας. Τέτοιου είδους αισθητήρες είναι τοποθετημένοι στις πόλεις και παρακολουθούν την ποιότητα του αέρα και ανιχνεύουν τυχόν υπερβολικές ποσότητες βλαβερών αερίων ως προς τους πολίτες. Χρησιμοποιούνται σε δασικές περιοχές για την ανίχνευση πυρκαγιάς καθώς παρακολουθούνε τη θερμοκρασία, την υγρασία και για τυχόν αέρια που δημιουργούνται κυρίως κατά το αρχικό στάδιο μιας πυρκαγιάς ώστε να ενημερωθούν άμεσα οι πυροσβεστικές αρχές και να αντιμετωπίσουν έγκαιρα τέτοια φαινόμενα. Συστήματα πρόβλεψης της κατολίσθησης βουνών χρησιμοποιούν τα ασύρματα δίκτυα αισθητήρων με σκοπό να μετρούν τις μικρές μεταβολές που γίνονται στο έδαφος και να γίνεται πρόβλεψη για πιθανές κατολισθήσεις πολύ νωρίτερα από τη στιγμή που πρόκειται να πραγματοποιηθούν.

Συστήματα ελέγχου για την ποιότητα του νερού τοποθετούνται σε ρυάκια, ποτάμια, λίμνες, ωκεανούς και υπόγεια κοιτάσματα νερού. Με αυτόν τον τρόπο δίνεται η δυνατότητα ώστε να πετύχουμε μια πληρέστερη εικόνα σχετικά με την κατάσταση του νερού αφού παίρνουμε δείγματα από περιοχές που είναι εξαιρετικά δύσκολη η συνεχής δειγματοληψία από τους ανθρώπους όπως είναι η συνεχής δειγματοληψία ανταλλαγή δεδομένων σε πραγματικό χρόνο σχετικά με τη ροή νερού σε έναν ποταμό όπου μπορεί να αποφευχθεί μια φυσική καταστροφή όπως είναι η πλημύρα.

Ακόμη τα δίκτυα ασύρματων αισθητήρων βρίσκουν μεγάλη εφαρμογή και σε μεγάλες εργοστασιακές υποδομές. Χρησιμοποιούνται πολύ συχνά για την παρακολούθηση της καλής λειτουργίας των μηχανημάτων αλλά και για την επίβλεψη στην παραγωγή ή τη διακίνηση ενός

προϊόντος καθώς μπορούν να εγκατασταθούν σε σημεία όπου είναι πρακτικά αδύνατη η εγκατάσταση άλλων αισθητήρων που βασίζονται στη σύνδεσή τους με καλώδια, όπως για παράδειγμα σε κάποια περιστροφικά μηχανήματα αλλά και οχήματα μεταφοράς των προϊόντων. Αποθηκεύουν τα δεδομένα που συλλέγουν, όπως για παράδειγμα τη θερμοκρασία από κάποιο ψυγείο ή τη στάθμη του νερού σε κάποιο δοχείο με σκοπό να γίνει στατιστική ανάλυση των δεδομένων που συλλέγονται και να ελεγχθεί η σωστή λειτουργία των μηχανημάτων. Σημαντικό προτέρημα των ασύρματων αισθητήρων σε σχέση με τις συσκευές που χρησιμοποιούνται αποκλειστικά για την αποθήκευση δεδομένων είναι η δυνατότητα αποστολής δεδομένων «ζωντανά» τη στιγμή που γίνονται.

Κεφάλαιο 2 – Εφαρμογές και πράσινη διαχείριση

2. Εφαρμογές και πράσινη διαχείριση

Όπως υποστηρίζεται από διεθνείς φορείς, η κατασκευή βιώσιμων κτηρίων με οικολογικές προδιαγραφές, θα γίνει στο μέλλον μια κοινή πρακτική (Heerwagen, 2000). Η προϋπόθεση όμως είναι να αντιληφθούν τα εμπλεκόμενα μέλη τα οφέλη που θα αποκομίσουν από την πράσινη πρακτική αυτή (Heerwagen, 2000). Τα οφέλη που προσφέρονται σε μια εταιρεία ή έναν οργανισμό, μετατρέποντας τις υποδομές της σε «πράσινες», δηλαδή οικολογικές, και κάνοντάς τις πιο βιώσιμες, είναι πολλαπλά (Wills, 2014). Σύμφωνα με τη βασική αρχή της διαρκούς βελτίωσης της κοινωνικής, περιβαλλοντολογικής και οικονομικής επίδοσης, η οικολογική πρακτική είναι ο πιο απλός τρόπος για να επιτευχθεί ο περιορισμός των εξόδων (Wills, 2014).

Πιο συγκεκριμένα, δεδομένου ότι τα συστήματα θέρμανσης, εξαερισμού και ψύξης ενός κτηρίου σπαταλάνε την περισσότερη ενέργεια και συμβάλλουν σημαντικά στη εκπομπή διοξειδίου του άνθρακα, η αντικατάσταση αυτών των συστημάτων από νέα, μπορεί να συμβάλλει σημαντικά στη μείωση του λειτουργικού κόστους (NRDC, 2015). Σύμφωνα με την Αμερικάνικη Υπηρεσία Περιβαλλοντικής Προστασίας EPA (Environmental Protection Agency), η αντικατάσταση έστω και μερών των προαναφερθέντων συστημάτων με πιο λειτουργικά, μπορεί να συμβάλλει στη μείωση του κόστος μέχρι και είκοσι τοις εκατό (20%).

Πιο συγκεκριμένα, σε διάφορες χώρες, όλο και περισσότεροι διαχειριστές εγκαταστάσεων ή ιδιοκτήτες κτηρίων γενικότερα, επιλέγουν την εγκατάσταση συστημάτων αυτοματοποίησης κτηρίων (Building Automation Systems – BAS) ώστε να διαχειρίζονται με μεγαλύτερη ευκολία, αλλά και να ελέγχουν τα κόστη θέρμανσης και ψύξης των κτηρίων (Callahan, 2015). Σημαντικόείναιναση μειωθείότιένασύστημααυτοματοποίησηςκτηρίωνμπορείναεξοικονομήσει από $0.20 μέχρι $0.40 ανά τετραγωνικό μέτρο στα έξοδα διαχείρισης, έπειτα από εκτίμηση της Μητροπολιτικής επιτροπής ενεργειακής πολιτικής της Μινεάπολις (Callahan, 2015). Αυτό προκύπτει καθώς με την απλή πρακτική του να χρησιμοποιείς την ενέργεια μόνο όταν τη χρειάζεσαι μπορείς να εξοικονομήσεις χρηματικούς πόρους, όμως πολλές φορές αυτό δεν εφαρμόζεται. Για την εξασφάλιση αυτής της πρακτικής, οι διαχειριστές ή οι ιδιοκτήτες κτηρίων, μπορούν να κάνουν χρήση των έξυπνων δικτύων.

2.1. Smart HVAC Controls

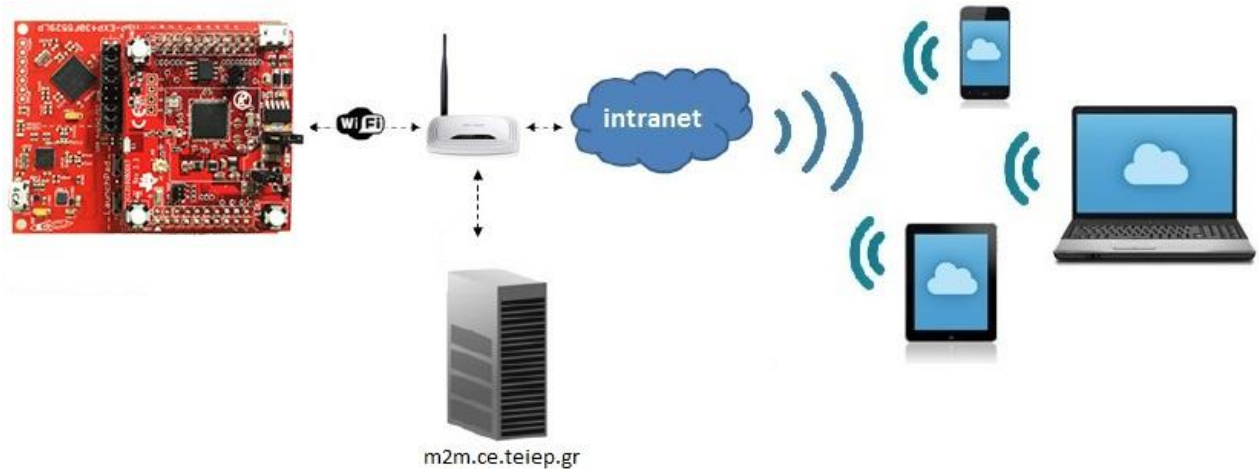
Το Internet of things είναι ένα καινούριο σενάριο όσον αφορά στον τομέα της διαχείρισης των κλιματιστικών μονάδων. Παρέχει τη δυνατότητα διαχείρισής τους πιο οικονομικά και από απόσταση. Οι μονάδες αυτές πλέον έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους και να λαμβάνουν αποφάσεις που είναι πιο ακριβείς, ενεργειακά αποδοτικές και να στέλνουν διάφορα δεδομένα και να διαχειρίζονται από απόσταση. Αυτό είναι επιθυμητό κυρίως από διαφόρους τομείς με μεγάλες εγκαταστάσεις, όπως είναι τα εργοστάσια, τα δημόσια κτήρια, τα νοσοκομεία αλλά και τα σπίτι, όπου υπάρχει ανάγκη για την καλύτερη δυνατή αποδοτικότητα των κλιματιστικών μονάδων ώστε να λειτουργούν πιο αποδοτικά και να πετυχαίνουν σημαντικά μεγάλη εξοικονόμηση ενέργειας. (blueapp.io, 2015)

Στις βιομηχανίες η περισσότερη κατανάλωση ενέργειας γίνεται από τις μονάδες θέρμανσης, ψύξης και εξαερισμού. Αυτή η αλλαγή που θα γίνει στη διαχείριση των μονάδων έχει άμεση επίπτωση στην εξοικονόμηση ενέργειας που ως αποτέλεσμα έχει την σημαντική μείωση των εκπομπών διοξειδίου του άνθρακα και τη μείωση των εξόδων. Αυτό επιτυγχάνεται αυτοματοποιώντας την διαδικασία ελέγχου των κλιματιστικών μονάδων με τη χρήση αισθητήρων. Διάφοροι τύποι που είναι οι πιο δημοφιλείς και χρησιμοποιούνται για τέτοιου είδους αυτοματισμούς είναι:

- Οι αισθητήρες θερμοκρασίας, που μετράνε ανά τακτά χρονικά διαστήματα τη θερμοκρασία του χώρου και τη διατηρούν στα επιθυμητά επίπεδα.
- Οι αισθητήρες υγρασίας, για τον έλεγχο της υγρασίας σε χώρους όπου αυτό είναι απαραίτητη.
- Οι αισθητήρες πίεσης, χρησιμοποιούνται κυρίως σε εγκαταστάσεις όπου η πίεση του χώρου πρέπει να διατηρείται σε συγκεκριμένα επίπεδα.
- Οι αισθητήρες ποιότητας του αέρα, που βοηθούν και αυτοί στην εξοικονόμηση ενέργειας καθώς ελέγχουν τους ανεμιστήρες και τη ροή του αέρα σε κάποιο χώρο.

Με αυτό τον τρόπο τα έξυπνα κλιματιστικά συστήματα δίνουν τη δυνατότητα στους χρήστες να αλλάζουν άμεσα τη λειτουργία των ενεργειακά μη αποδοτικών μονάδων διατηρώντας έτσι τις απαιτούμενες τιμές, θερμοκρασίας, υγρασίας, πίεσης και ποιότητας του αέρα όπου αυτό είναι απαραίτητο. Με τη χρήση έξυπνου υλικού και λογισμικού παρέχεται επίσης η δυνατότητα απευθείας σύνδεσης με σταθμούς πρόγνωσης του καιρού με αποτέλεσμα να επιτυγχάνεται γρήγορη προσαρμογή της θερμοκρασίας ενός χώρου δημιουργώντας πιο άνετες συνθήκες πριν ακόμη αυτό γίνει αναγκαίο. Τέλος όλοι οι χρήστες θα έχουν τη δυνατότητα να παρακολουθούνε τη λειτουργία των έξυπνων συσκευών και να τις διαχειρίζονται εξ’ αποστάσεως ακόμη και μέσω του smartphone. (blueapp.io, 2015)

Έτσι, κατανοώντας την ανάγκη και τα οφέλη από την έξυπνη διαχείριση των κλιματιστικών συστημάτων, στη συνέχεια αναπτύσσεται ένας έξυπνος ασύρματος αισθητήρας διαχείρισης των κλιματιστικών μονάδων. Στόχος είναι να προγραμματιστεί και να συνδεθεί στο κλιματιστικό σύστημα από όπου θα συνδέεται στο ασύρματο δίκτυο, θα επικοινωνεί με το server, θα στέλνει τα δεδομένα που συλλέγει από τους αισθητήρες και θα λαμβάνει εντολές που θα αφορούν στη λειτουργία της μονάδας. (blueapp.io, 2015)



Εικόνα 2.1 Συνδεδεμένος, πλήρως λειτουργικός κόμβος

Το έξυπνο σύστημα διαχείρισης αποτελείται από ένα μικροελεγκτή, που διαχειρίζεται την επικοινωνία με το server, καθώς και τη λήψη και αποστολή των τιμών θερμοκρασίας και υγρασίας από τους αισθητήρες. Επίσης, λαμβάνει εντολές για τη διαχείριση της κλιματιστικής μονάδας, που πραγματοποιείται μέσω του συνδεδεμένου ηλεκτρομαγνητικού διακόπτη. Στο επόμενο κεφάλαιο, γίνεται αναλυτική περιγραφή των επιμέρους υλικών και εξαρτημάτων που θα χρησιμοποιηθούν.

Κεφάλαιο 3 – BOM και συνδεσμολογία

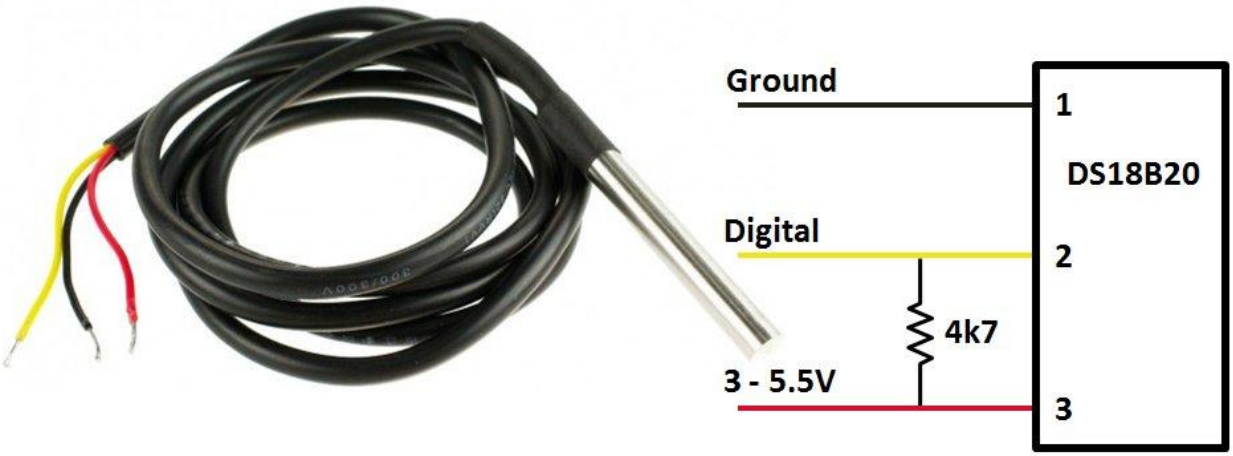
3. Απαιτούμενο Υλικό

Τα υλικά που χρειάζονται για τη δημιουργία του ασύρματου αισθητήρα για την παρακολούθηση της θερμοκρασίας και της υγρασίας του χώρου και της θερμοκρασίας του ψυκτικού υγρού της κλιματιστικής μονάδας είναι τα παρακάτω.

- Ο μικροελεγκτής της Texas Instruments MSP-EXP430F5529LP στην έκδοση του firmware 1.0.0.
- Το booster pack της Texas Instruments CC3100 BOOST rev.3.3 για την σύνδεση με το ασύρματο δίκτυο.
- Ένας ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας χώρου DHT11.
- Μια αντίσταση 4.7k
- Jumper wires female to female
- Jumper wires male to female
- Ένας ψηφιακός αδιάβροχος αισθητήρας DS18B20 για τη μέτρηση της θερμοκρασίας του ψυκτικού υγρού της κλιματιστικής μονάδας.
- Δύο 5V Relay για τη διαχείριση των καταστάσεων (On/Off και Hi/Low) της κλιματιστικής μονάδας.

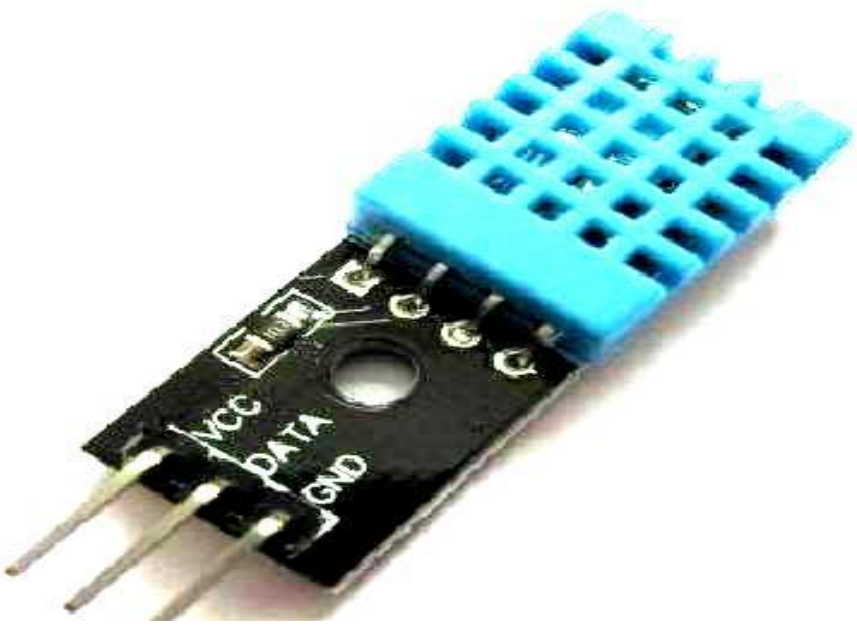
3.1. Συνδεσμολογία

Η σύνδεση του αισθητήρα DS18B20 για τη μέτρηση της θερμοκρασίας του ψυκτικού υγρού της κλιματιστικής μονάδας απαιτεί σύνδεση με παροχή ρεύματος 3.3 – 5 volt dc, γείωση και σύνδεση σε pinπου διαβάζει ψηφιακά δεδομένα. Η αντίσταση 4.7k χρησιμοποιείται ως pull up resistor, συνδέεται στην παροχή ρεύματος και στο pin των δεδομένων όπως φαίνεται στην εικόνα 3.1.



Εικόνα 3.1 Αδιάβροχος αισθητήρας θερμοκρασίας DS18B20 πηγή: cm.lnwfile.com

Για τη σύνδεση του αισθητήρα θερμοκρασίας και υγρασίας DHT11 θα χρειαστούμε σταθερής τάσης ρεύμα από 3.3volt έως 5volt, σύνδεση με γείωση και σύνδεση του pinγια την αποστολή των ψηφιακών δεδομένων από τις μετρήσεις του αισθητήρα. Εικόνα 3.2.

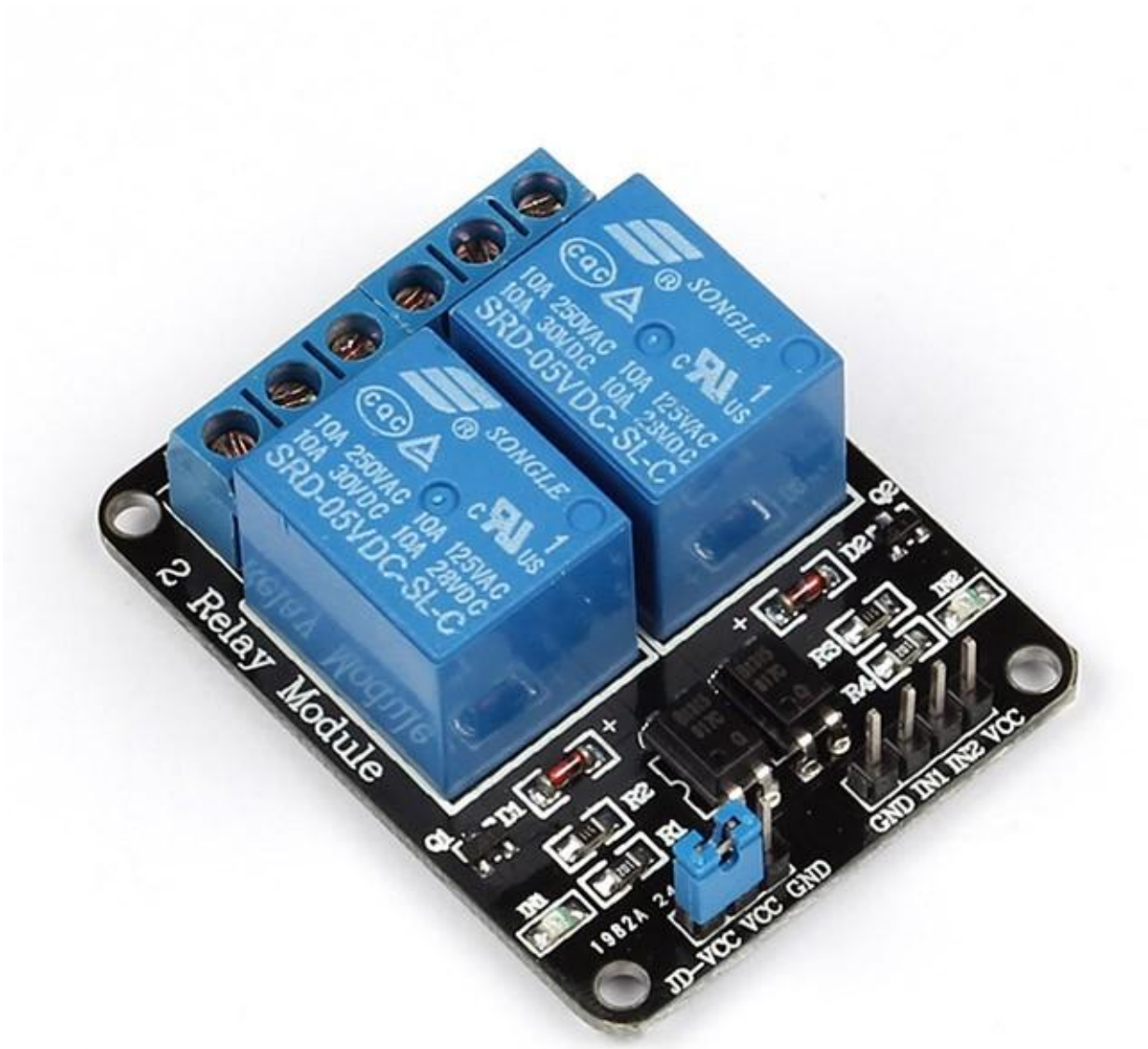


Εικόνα 3.2 Ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας DHT11 πηγή: cmostronics.in

Αρχικά θα συνδέσουμε το CC3100 boosterpack στο 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 για δεδομένα.

Χρησιμοποιώντας jumper wires θα συνδέσουμε το pin Vcc του αισθητήρα DHT11 με το pin1 του μικροελεγκτή για παροχή σταθερού ρεύματος τάσης 3.3volt και το pinτης γείωσης με το pin 20 της γείωσης του μικροελεγκτή. Τα πράσινα pinsόπως φαίνονται στην εικόνα 4 είναι είσοδοι και έξοδοι ψηφιακών δεδομένων. Το pinπου θα συνδέσουμε τον DHT11 είναι το p3.7 ή 32 σύμφωνα με την αρίθμηση του Energia.

Θα συνδέσουμε τον αισθητήρα DS18B20 για τη μέτρηση της θερμοκρασίας του ψυκτικού υγρού στο pin 21 παροχής σταθερής τάσης 5voltκαι στο pin22 της γείωσης. Τα δεδομένα θα τα διαβάζουμε από το pin p6.5 ή 2. Η αντίσταση θα συνδεθεί αντίστοιχα στο pin21 και 22.

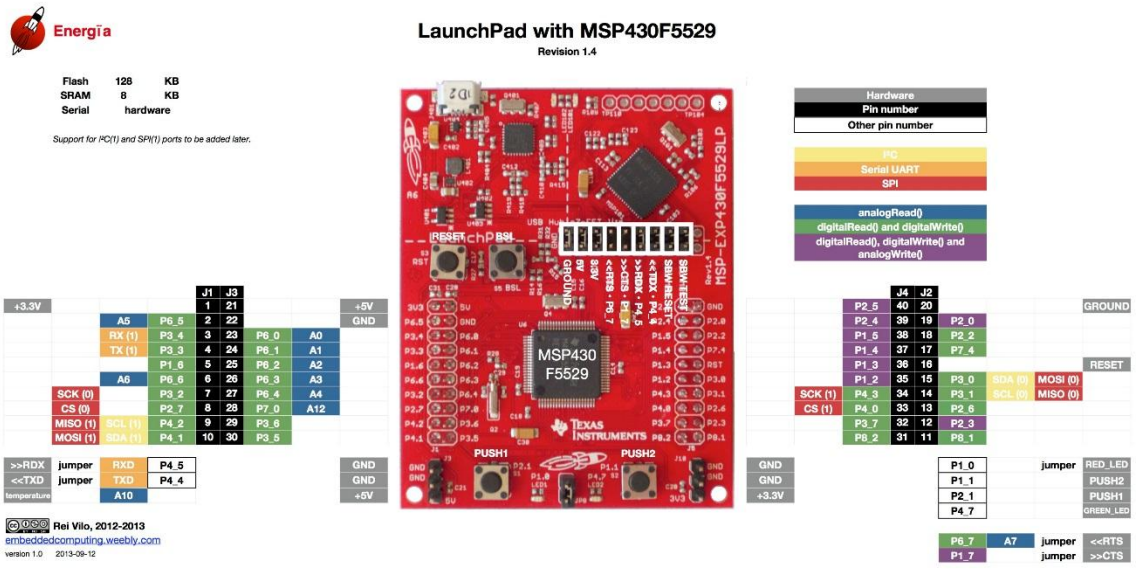


Εικόνα 3.3 Ηλεκτρομαγνητικός διακόπτης 2 καναλιών 5 volt. Πηγή: ecx.images-amazon.com

Η σύνδεση των relay θα γίνει με jumper wires female to male και θα συνδεθούν στα αντίστοιχα pins 5volt και της αντίστασης 21 και 20. Η είσοδος in1 θα συνδεθεί στο pin p7.4 ή 17 και η είσοδος in2 θα συνδεθεί στο pin p3.6 ή 29 εικόνα 3.3.

3.2. Διαδικασία flash image

Ο προγραμματισμός του μικροελεγκτή θα γίνει μέσα από το Energia 14. Θα χρειαστεί να εισάγουμε στο Energia τις αντίστοιχες βιβλιοθήκες για τους αισθητήρες που χρησιμοποιούμε. Θα τις κατεβάσουμε από το επίσημο forumtou Energia, το forum.43oh.com. Τη βιβλιοθήκη για τον αισθητήρα θερμοκρασίας και υγρασίας DHT11 θα τη κατεβάσουμε από το http://forum.43oh.com/topic/2873-energia-library-ladyadas-dht-library-ported/. Θα κάνουμε αποσυμπίεση το αρχείο και θα αντιγράψουμε το φάκελο στο φάκελο με τις βιβλιοθήκες του Energia στο C:\Program Files\energia-0101E0014\hardware\msp430\libraries. Αφού κατεβάσουμε και τη βιβλιοθήκη για τον αισθητήρα θερμοκρασίας DS18B20 από το http://forum.43oh.com/topic/3314-energia-library-onewire-ds18b20-430-stellaris/, θα κάνουμε αποσυμπίεση μέσα στο φάκελο με τις βιβλιοθήκες. Στη συνέχεια κάνουμε εισαγωγή το sketch με όνομα CC3100HVAC.ino,συνδέουμε τον μικροελεγκτή στον υπολογιστή σε μια θύρα USB και από το menu επιλέγουμε tools -> board και από εκεί επιλέγουμε το Launchpad w/ MSP430F5529LP (25MHz). Στη συνέχεια πατάμε το βελάκι στο Energia για να κάνουμε compile και να κάνουμε flash τον μικροελεγκτή με το πρόγραμμα για τη διαχείριση της κλιματιστικής μονάδας.



Εικόνα 3.4 Texas Instruments MSP430F5529 LaunchPad pin numbers diagram Πηγή: Energia.nu

Κεφάλαιο 4 – Προγραμματισμός μικροελεγκτή

4. Περιβάλλον ανάπτυξης Energia

Το Energia είναι ένα περιβάλλον ανάπτυξης ανοιχτού κώδικα για τον προγραμματισμό μικροελεγκτών κυρίως της Texas Instruments. Ξεκίνησε από τον Robert Wessels τον Ιανουάριο του 2012 με στόχο τον προγραμματισμό των μικροελεγκτών MPS430 μέσα από το γνώριμο περιβάλλον ανάπτυξης του Arduino και τη γλώσσα Wiring. Υποστηρίζει όλα τα λειτουργικά συστήματα όπως το MAC OS, τα Windows, και το Linux. Χρησιμοποιεί τον mspgcc – του Piter Bigot – για τις μεταγλωττίσεις και είναι βασισμένο στο Arduino. Το Energia περιλαμβάνει επιπλέον και το περιβάλλον ανάπτυξης που είναι βασισμένο στην Processing. Ο κώδικας του Energia είναι φορητός και μπορεί να χρησιμοποιηθεί και σε άλλα περιβάλλοντα ανάπτυξης, όπως είναι το Xcode, το Visual Studio και το Code Composer Studio, χάρη στις επεκτάσεις που έχουν δημιουργηθεί μέσω της κοινότητας των χρηστών.

Εικόνα 4 1 Περιβάλλοντα ανάπτυξης κώδικα. Πηγή: www.ti.com

Το Energia όπως και το Arduino είναι βασισμένο στη Wiring που δημιουργήθηκε και αναπτύχθηκε από τον Hernando Barragan έχοντας στο μυαλό τους σχεδιαστές και τους καλλιτέχνες για την ενθάρρυνσή και των αρχαρίων αλλά και των επαγγελματιών προγραμματιστών στην ανάπτυξη και στο διαμοιρασμό των ιδεών τους μέσω της ανοιχτής κοινότητας. Η φιλοσοφία του Energia είναι να μαθαίνεις δημιουργώντας και γίνεται προσπάθεια για την ανάπτυξη εφαρμογών που τρέχει κατευθείαν στις συσκευές. Επαγγελματίες μηχανικοί, επιχειρηματίες, κατασκευαστές ακόμη και μαθητές μπορούν να επωφεληθούν από την ευκολία χρήσης του Energia για τον προγραμματισμό των μικροελεγκτών.

Οι υποστηριζόμενοι μικροελεγκτές είναι της Texas Instruments που είναι βασισμένοι στον MSP430. Πρόσφατα όμως υποστηρίζονται και αρκετοί άλλοι όπως είναι ο MSP432, ο TM4C, ο C2000 και ο CC3200. Ακόμη υποστηρίζεται και το kit ανάπτυξης από την κοινότητα χρηστών της RedBearLab που είναι και αυτό βασισμένο στο CC3200. Έτσι τα LaunchPad μπορούν εύκολα να χρησιμοποιηθούν για την αλληλεπίδρασή τους με άλλα περιφερειακά όπως είναι οι διακόπτες και οι αισθητήρες και να χειρίζονται led και μοτεράκια. Η επικοινωνία με τον μικροελεγκτή γίνεται μέσω σύνδεσης με τον υπολογιστή αλλά υπάρχει και η δυνατότητα ασύρματης επικοινωνίας μέσω WiFi, NFC, Bluetooth, Zigbee ακόμα και GPRS.

4.1. Project

Στόχος του προγράμματος είναι έλεγχος της κλιματιστικής μονάδας του κτηρίου μέσω του MSP430F5529LP + CC3100BOOSTER της Texas Instruments.

Αρχικά αφού συνδέσουμε τον μικροελεγκτή στη μονάδα κλιματισμού και τροφοδοτήσουμε με ρεύμα τη συσκευή τότε ξεκινάει να τρέχει το πρόγραμμα. Το πρόγραμμα ξεκινάει με τα δύο Led (Red, Green) αναμμένα δείχνοντας πως βρίσκεται σε κατάσταση configuration. Έχει ξεκινήσει σε SmartConfig Mode, ψάχνει στα αποθηκευμένα προφίλ για τη σύνδεση του σε κάποιο γνωστό ασύρματο δίκτυο. Αν δεν υπάρχει τότε θα πρέπει μέσω της εφαρμογής WIFI Starter (Android/iOS) να ορίσουμε ένα ασύρματο δίκτυο.

Μόλις συνδεθεί και πάρει διεύθυνση IP τότε τα Led σβήνουν, η συσκευή προσπαθεί να κάνει επίλυση του URL του server. Αν δεν γίνει επιτυχώς τότε το κόκκινο Led ανάβει για 2 δευτερόλεπτα και ξαναγίνεται προσπάθεια επίλυσης του URL του server μέχρι να γίνει επιτυχώς.

Στη συνέχεια γίνεται προσπάθεια σύνδεσης με το server, αν δεν γίνει η σύνδεση τότε ανάβει το κόκκινο Led για 30 δευτερόλεπτα και ξαναγίνεται προσπάθεια σύνδεσης.

Στην κύρια λειτουργία του προγράμματος γίνεται έλεγχος αν υπάρχει σύνδεση με το ασύρματο δίκτυο, αν έχουμε σύνδεση με το server και αν έχει περάσει ο χρόνος (2 λεπτά) για την αποστολή μηνύματος στο server με τις τιμές που διαβάζονται από τους αισθητήρες με τη συνάρτηση createMsg(). Παράλληλα αν ο server στείλει κάποιο μήνυμα εντολής με την τιμή check=false, θα διαβαστεί από τη συνάρτηση serverMsg() και τότε θα γίνουν οι αντίστοιχες αλλαγές στη λειτουργία της συσκευής και θα σταλεί πίσω καινούργιο μήνυμα με τις ενημερωμένες πλέον τιμές.

Αν η σύνδεση με το server χαθεί τότε γίνεται προσπάθεια μέχρι να συνδεθεί η συσκευή μαζί του. Αν η σύνδεση με το Access Point χαθεί τότε γίνεται επανεκκίνηση της συσκευής, έτσι με αυτόν τον τρόπο το msgCount μηδενίζεται και μας δίνει τη δυνατότητα μέσα από τα logs του server να βλέπουμε κάθε πότε χάνεται η σύνδεση με το Access Point.

Οι βιβλιοθήκες που χρησιμοποιήθηκαν για τους αισθητήρες :

- DHT11: http://forum.43oh.com/topic/2873-energia-library-ladyadas-dht-library-ported/
- DS18B20: http://forum.43oh.com/topic/3314-energia-library-onewire-ds18b20-430-stellaris/

Υποστηρίζονται από μέλη της κοινότητας του επίσημου forum http://forum.43oh.com του Energia.

serverMsg()

Το μήνυμα που δέχεται είναι τύπου JSON και περιμένει να διαβάσει τις ακόλουθες τιμές:

- Check : Boolean
- msgCount : int
- OnOff : Boolean
- HiLow : Boolean
- WinterSummer : Boolean

createMsg()

Δημιουργείται ένα μήνυμα τύπου JSON για την αποστολή των μετρήσεων από τους αισθητήρες. Το μήνυμα είναι ένα αντικείμενο JSON της μορφής :

{

"NodeID": "00:11:22:33:44:55",

"Type": "ControlSensor/Sensor",

"Humidity": <int>,

"Temperature": <int>,

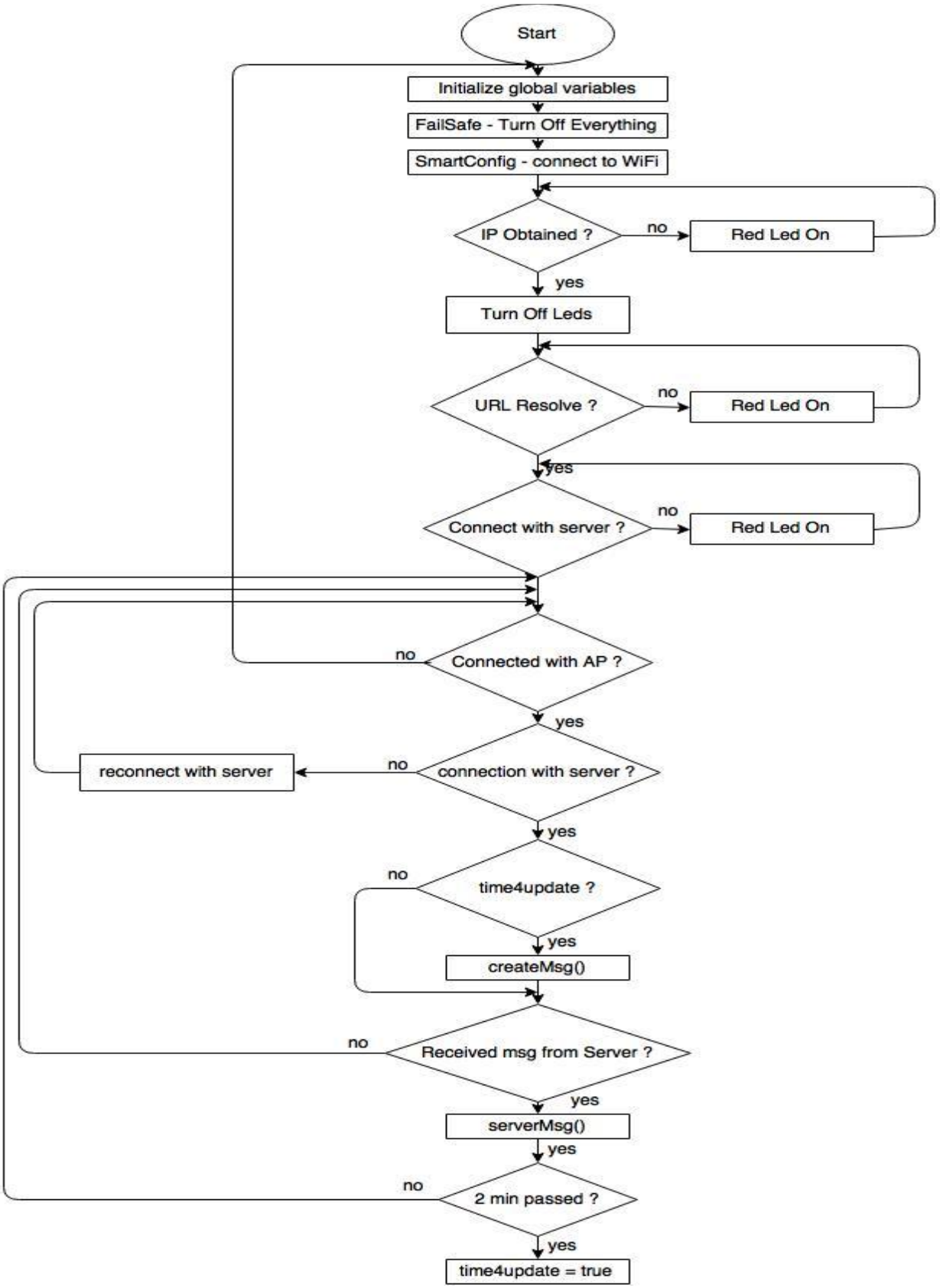
"CoolantTemp": <int>,

"OnOff": <Boolean>,

"HiLow": <Boolean>,

“msgCount”: <int>

}



Εικόνα 4 2 Διάγραμμαροής

4.2. Project Code

4.2.1. Header Files, Libraries, Global Variables

Αρχικά γίνεται εισαγωγή των header files, SPI.h, WiFi.h, WiFiClient.h, DHT.h, aJSON.h και GFDS18B20.h, που περιέχουν όλο το API (Application Programming Interface) που είναι απαραίτητο για τη λειτουργία του προγράμματος. Στη συνέχεια γίνεται ο ορισμός των τιμών των pinγια την επικοινωνία, την αποστολή και τη λήψη δεδομένων με τους αισθητήρες και τη διαχείριση του Relay. Γίνεται ο ορισμός των λεπτών που πρέπει να περιμένει η συσκευή μέχρι να ξαναστείλει τα καινούρια δεδομένα, τα λεπτά που περιμένει μέχρι να μεταβεί σε failsafe mode, το URL του server που θα γίνει επίλυση και προσπάθεια σύνδεσης για την αποστολή των δεδομένων. Ακόμη κάνουμε δήλωση της πόρτας που ακούει ο serverγια τη λήψη των δεδομένων, του χρόνου που πέρασε από την τελευταία αποστολή μηνύματος στο serverκαι του χρόνου που δεν έχουμε σύνδεση με το server προκειμένου να μεταβούμε σε failsafe mode.

Στη συνέχεια γίνεται δήλωση των μεταβλητών που χρησιμοποιούνται από τη βιβλιοθήκη του OneWire για τη λήψη της θερμοκρασίας από τον αισθητήρα DS18B20, οι μεταβλητές για το JSON Object, οι μεταβλητές για την αποστολή και λήψη του stringμε τις μετρήσεις των αισθητήρων και την κατάσταση του Relay στο server, οι μεταβλητές για την αποθήκευση των τιμών των αισθητήρων, των καταστάσεων του Relay και της αρχικοποίησης των DHT11 και DS18B20.

4.2.2. Setup function

Στη ρουτίνα Setup() ξεκινάμε το configuration της συσκευής ορίζοντας αρχικά τα δύο Led, (GREEN_LED) πράσινο και (RED_LED) κόκκινο ως εξόδους και τα θέτουμε LOW, με σκοπό να τα χρησιμοποιήσουμε ως οπτικές ενδείξεις προς το χρήστη για τη σωστή λειτουργία του προγράμματος (GREEN_LED ->ON) ή αν παρουσιαστεί κάποιο σφάλμα (RED_LED ->ON).

Έπειτα θα ορίσουμε ως εξόδους τα pin που θα διαχειρίζονται τις καταστάσεις των Relay – ONOFF_PIN, FAN_PIN, WINSUM_PIN – και τα θέτουμε σε HIGH γιατί όταν στην είσοδο του Relay φτάνει τάση από 3.3 volt μέχρι 5 volt τότε η κατάσταση του είναι ΛΟΓΙΚΟ ΑΝΟΙΧΤΟ, άρα θέτουμε το διακόπτη σε θέση OFF.

Δηλώνουμε το boundrate της σειριακής κονσόλας στα 9600, σε περίπτωση που χρειάζεται να γίνει κάποια αποσφαλμάτωση (Debug) ως προς τη λειτουργία του προγράμματος, θέτουμε και τα δυο Led σε HIGH ενημερώνοντας το χρήστη πως η συσκευή είναι σε κατάσταση SmartConfig και περιμένει να γίνει η σύνδεσή της σε κάποιο ασύρματο δίκτυο μέσω της εφαρμογής Wifi Starter.

Τυπώνουμε στη σειριακή κονσόλα το μήνυμα “Started SmartConfig Mode” και ξεκινάμε το SmartConfig Mode μέσω της βιβλιοθήκης WiFi. Το παράδειγμα που παρέχεται με την προκαθορισμένη βιβλιοθήκη του Energia λειτουργεί ως Proof of Concept της λειτουργίας SmartConfig και αν το αφήσουμε ως έχει τότε κάθε φορά που η εκτέλεση του κώδικα θα

ξεκινάει και θα φτάνει σε αυτό το σημείο τα αποθηκευμένα προφίλ θα διαγράφονται και θα περιμένει να το συνδέσουμε πάλι σε κάποιο ασύρματο δίκτυο. Γι' αυτό θα επεξεργαστούμε το αρχείο της βιβλιοθήκης και θα αλλάξουμε τον κώδικα έτσι ώστε να μη γίνεται η οποιαδήποτε διαγραφή των αποθηκευμένων προφίλ των ασύρματων δικτύων και αν υπάρχει κάποιο διαθέσιμο αποθηκευμένο δίκτυο να συνδέεται αμέσως. Στα Windows μεταβαίνουμε στο φάκελο C:\ProgramFiles\energia-0101E0014\hardware\msp430\libraries\WiFi και ανοίγουμε προς επεξεργασία το αρχείο WiFi.cpp. Το πρόγραμμα που χρησιμοποιήθηκε για την επεξεργασία όλων των αρχείων κατά τη διάρκεια της ανάπτυξης του κώδικα είναι το Notepad++, καθώς προσφέρει υποστήριξη πολλών γλωσσών προγραμματισμού και τονίζει τη σύνταξη των δεσμευμένων λέξεων.

Στο αρχείο WiFi.cpp κάνουμε αναζήτηση πατώντας ctrl + F τη λέξη "smartconfig" και πατάμε εύρεση μέχρι να μεταφερθούμε στη συνάρτηση int WiFiClass::startSmartConfig(). Εκεί αρχικά θα βάλουμε σε σχόλια τις γραμμές του κώδικα που σβήνουν τα αποθηκευμένα προφίλ βάζοντας δύο slash. Έτσι θα γίνει: `// if(sl_WlanProfileDel(WLAN_DEL_ALL_PROFILES) < 0)`

```
// return -1;
```

Μετά θα σχολιάσουμε και τον κώδικα που βάζει τη συσκευή σε SmartConfig Mode και περιμένει μέχρι μέσω της εφαρμογής WiFiStarter να ξανασυνδέσουμε τη συσκευή στο ασύρματο δίκτυο. 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 θα σχολιάσουμε και το κομμάτι του κώδικα που αλλάζει την κατάσταση του CC3100 μόνο σε AutoMode και απενεργοποιεί το SmartConfig Mode.

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

Σύμφωνα με το API του CC31xx από την Texas Instruments για να ορίσουμε την ασύρματη λειτουργία της συσκευής θα το κάνουμε μέσω της συνάρτησης `sl_WlanPolicySet`,

```
int sl_WlanPolicySet ( unsigned char       Type,
```

```
    const unsigned char  Policy,
    unsigned char *      pVal,
    unsigned char        ValLen
)
```

όπου ως δεύτερο όρισμα μέσω της συνάρτησης `SL_CONNECTION_POLICY(0,0,0,0,0)` ορίζουμε την ασύρματη λειτουργία. Το πρώτο όρισμα καθορίζει τη λειτουργία Auto που με βάση την προτεραιότητα των αποθηκευμένων προφίλ συνδέεται στο ασύρματο δίκτυο. Το δεύτερο όρισμα καθορίζει τη λειτουργία Fast που σε συνδυασμό με το Auto συνδέει τη συσκευή στο πρόσφατο ασύρματο δίκτυο που είχε γίνει σύνδεση και το τελευταίο όρισμα είναι η λειτουργία AutoSmart Config σε συνδυασμό με την Auto το CC3100 θα συνδεθεί σε κάποιο από τα αποθηκευμένα προφίλ αλλά σε περίπτωση που δεν υπάρχει κάποιο τότε θα παραμείνει σε λειτουργία SmartConfig και θα περιμένει να κάνει κάποια σύνδεση ο χρήστης, γι' αυτό το λόγο η πολιτική σύνδεσης θα είναι `SL_CONNECTION_POLICY(1,0,0,0,1)` σε Auto και AutoSmart Config.

Μόλις γίνει σύνδεση σε κάποιο ασύρματο δίκτυο τότε το πρόγραμμα θα περιμένει για να πάρει διεύθυνση IP από των DHCP Server του δικτύου κάνοντας έλεγχο κάθε 300 msec. Τότε θα σβήσουν και τα δύο Led αποδεικνύοντας στο χρήστη πως έχει γίνει σύνδεση με το ασύρματο δίκτυο και η συσκευή έχει πάρει διεύθυνση IP.

Αμέσως μετά θα γίνεται έλεγχος για την επίλυση του URL του server που θα στέλνονται τα μηνύματα. Αν δε γίνει επιτυχώς επίλυση από τους DNS servers τότε θα ανάψει το κόκκινο λαμπάκι για 2 δευτερόλεπτα ενημερώνοντας το χρήστη ότι υπάρχει πρόβλημα και θα ξαναγίνει προσπάθεια επίλυσης μέχρι να γίνει επιτυχώς.

Ο επόμενος έλεγχος που γίνεται αφορά στην εγκαθίδρυση TCP σύνδεσης με το server για τη διαχείριση της συσκευής. Κάθε εκατό αποτυχημένες προσπάθειες σύνδεσης ανάβει το κόκκινο λαμπάκι για 30 δευτερόλεπτα και ξαναγίνεται προσπάθεια σύνδεσης μέχρι να γίνει επιτυχώς. Μόλις συνδεθεί με το server τότε τρέχουν οι συναρτήσεις αρχικοποίησης του αισθητήρα θερμοκρασίας του ψυκτικού υγρού και του αισθητήρα θερμοκρασίας και υγρασίας χώρου και ξεκινάει ο μετρητής του χρόνου για την αποστολή κάθε δύο λεπτά των μηνυμάτων στο server.

4.2.3. Loop function

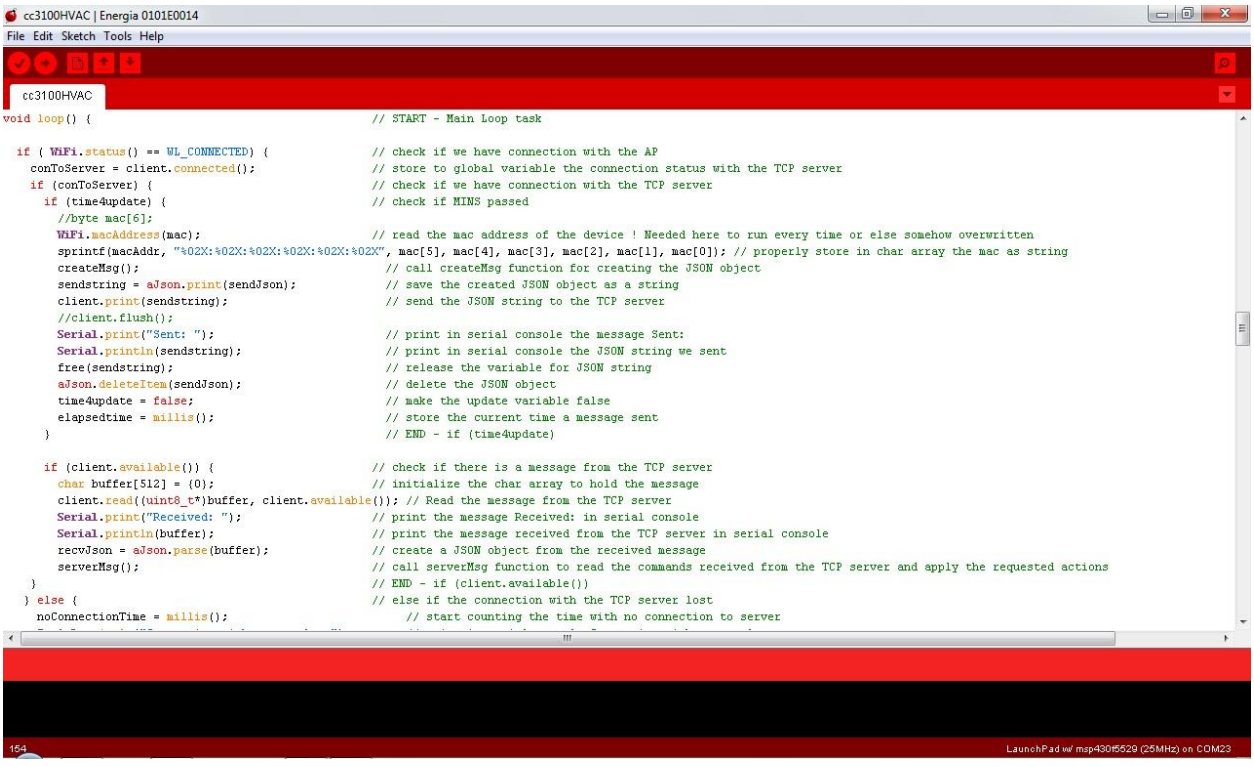
Στην κύρια συνάρτηση, τη loop() αρχικά γίνεται έλεγχος για το αν υπάρχει σύνδεση με το Access Point και το ασύρματο δίκτυο που είναι απαραίτητο για την επικοινωνία και την αποστολή μηνυμάτων στο server. Αμέσως μετά αποθηκεύεται στη μεταβλητή `conToServer` η κατάσταση της σύνδεσης με το server, δηλαδή αν είναι ενεργή τότε η μεταβλητή θα είναι αληθής ενώ αν δεν υπάρχει ενεργή σύνδεση τότε θα είναι ψευδής, και γίνεται έλεγχος για την κατάσταση της σύνδεσης. Αν η σύνδεση είναι ενεργή τότε γίνεται ο επόμενος έλεγχος με βάση τη μεταβλητή `time4update` που γίνεται αληθής κάθε δύο λεπτά ή όταν έρθει ένα νέο μήνυμα από το server.

Τότε αποθηκεύεται στη μεταβλητή `mac` η mac address του CC3100 λόγω της μοναδικότητάς της και θα χρησιμοποιηθεί ως αναγνωριστικό της συσκευής από το server. Αυτό θα πρέπει να γίνεται κάθε φορά πριν τη δημιουργία και την αποστολή του μηνύματος γιατί κατά τη διάρκεια των δοκιμών, μετά από την αποστολή 14 μηνυμάτων στο server η πρώτη θέση του πίνακα όπου

αποθηκεύονται οι τιμές της διεύθυνσης mac, δηλαδή η `mac[0]` χάνει τα δεδομένα που έχει αποθηκευμένα και αντικαθίστανται από τυχαίες συμβολοσειρές. Αυτό οφείλεται στο ότι η συνάρτηση sprintf() έχει bugκαθώς προέρχεται από τη βιβλιοθήκη του Arduino αυτούσια και χρειάζεται κάποιες τροποποιήσεις για να τρέξει χωρίς πρόβλημα και στα LaunchPad της Texas Instruments, όπως φαίνεται και στη συζήτηση [https://github.com/energia/Energia/issues/249] που γίνεται στην επίσημη σελίδα υποστήριξης του Energia στο GitHub.

Με την sprintf() καταχωρούμε στη μεταβλητή macAddr τις τιμές που παίρνουμε από τον πίνακα της μεταβλητής mac που είναι σε 16αδική μορφή και διαμορφώνουμε κατάλληλα τις τιμές του ώστε να έχουν τη μορφή της διεύθυνσης mac που χρειαζόμαστε για να εισάγουμε στο μήνυμα προς το server.

Στη συνέχεια καλούμε τη συνάρτηση createMsg() όπου γίνεται η σύνθεση του μηνύματος σε μορφή JSON που θα σταλεί στο server. Πρώτα δημιουργείται και αποθηκεύεται στη μεταβλητή sendJson ένα αντικείμενο τύπου JSON μέσω του API της βιβλιοθήκης aJSON. Στο αντικείμενο αυτό θα πρέπει να προσθέσουμε όλες τις τιμές που χρειάζεται να στείλουμε, γι' αυτό για να προσθέσουμε μια τιμή θα πρέπει να γίνει έλεγχος αν πραγματικά έχει δημιουργηθεί το αντικείμενο. Αν όχι τότε τυπώνεται στην κονσόλα ελέγχου και αποσφαλμάτωσης ένα μήνυμα "JSON object not created" και ενημερώνει τον προγραμματιστή κατά τη διάρκεια της αποσφαλμάτωσης.



Εικόνα 4 3 Περιβάλλον ανάπτυξης Energia 14.

Έπειτα προσθέτουμε στο αντικείμενο sendJson την πρώτη τιμή που είναι τύπου stringκαι σύμφωνα με το API γίνεται με τη υποσυνάρτηση addStringToObject("JSON Object", "string", "string") που δέχεται τρία ορίσματα. Το πρώτο είναι το αντικείμενο JSON, το δεύτερο είναι το

κείμενο της ετικέτας και το τελευταίο είναι το κείμενο με την τιμή που θα έχει τη mac address του CC3100 από τη μεταβλητή macAddr. Προσθέτουμε και την επόμενη τιμή στο sendJson που είναι ίδιου τύπου και δείχνει το είδος του αισθητήρα που χρησιμοποιούμε, “ControlSensor”.

Αποθηκεύουμε τις τιμές υγρασίας και θερμοκρασίας από τον αισθητήρα DHT11 στις αντίστοιχες μεταβλητές που είναι τύπου integer και κάνουμε έλεγχο αν ήταν επιτυχής η ανάγνωση από τον αισθητήρα. Σύμφωνα με τη βιβλιοθήκη του αισθητήρα όταν δε γίνεται σωστά η ανάκτηση των τιμών ή υπάρχει κάποιο σφάλμα κατά την ανάγνωση από αυτόν τότε επιστρέφονται οι τιμές μηδέν 0 και για τη θερμοκρασία αλλά και για την υγρασία. Αυτό ως αποτέλεσμα είναι μη αποδεκτό και από το datasheet του DHT11 [http://www.micropik.com/PDF/dht11.pdf], όπου για 0^oC η ελάχιστη υγρασία που μπορεί να διαβαστεί είναι 30% H. Άρα όταν και οι δύο τιμές είναι μηδέν τότε υπάρχει σφάλμα ή δεν κάνουνε καλή επαφή τα pinσύνδεσης και πρέπει να ελεγχθούν. Τότε εμφανίζεται σχετικό μήνυμα στην κονσόλα αποσφαλμάτωσης. Οι τιμές που θα προστεθούν στο sendJson είναι οι τιμές -100, και για τη θερμοκρασία αλλά και για την υγρασία, όπως έχει συμφωνηθεί για την αναγνώρισή τους από το server ως πρόβλημα στον αισθητήρα.

Οι τιμές που θα προστεθούν είναι τύπου integer και γι’ αυτό θα χρησιμοποιήσουμε την υποσυνάρτηση addNumberToObject(“object”, “string”, “int”), που δέχεται τρία ορίσματα. Πρώτα το αντικείμενο μετά το κείμενο της ετικέτας και τρίτο την τιμή που θέλουμε να προσθέσουμε. Αυτή μπορεί να πάρει τιμές τύπου integer αλλά και πραγματικές τιμές τύπου float.

Αν η ανάγνωση των τιμών από τον αισθητήρα γίνει επιτυχώς και δεν έχουν κάποιο σφάλμα τότε τις προσθέτουμε στο αντικείμενο sendJson με την ετικέτα Humidityκαι την τιμή της μεταβλητής h που διαβάστηκε από τον αισθητήρα υγρασίας του DHT11 και με ετικέτα Temperatureκαι την τιμή της t που πήραμε από τον αισθητήρα θερμοκρασίας του DHT11.

Στη συνέχεια μέσω της συνάρτησης readOWTemperature() θα διαβάσουμε την τιμή της θερμοκρασίας του ψυκτικού υγρού από τον αισθητήρα DS18B20 μέσω της βιβλιοθήκης του OneWire που χρησιμοποιείται από τον συγκεκριμένο αισθητήρα. Κατά τη διάρκεια ανάγνωσης του πίνακα θερμοκρασιών του αισθητήρα αποθηκεύεται μέσω της συνάρτησης readOWTemperature() στη global μεταβλητή ctη τιμή της θερμοκρασίας που κατέγραψε ο αισθητήρας. Αμέσως μετά προστίθεται στο αντικείμενο sendJson με ετικέτα CoolantTemp η τιμή της θερμοκρασίας από τη μεταβλητή ct.

Οι επόμενες τιμές με βάση το μήνυμα JSON που πρέπει να δημιουργηθεί θα είναι τύπου Boolean. Θα διαβαστεί η τρέχουσα κατάσταση του pin ONOFF (p3_6) μέσω της συνάρτησης digitalRead() και αν έχει τάση από 3.3v έως 5v τότε η μεταβλητή onoff θα γίνει true ενώ αν εχει τάση μικρότερη από 3.3v τότε θα γίνει false. Η τιμή αυτή θα προστεθεί στο αντικείμενο sendJson με την υποσυνάρτηση addBooleanToObject(“object”,“string”, “boolean”), που δέχεται τρία ορίσματα. Το πρώτο είναι το αντικείμενο JSON, το δεύτερο η ετικέτα και το τρίτο είναι μια τιμή τύπου Boolean, δηλαδή true, false ή 1, 0 και θα προστεθεί στο sendJson με ετικέτα “OnOff” η τιμή που θα έχει η αντίστοιχη μεταβλητή onoff. Έπειτα θα διαβαστεί και η κατάσταση του

FAN_PIN (p7_4) με τη συνάρτηση digitalRead() και θα αποθηκευτεί η τιμή της στη μεταβλητή hilow και θα προστεθεί στο αντικείμενο sendJson με ετικέτα “HiLow”.

Τέλος επειδή χρειαζόμαστε έναν τρόπο για την επαλήθευση με το server πως δεν χάνονται τα μηνύματα που θα στέλνουμε αλλά και για να κρατάμε στο log του server το πλήθος τους, προσθέτουμε με τη μεταβλητή msgCount και ετικέτα “msgCount” στο μήνυμα τον αριθμό που το συνοδεύει καθώς αυξάνεται κάθε φορά που στέλνουμε ένα καινούριο μήνυμα. Μόλις τελειώσει η διαδικασία της δημιουργίας του αντικειμένου JSON μέσω της συνάρτησης createMsg() τότε το μετατρέπουμε σε string και το αποθηκεύουμε στη μεταβλητή sendstring. Αμέσως μετά γίνεται αποστολή του μηνύματος στο server και τυπώνεται στο τερματικό το μήνυμα “Send: “ και η τιμή της μεταβλητής sendstring που αποτελεί το μήνυμα σε μορφή JSON που στάλθηκε στο server. Αφού η διαδικασία αποστολής έχει τελειώσει τότε είναι απαραίτητο να ελευθερώσουμε τη μεταβλητή sendstring με την εντολή free(sendstring) και να απελευθερώσουμε τη δεσμευμένη μνήμη για το αντικείμενο JSON, που δημιουργήσαμε, όπως είναι απαραίτητο σύμφωνα με το documentation της βιβλιοθήκης [https://github.com/interactive-matter/aJson/blob/master/README.md] , καθώς η μνήμη που χρειάζεται και δεσμεύεται για τη δημιουργία του αντικειμένου είναι τις περισσότερες φορές όλη η διαθέσιμη μνήμη που διαθέτει το LaunchPad.

Μετά ενημερώνεται η μεταβλητή time4update και γίνεται false ώστε να μη σταλεί ξανά μήνυμα εάν δεν έχει περάσει ο απαιτούμενος χρόνος. Αποθηκεύεται στη μεταβλητή elapsedtime η χρονική στιγμή που στάλθηκε το μήνυμα από τη συνάρτηση millis(), που χρησιμοποιεί τον timer του LaunchPad και μετράει το χρόνο που έχει περάσει από τη στιγμή που ξεκινάει να τροφοδοτείται με ρεύμα η συσκευή.

Μόλις τελειώσει ο έλεγχος για την αποστολή του μηνύματος τότε γίνεται έλεγχος στο buffer του δικτύου του CC3100 αν έχει σταλεί κάποιο μήνυμα από το server. Τότε αρχικοποιείται και καθαρίζεται, από τα προηγούμενα δεδομένα που έχει, ένας πίνακας χαρακτήρων 512 θέσεων για την αποθήκευση του εισερχόμενου μηνύματος και διαβάζεται η ροή δεδομένων μέσω της υποσυνάρτησης read της βιβλιοθήκης WiFi που δέχεται δυο ορίσματα, το πρώτο είναι η μεταβλητή όπου θα αποθηκευτούν τα εισερχόμενα δεδομένα και το δεύτερο είναι μια μεταβλητή τύπου Boolean. Αν είναι truetότε αυτό σημαίνει πως υπάρχουν ακόμη δεδομένα προς ανάγνωση στο stream και συνεχίζεται το διάβασμά τους, αλλιώς αν είναι false τότε σημαίνει πως τα εισερχόμενα δεδομένα έχουνε τελειώσει και τερματίζεται η ανάγνωση του stream. Τυπώνεται στην κονσόλα το μήνυμα “Received: “ και η τιμή της μεταβλητής buffer με τα δεδομένα του μηνύματος. Μετατρέπουμε το μήνυμα σε αντικείμενο JSON και το αποθηκεύουμε στη μεταβλητή recvJson. Καλούμε τη συνάρτηση serverMsg() όπου θα γίνει ανάλυση του αντικειμένου recvJson και θα ανακτηθούν οι τιμές με τα δεδομένα εντολών από το server.

Αρχικά γίνεται ο απαραίτητος έλεγχος αν το αντικείμενο που δημιουργήθηκε δεν έχει κάποιο σφάλμα και είναι έγκυρος τύπος JSON. Δημιουργείται η μεταβλητή check και παίρνει την τιμή από το αντικείμενο με ετικέτα “Check” που είναι τύπου Boolean. Ελέγχεται αν υπάρχει στα περιεχόμενα του μηνύματος ετικέτα με όνομα “Check” και ταυτόχρονα ελέγχεται αν η τιμή που έχει είναι αληθής. Τότε δημιουργείται η μεταβλητή msgCnt και αποθηκεύει την τιμή με ετικέτα “msgCount” που υπάρχει στο αντικείμενο. Έχει συμφωνηθεί μετά από την παραλαβή

μηνύματος ο server να απαντάει με το “Check” και την τιμή του msgCount που δέχεται από το node ώστε να γίνεται επαλήθευση για τυχόν μηνύματα που έχουνε χαθεί. Έτσι γίνεται ο έλεγχος αν η τιμή της μεταβλητής msgCnt είναι η ίδια με την προηγούμενη τιμή της τοπικής μεταβλητής msgCount τότε το μήνυμα που έχει φτάσει στο server είναι το αμέσως επόμενο χωρίς να έχει χαθεί κάποιο. Μετά τυπώνεται στην κονσόλα το μήνυμα “ OK “ και ανάβει το πράσινο LED του LaunchPad για να ενημερώσει οπτικά το χρήστη πως δεν υπάρχει κάποιο σφάλμα στην αποστολή των μηνυμάτων, ενώ στην αντίθετη περίπτωση, αν η τιμή της msgCnt δεν είναι η ίδια με την προηγούμενη τιμή της msgCount τότε ανάβει το κόκκινο LED του LaunchPad και το πράσινο σβήνει για να ενημερωθεί οπτικά και ο διαχειριστής για την κατάσταση των μηνυμάτων που έχουν χαθεί.

Στην περίπτωση που δεν υπάρχει στο μήνυμα που έχει διαβαστεί η ετικέτα “Check” ή έχει την τιμή false, τότε πρόκειται για ένα μήνυμα εντολών. Σε αυτή την περίπτωση είναι απαραίτητο να δημιουργηθούν οι μεταβλητές ελέγχου, on, fan(και ws), που θα κρατάνε τις τιμές από τις αντίστοιχες ετικέτες του αντικειμένου recvJson, τις OnOff, HiLow (και WinterSummer).

Έπειτα γίνεται ο έλεγχος αν υπάρχουν οι τιμές στο αντικείμενο και στην περίπτωση που κάποια από αυτές έχει την τιμή true ή 1 τότε το αντίστοιχο pin γίνεται HIGH, δηλαδή τροφοδοτείται με τάση από 3.3v – 5v ενώ αν είναι false ή 0 τότε γίνεται LOW, δηλαδή η τάση στο συγκεκριμένο pin γίνεται μικρότερη από 3.3v ή ακριβώς 0. Η μεταβλητή time4update γίνεται true διότι έπειτα από ένα μήνυμα εντολής προς το node η κατάσταση των pin θα έχει αλλάξει και γι‘ αυτό είναι απαραίτητο να αποσταλεί εκ νέου μήνυμα στο server με τις νέες τιμές των διακοπτών και την τρέχουσα κατάσταση του node. Τέλος ελευθερώνουμε το αντικείμενο recvJson και το διαγράφουμε αποδεσμεύοντας έτσι τη μνήμη που χρησιμοποιεί και επιστρέφουμε στην εκτέλεση του κύριου βρόχου.

Εν συνεχεία στον κύριο βρόχο του προγράμματος ελέγχεται η περίπτωση που η σύνδεση με το server έχει τερματιστεί. Τότε στη μεταβλητή noConnectionTime αποθηκεύεται μέσω της συνάρτησης millis() η χρονική στιγμή που χάνεται η σύνδεση και τυπώνεται στην κονσόλα το μήνυμα “Connection with server lost.” και το μήνυμα για την προσπάθεια επανασύνδεσης “Attempting to reconnect to server”. Αρχικοποιείται η μεταβλητή tries = 0 και όσο δεν υπάρχει σύνδεση τυπώνονται τελείες « . » στην κονσόλα και γίνεται έλεγχος αν οι αποτυχημένες προσπάθειες σύνδεσης ξεπεράσουν τις εκατό τότε τυπώνεται μήνυμα στην κονσόλα ότι ο server δεν αποκρίνεται, η μεταβλητή tries ξανά αρχικοποιείται σε 0, ανάβει το κόκκινο LED του LaunchPad και το πράσινο σβήνει για να ενημερωθεί οπτικά και ο διαχειριστής για το σφάλμα και να γίνει έλεγχος της επικοινωνίας και στο server.

Μετά από κάθε έλεγχο υπάρχει μια χρονοκαθυστέρηση 1 sec και έπειτα γίνεται και ο έλεγχος για το χρόνο που έχει χαθεί η σύνδεση με το server. Αν είναι περισσότερο χρόνο από το χρόνο που έχουμε ορίσει στη σταθερή μεταβλητή FAILSAFEMINS αποσυνδεμένο το node τότε μπαίνει σε κατάσταση failsafe καλώντας την αντίστοιχη συνάρτηση failsafe() που θέτει όλα τα pin ONOFF_PIN, FAN_PIN, (WINSUM_PIN) σε HIGH. Έτσι δίνεται η εντολή στο relay να σταματήσει τη λειτουργία της κλιματιστικής μονάδας, τυπώνεται στην κονσόλα το μήνυμα “Failsafe Mode” και επιστρέφει στο βρόχο ελέγχου για την επανασύνδεση με το server.

Τέλος στην περίπτωση όπου δεν υπάρχει σύνδεση με το ασύρματο δίκτυο, δίνεται η εντολή για επανεκκίνηση του node μηδενίζοντας τον watchdog time. Έτσι θα γίνει ξανά προσπάθεια σύνδεσης από τη συνάρτηση Setup() με το Access Point. Η τελευταία εντολή του κύριου βρόχου loop() ελέγχει συνεχώς μέσω της συνάρτησης millis() το χρονικό διάστημα που έχει περάσει από το τελευταίο μήνυμα που στάλθηκε στο server και αν είναι περισσότερο από το χρόνο που έχουμε ορίσει στη σταθερή μεταβλητή MINS, τότε η μεταβλητή time4update γίνεται αληθής ώστε να γίνει αποστολή καινούριου μηνύματος στο server.

Η συνάρτηση printWifiStatus() είναι αυτούσια από το παράδειγμα που παρέχεται στο sketch του Energia 14, το ClientSendReceiveString.ino. Στο συγκεκριμένο παράδειγμα χρησιμοποιείται μόνο για την εμφάνιση μηνυμάτων στην κονσόλα που αφορούν στο όνομα του δικτύου που είναι συνδεδεμένο το node, τη διεύθυνση IP που έχει πάρει από τον DHCP server του δικτύου και την ισχύ του ασύρματου σήματος. Προς το παρόν μέχρι την έκδοση 14 του Energia η βιβλιοθήκη WiFi δεν έχει ολοκληρωθεί. Η συνάρτηση που επιστρέφει την ισχύ του ασύρματου σήματος επιστρέφει λανθασμένα την τιμή 0. Πλέον από την έκδοση 16 του Energia έχει διορθωθεί η βιβλιοθήκη WiFi και εμφανίζεται σωστά. Για να πετύχουμε το ίδιο αποτέλεσμα και στο Energia 14 πρέπει να ανανεώσουμε τον κώδικα του αρχείου WiFi.cpp που βρίσκεται στο φάκελο με τις βιβλιοθήκες στη θέση …Energia-0101E0014/hardware/msp430/Libraries/WiFi. Μπορούμε να κατεβάσουμε και να αντικαταστήσουμε το αρχείο WiFi.cpp με τον ανανεωμένο κώδικα από το επίσημο αποθετήριο του Energia στο GitHub [https://github.com/energia/Energia/tree/master/hardware/msp430/libraries/WiFi].

Η συνάρτηση deviceStatus() χρησιμοποιήθηκε αποκλειστικά για σκοπούς καλύτερης εκμάθησης και εξοικείωσης με τη συνάρτηση digitalRead() και δε χρησιμοποιείται στο πρόγραμμα.

Οι συναρτήσεις createMsg() και serverMsg() χρησιμοποιούνται για τη δημιουργία του απαιτούμενου μηνύματος σε μορφή JSON και την ανάλυση του εισερχόμενου μηνύματος από το server αντίστοιχα, όπως αναλύθηκε η λειτουργία τους προηγουμένως.

Τέλος η συνάρτηση readOWTemperature() χρησιμοποιείται για να διαβάσει τη θερμοκρασία από τον αισθητήρα DS18B20 μέσω του πρωτοκόλλου OneWire και επιστρέφει την τιμή της θερμοκρασίας μέσω της global μεταβλητής ct. Μέσα από τη συνάρτηση αυτή γίνεται κλήση και στις δύο τελευταίες συναρτήσεις, τις saveTemperature(), readOW() και findOW(), που χρησιμοποιήθηκαν αυτούσιες από το παράδειγμα της βιβλιοθήκης του αισθητήρα. Η πρώτη συνάρτηση χρησιμοποιείται για την αποθήκευση και τη μετατροπή της τιμής της θερμοκρασίας σε δεκαδική μορφή και με ακρίβεια 12 bit. Η readOW() χρησιμοποιείται για την ανάγνωση του πίνακα τιμών από το scratchpad του αισθητήρα και η findOW() που κάνει αναζήτηση στον πίνακα του αισθητήρα, διαβάζει και τυπώνει στην κονσόλα το μοντέλο που στο συγκεκριμένο παράδειγμα είναι ο DS18B20, δηλαδή ο αισθητήρας με κωδικό 28.

Κατά τη διάρκεια των δοκιμών για την ανάκτηση της θερμοκρασίας παρατηρήθηκε πως οι τιμές που παίρνουμε από τον αισθητήρα είναι ακέραιοι αριθμοί. Η ανάλυση της θερμοκρασίας που κάνουμε ανάκτηση από τον αισθητήρα είναι με ακρίβεια 12bit, συνεπώς θα έπρεπε να έχουμε πιο ακριβή αποτελέσματα. Επίσης οι τιμές που μπορούμε να διαβάσουμε είναι μόνο για

θερμοκρασία από μηδέν βαθμούς κελσίου και πάνω καθώς χρειάζεται τροποποίηση στη βιβλιοθήκη OneWire ώστε να διαβάζει και να μετατρέπει σωστά τους βαθμούς θερμοκρασίας υπό του μηδενός.

Κεφάλαιο 5 – Προτάσεις και ιδέες

5. Αποτέλεσμα

Στόχος του projectήταν ο προγραμματισμός του μικροελεγκτή της Texas Instruments, του MSP430F5529 μαζί με το boosterpack CC3100 για την ασύρματη σύνδεση και τη διαχείριση των κλιματιστικών μονάδων του κτηρίου, ώστε να γίνεται εξοικονόμηση ενέργειας, καλύτερη διαχείριση όσον αφορά στην κατανάλωση ρεύματος ή πετρελαίου για την ψύξη ή τη θέρμανση με όσο το δυνατόν λιγότερες επιπτώσεις για το περιβάλλον και καλύτερη οικονομική διαχείριση.

Τα αποτελέσματα από το τελικό προϊόν είναι ικανοποιητικά αφού πετυχαίνουμε το στόχο που είχαμε θέσει αρχικά για τη δημιουργία του όλου project. Παρόλα αυτά υπάρχει η δυνατότητα για τη βελτίωσή του αλλά και διάφορες επιπλέον προτάσεις για παρόμοια projectπου βασίζονται πάνω σε αυτό.

5.1 Εναλλακτικό Hardware

Αρχικά θα πρέπει να γίνει αναφορά και στα υπόλοιπα LaunchPad που προσφέρουν παρόμοιες αλλά και πιο εξελιγμένες δυνατότητες όπως είναι το πιο καινούριο LaunchPad της σειράς CC3X00 της Texas Instruments, το CC3200. Ο συγκεκριμένος μικροελεγκτής έχει σχεδιαστεί με βάση το Internet of Things και είναι από τους πρώτους που ενσωματώνει τη δυνατότητα ασύρματης σύνδεσης στο ίδιο chip. Επιπλέον ενσωματώνει και έναν επεξεργαστή ARM Cortex M4 που τρέχει στα 80MHz με αποτέλεσμα να παρέχει τη δυνατότητα στους προγραμματιστές να τρέχουν ολοκληρωμένες εφαρμογές μέσα από ένα μόνο ηλεκτρονικό κύκλωμα. Έτσι έχει τη δυνατότητα να τρέχει το ειδικά αναπτυγμένο ενσωματωμένο λειτουργικό σύστημα της Texas Instruments, το TI-RTOS αλλά και το ανοιχτού κώδικα αντίστοιχο λειτουργικό σύστημα, το FreeRTOS.

Υποστηρίζει ακόμη και εγγενώς στο υλικό του chipτις μηχανές κρυπτογράφησης, AES, DESκαι 3DES, SHA2 και MD5 αλλά και CRC και Checksum. Έτσι μπορεί να συνδεθεί και να επικοινωνήσει με τις αντίστοιχες υπηρεσίες και τους servers που απαιτούν τα συγκεκριμένα πρωτόκολλα κρυπτογράφησης ώστε να γίνει η απαραίτητη ανταλλαγή δεδομένων με ασφάλεια και χωρίς να χρειάζεται κάποια επιπλέον βιβλιοθήκη για να το πετύχουμε αυτό.

Ένα άλλο παρόμοιο LaunchPad έχει δημιουργηθεί με βάση το CC3200 από την RedBearLab και υποστηρίζεται είδη από το Energia, το GCC, IAR System και το Code Composer Studiotης Texas Instruments για την ανάπτυξη του firmware σε αυτό. Επιπλέον ενσωματώνει κάποια χαρακτηριστικά ακόμη εκτός από αυτά που προσφέρει το αντίστοιχο LaunchPad της Texas Instruments. Έχει τη δυνατότητα να συνδέεται και να αλληλοεπιδρά με τα περισσότερα shieldπου είναι κατασκευασμένα για το Arduino, μπορεί να επαναπρογραμματίζεται χωρίς την ανάγκη για αλλαγή του jumper όπως γίνεται με το πρωτότυπο CC3200 και μπορεί να συνδεθεί και με τις υπόλοιπες πλακέτες που προσφέρονται από την ίδια εταιρία για να του προσθέσουν και επιπλέον συνδεσιμότητα με συσκευές Bluetooth.

Ακόμη ένα chip που ενσωματώνει ίδιες λειτουργίες με το παραπάνω είναι το ESP8266 [http://www.esp8266.com/]. Είναι και αυτό ένα ολοκληρωμένο σύστημα που περιλαμβάνει ασύρματη συνδεσιμότητα στο ίδιο chip. Οι δυνατότητές του είναι λίγο πιο περιορισμένες καθώς δεν υποστηρίζει τις μηχανές κρυπτογράφησης αλλά είναι ικανό να προσφέρει την απαραίτητη συνδεσιμότητα που χρειάζεται και λόγο του μεγέθους του αλλά και των δυνατοτήτων του κατέχει και αυτό μια θέση ανάμεσα στις συσκευές που είναι έτοιμες για το Internet of Things.

5.2 Ιδέες για μελλοντικά project

Το παραπάνω projectαποτελεί μια από τις πολλές εφαρμογές που μπορεί να έχει μια αντίστοιχη συσκευή, όμως δε σημαίνει πως δεν επιδέχεται περεταίρω βελτιώσεις ή αναβαθμίσεις. Για το λόγο αυτό και το ότι τέτοιου είδους συσκευές μπορεί να τοποθετηθούν σε συσκευές ή σημεία όπου θα είναι δύσκολη η αποσύνδεσή τους ώστε να γίνει η αναβάθμιση του πηγαίου κώδικα του LaunchPad, η Texas Instruments έχει δημιουργήσει το Host Prog add-on [http://processors.wiki.ti.com/index.php/CC31xx_Host_Programming_Application]. Αυτή τη στιγμή βρίσκεται σε αρχικό στάδιο για τον προγραμματισμό OTA Over The Air πραγματοποιόντας αναβάθμιση μόνο του firmware του CC3100. Η εγγραφή και μεταφορά όμως των δεδομένων γίνεται απευθείας στη μνήμη flash της συσκευής με αποτέλεσμα αργότερα να είναι αρκετά πιο εύκολη η αναβάθμιση των LaunchPad αφού θα γίνεται ασύρματα και πιθανώς αυτόματα μέσω του κεντρικού serverπου θα κάνει τη διαχείριση των κόμβων αλλά και σε πολλές συσκευές ταυτόχρονα.

Ακόμη ένας τομέας βελτίωσης είναι η δημιουργία ενός τυπωμένου κυκλώματος, PCB Printed Circuit Board, όπου ολόκληρη η συνδεσμολογία που απαιτείται να υπάρχει έτοιμη προς σύνδεση ως ένα επιπλέον Booster Pack. Όλοι οι απαραίτητοι αισθητήρες, οι αντιστάσεις και τα relayθα είναι έτοιμα και λειτουργικά με μια απλή τοποθέτηση χωρίς να χρειάζονται επιπλέον γνώσεις από τον απλό χρήστη για την τοποθέτηση του κόμβου.

Επιπλέον ένα χαρακτηριστικό που είναι σημαντικό για τους κόμβους που τοποθετούνται σε βιομηχανικά περιβάλλοντα, πόλεις ή μέρη όπου η απευθείας σύνδεση με το ασύρματο δίκτυο δεν είναι εφικτή λόγω παρεμβολών μηχανημάτων, κτηρίων ή μεγάλης απόστασης από τα Access Point είναι η δυνατότητα της σύνδεσης και της επαναπροώθησης των μηνυμάτων προς το server μέσω ενός P2P – Multihop δικτύου. Η δυνατότητα σύνδεσης P2P υποστηρίζεται από το CC3100 και παρέχεται παράδειγμα για τη σύνδεση δυο συσκευών μεταξύ τους από την Texas Instruments καθώς και ο πηγαίος κώδικας [http://processors.wiki.ti.com/index.php/CC31xx_P2P_Application].

Επίσης η δημιουργία μιας εφαρμογής για τη διαχείριση των κόμβων μέσω μιας έξυπνης συσκευής, για iOS, Android και Windows Phone θα διευκολύνει τον διαχειριστή καθώς δε θα είναι αναγκασμένος για κάθε αλλαγή που θα πρέπει να κάνει να βρίσκεται σε κάποιον υπολογιστή με πρόσβαση στο δίκτυο αλλά να μπορεί να επιβλέπει και να πραγματοποιεί τις επιθυμητές αλλαγές με ευκολία οπουδήποτε και να βρίσκεται.

Εκτός από τη βελτίωση του συγκεκριμένου project μπορούν να υλοποιηθούν και άλλα projectώστε να καλύπτουν και άλλες ανάγκες. Υπάρχει η δυνατότητα για την ανάπτυξη ενός αυτόνομου κόμβου ελέγχου για μεμονωμένες κλιματιστικές μονάδες όπου δεν υπάρχει η ανάγκη για κεντρική και μαζική διαχείριση αυτών. Παραδείγματα ελέγχου της συσκευής μέσω του ενσωματωμένου webserver στο CC3100 BoosterPack παρέχονται [http://processors.wiki.ti.com/index.php/CC31xx_HTTP_Server] από την Texas Instruments με οδηγίες αλλά και όλο τον πηγαίο κώδικα που χρειάζεται για να τροποποιηθεί και να έχουμε το επιθυμητό αποτέλεσμα.

Βέβαια καθώς ο αριθμός των συνδεδεμένων συσκευών θα αυξάνεται κρίνεται απαραίτητο να μελετηθεί το θέμα του scalability. Οι χρόνοι απόκρισης των κόμβων ίσως αρχίσουν να αυξάνονται λόγω της διακίνησης μεγάλου όγκου δεδομένων μέσα στο δίκτυο κάτι που επηρεάζει τις υποδομές που έχουν σχεδιαστεί να δουλεύουν και να ανταποκρίνονται με μεγάλη ακρίβεια χρόνου. Για αυτό το λόγο θα πρέπει να γίνουν διάφοροι έλεγχοι και δοκιμές στο πρωτόκολλο που χρησιμοποιείται για την ανταλλαγή των μηνυμάτων με το server. Τα πρωτόκολλα που χρησιμοποιούνται σε τέτοιου είδους εφαρμογές, όπου υπάρχει μεγάλος αριθμός συσκευών που ανταλλάσσει μηνύματα ταυτόχρονα, είναι το MQTT, AMQP, XMPP, HTTP/2, WebSockets, DDS, CoAP, και STOMP. Τα πιο διαδεδομένα είναι τα τρία πρώτα και χρησιμοποιούνται είδη από έτοιμα παραδείγματα της Texas Instruments για το MQTT και το XMPP καθώς παρέχεται και ο πηγαίος κώδικας για κάθε ένα από αυτά. Σε ένα πιο σύνθετο και μεγάλο περιβάλλον, όπως είναι μια έξυπνη πόλη η ένα μεγάλου μεγέθους εργοστάσιο είναι δυνατή η χρήση περισσοτέρων από ένα πρωτόκολλο με στόχο την αποφυγή bottleneck στο δίκτυο. Στο συγκεκριμένο projectχρησιμοποιείται το πρωτόκολλο TCP και μέσω αυτού στέλνουμε ένα μήνυμα σε μορφή JSON στο server. Κατά τη διάρκεια των δοκιμών που έγιναν με τέσσερα nodeταυτόχρονα συνδεδεμένα και πλήρως λειτουργικά δεν παρατηρήθηκε κάποια καθυστέρηση όσον αφορά στην απόκρισή τους, όμως καθώς ο αριθμός τους θα αυξάνεται κρίνεται απαραίτητη η αλλαγή του πρωτοκόλλου επικοινωνίας με το server και προτείνεται η χρήση του MQTT ή του XMPP καθώς είναι σχεδιασμένα για αυτό το σκοπό.

Βιβλιογραφία

blueapp.io (2015) “Smart HVAC Controls: Energy Efficient Solutions for Heating and Cooling”. [Online] Available athttp://blueapp.io/blog/smart-hvac-controls-energy-efficient-solutions-for-heating-and-cooling/ (last accessed: 13 March 2016)

Callahan, K., (2015) “Building automation systems can help to manage the wide array of physical environments found within a single health facility” [Online] Available at: http://www.hfmmagazine.com/display/HFM-news-article.dhtml?dcrPath=/templatedata/HF_Common/NewsArticle/data/HFM/Magazine/2015/Nov/marketplace-building-automation-systems

Heerwagen, J., (2000) Building Research and Information 28 (5/6), 353-367

N.R.D.C. (2015), “SMARTER BUSINESS: GREENING ADVISOR”. [Online] Available at: http://www.nrdc.org/enterprise/greeningadvisor/cri-hvac.asp (last accessed: 06 February 2016)

Raji, R., (1994) “Smart networks for Control”, IEEE Spectrum

Weisser, M., (1991) “The computer of the 21st century”, Scientific American 265 (3), pp 94 - 104

Wills, B., (2014), “TURN OVER A NEW LEAF IN 2015”. [Online] Available at: http://greenenterprise.ca/GEM/turn-over-a-new-leaf-in-2015/ (last accessed: 06 February 2016)

http://energia.nu

http://www.ti.com/tool/ccstudio

http://www.ti.com/tool/WIFISTARTER

http://software-dl.ti.com.ecs/cc31xx/APIs/public/cc31xx_simplelink/latest/htmlindex.html

https://github.com/energia/Energia/issues/249

http://www.micropik.com/PDF/dht11.pdf

https://github.com/interactive-matter/aJson/blob/master/README.md

https://github.com/energia/Energia/tree/master/hardware/msp430/libraries/WiFi

http://www.esp8266.com/

Βιβλιογραφία Εικόνων

Εικόνα 1.1 Internet of Things [Online] available at: http://www.3g.co.uk/g_phones/large/internet-of-things-everything-you-need-to-know.jpg

Εικόνα 2.1 Αδιάβροχος αισθητήρας θερμοκρασίας DS18B20 [Online] available at: http://cm.lnwfile.com/_/cm/_raw/rk/jy/hs.jpg

Εικόνα 3.1 Αδιάβροχος αισθητήρας θερμοκρασίας DS18B20 [Online] available at: http://cm.lnwfile.com/_/cm/_raw/rk/jy/hs.jpg

Εικόνα 3.2 Ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας DHT11 [Online] available at: http://cmostronics.in/images/Temperature%20and%20Humidity%20Sensor%20DHT11.JPG

Εικόνα 3.3 Ηλεκτρομαγνητικός διακόπτης 2 καναλιών 5 volt[Online] available at:http://ecx.images-amazon.com/images/I/51X0I2j5FFL._SY300_.jpg

Εικόνα 3.4 Texas Instruments MSP4430F5529 LaunchPad pin numbers diagram [Online] available at:http://energia.nu/img/LaunchPadMSP430F5529.jpg

Εικόνα 4.1 Περιβάλλοντα ανάπτυξης κώδικα [Online] available at:http://www.ti.com/ww/en/launchpad/img/launchpad_software_bubbles.png

Πηγαίος Κώδικας

Energia 14 IDE – http://energia.nu

Windows 7 Folder Path : C:\ProgramFiles\energia-0101E0014

Libraries Path : C:\ProgramFiles\energia-0101E0014\hardware\msp430\libraries

CC3100HVAC.ino

```
/* ThisexampledemonstratesusingtheMSP-EXP430F5529LPandthecc3100asawificlient (TCP) Theclientwillconnect to
the TCP server. Reading the temperature and humidity from DHT11 sensor and the temperature from DS18B20
sensor and send the data in JSON format to the tcp server. The server sends commands in JSON format */

#ifndef __CC3200R1M1RGC__
// Do not include SPI for CC3200 LaunchPad
#include<SPI.h>// header file for SPI
#endif
#include<WiFi.h>// header file for the WiFi connectivity
#include<WiFiClient.h>// header file for the TCP client
#include"DHT.h"// header file for DHT sensors
#include<aJSON.h>// header file for creating and parsing JSON
#include<GFDS18B20.h>// header file for OneWire

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

char ssid[] = "cc3200demo; // your network name also called SSID
char password[] = "password";// your network password
int keyIndex = 0;                  // your network key Index number (needed only for WEP)
#define DHTPIN P3_7                // the data pin of the DHT11 connected to
#define ONOFF_PIN P3_6             // the pin controling the On/Off state of the fancoil
#define FAN_PIN P7_4               // the pin controling the Hi/Low state of the fancoil
#define WINSUM_PIN P8_2            // the pin controling the Winter/Sumer state of the fancoil
#define MINS 120000                // 120000 milisecons = 2 minutes to wait before sending the message
to the TCP server
#define FAILSAFEMINS 60000         // FAILSAFEMINS passed enter failsafe
#define OWPIN  P6_5                // 32 for OneWire
#define MAXOW 10                   // Max number of OW's used for OneWire
#define  SERVER  "m2m.ce.teiep.gr"// the server url
IPAddress serverIP(0, 0, 0, 0);   // global variable to store the resolved IP Address of the server
uint16_t port = 5020;              // port number of the server
unsignedlong elapsedtime;         // holds the last time a message send to the TCP server
unsignedlong noConnectionTime;    // holds the last time the connection with the TCP server lost
byte ROMarray[MAXOW][8];           // the array used in OneWire library
byte ROMtype[MAXOW];               // 28 for temp', 12 for switch etc.
byte ROMtemp[MAXOW];               // used in OneWire library
byte result[MAXOW+5];              // used in OneWire library
byte data[12];                     // used in OneWire library
byte i;                            // used in OneWire library
byte addr[8];                      // used in OneWire library
uint8_t ROMmax=0;                  // used in OneWire library
uint8_t ROMcount=0;                // used in OneWire library
boolean foundOW =false;            // used in OneWire library
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
int dStatus;                       // global variable for the On/Off or Hi/Low status
byte mac[7];                       // byte array for storing the mac address
char macAddr[6];                   // global variable for storing the mac address
```

```
WiFiClient client;                  // the tcp client
boolean onoff = true;               // global variable to store the state of the pin
boolean hilow = true;               // global variable to store the state of the pin
boolean winsum = true;              // global variable to store the state of the pin
boolean time4update = true;         // global boolean variable changing according to MINS
byte ct;                            // global variable to store the temperature aquired from DS18B20
int msgCount = 0;                   // to count the messages we send
int conToServer;                    // if we are connected to server
DHT dht(DHTPIN, DHTTYPE);           // Initialize the DHT driver, currently on DHTPIN and DHTTYPE
DS18B20 ds(OWPIN);                  // Initialize the OneWire driver, currently on OWPIN

voidsetup() {                       // initial setup routine start

  pinMode(RED_LED, OUTPUT);         // configure Red Led as output
  pinMode(GREEN_LED, OUTPUT);       // configure Red Led as output
  digitalWrite(RED_LED, LOW);       // Turnoff Red Led
  digitalWrite(GREEN_LED, LOW);     // Turnoff Green Led
  pinMode(ONOFF_PIN, OUTPUT);       // configure ONOFF_PIN as output
  pinMode(FAN_PIN, OUTPUT);         // configure FAN_PIN as output
  pinMode(WINSUM_PIN, OUTPUT);      // configure WINSUM_PIN as output
  digitalWrite(ONOFF_PIN, HIGH);    // failsafe mode set to HIGH
  digitalWrite(FAN_PIN, HIGH);      // failsafe mode set to HIGH
  digitalWrite(WINSUM_PIN, HIGH);   // failsafe mode set to HIGH
  Serial.begin(9600);               //Initialize serial and wait for port to open:
  digitalWrite(RED_LED, HIGH);      // RED Led On to indicate configuration process
  digitalWrite(GREEN_LED, HIGH);   // GREEN Led On indicates configuration process
  Serial.println("Started SmartConfig Mode"); // print to the console that we starting in SmartConfig Mode
  WiFi.startSmartConfig();            // Start wifi in SmartConfig Mode
  Serial.println("\nYou're connected to the network"); // print in serial console that we have connected
to the AP
  Serial.println("Waiting for an ip address"); // print in serial console that we are waitting for an IP
address
while (WiFi.localIP() == INADDR_NONE) { // check if IP address aquired
    Serial.print(".");                  // print dots while we wait for an ip addresss
    delay(300);                         // wait 0.3 sec before checking the connection status
  }                                     // END - while (WiFi.localIP() == INADDR_NONE)

  Serial.println("\nIP Address obtained"); // print in serial console that the IP address aquired

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

  digitalWrite(RED_LED, LOW);           // configuration done turn off led
  digitalWrite(GREEN_LED, LOW);         // configuration done turn off led

int iRet;                             // iRet variable
while ((iRet = WiFi.hostByName(SERVER, serverIP)) < 0) { // Resolving TCP server url
if(iRet < 0) {                      // if host name lookup failed
      Serial.println("Host name lookup failed");  // print in serial console that the host name lookup
failed
      digitalWrite(RED_LED, HIGH);     // RED LED to indicate error to the user
      delay(2000);                     // wait 2 seconds to indicate the error with red led to the user
      digitalWrite(RED_LED, LOW);      // turn Red Led off
//WDTCTL = 0;                       // restart the device
    } else {                            // else if host name lookup succeded
      Serial.print("Server IP: ");     // print in serial console that message
      Serial.println(serverIP);        // print in serial the resolved IP for the TCP server
      Serial.flush();                  // Flushes the buffer of incoming serial data
    }                                   // END - if(iRet < 0)
  }                                     // END - while ((int iRet = WiFi.hostByName(SERVER, serverIP)) < 0)

  Serial.println("Attempting to connect to server");  // attempt to connect to the server
  uint8_t tries = 0;                                  // variable to store the number of tries for
connecting to the TCP server
while (client.connect(serverIP, port) == false) {   // check if we're connected with the TCP server
    Serial.print(".");                                  // print dots wile we're waitting for the connection
if (tries++ > 100) {                              // if we have over 100 failed tries to connect with the
TCP server
      Serial.println("\nThe server isn't responding");// print that message to the serial console
```

```
      tries = 0;                                        // zero tries variable
      digitalWrite(RED_LED, HIGH);                      // indicate connecton error with red led to the user
      delay(30000);                                     // wait 30 seconds
      digitalWrite(RED_LED, LOW);                       // turn off red led
    }                                                   // END - if (tries++ > 100)
    delay(1000);                                        // wait 1 sec before checking for the TCP server
connection status
  }                                                     // END - while (client.connect(serverIP, port) ==
false)

  Serial.println("Connected to the server!");          // we're connected to the TCP server by this point
//delay(500);
  findOW();                                             // find the OneWire sensor
//displayOW();                                        // print in serial console the OneWire sensor
  dht.begin();                                          // initialize the DHT11 sensor
char* fw = WiFi.firmwareVersion();                    // store in variable the current Firmware Version
  Serial.print("Firmware: ");                           // print in serial console that message
  Serial.println(fw);                                   // print in serial console the Firmware Verion
  elapsedtime = millis();                               // initialize the variable with current miliseconds
passed

}                                                       // END - Initial setup routine

voidloop() {                                           // START - Main Loop task

if ( WiFi.status() == WL_CONNECTED) {                 // check if we have connection with the AP
    conToServer = client.connected();                   // store to global variable the connection status
with the TCP server
if (conToServer) {                                  // check if we have connection with the TCP server
if (time4update) {                                 // check if MINS passed
//byte mac[6];
        WiFi.macAddress(mac);                           // read the mac address of the device ! Needed here
to run every time or else somehow overwritten
        sprintf(macAddr, "%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
        createMsg();                                      // call createMsg function for creating the JSON
object
        sendstring = aJson.print(sendJson);               // save the created JSON object as a string
        client.print(sendstring);                         // send the JSON string to the TCP server
//client.flush();
        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
        time4update = false;                              // make the update variable false
        elapsedtime = millis();                           // store the current time a message sent
      }                                                   // END - if (time4update)

if (client.available()) {                           // check if there is a message from the TCP server
char buffer[512] = {0};                            // initialize the char array to hold the message
        client.read((uint8_t*)buffer, client.available()); // Read the message from the TCP server
        Serial.print("Received: ");                       // print the message Received: in serial console
        Serial.println(buffer);                           // print the message received from the TCP server in
serial console
        recvJson = aJson.parse(buffer);                   // create a JSON object from the received message
        serverMsg();                                      // call serverMsg function to read the commands
received from the TCP server and apply the requested actions
    }                                                     // END - if (client.available())
   } else {                                               // else if the connection with the TCP server lost
     noConnectionTime = millis();                           // start counting the time with no connection
to server
     Serial.println("Connection with server lost");       // print in serial console Connection with
server lost
     Serial.println("Attempting to reconnect to server"); // print in serial console Attempting to
reconnect to server
     uint8_t tries = 0;                                     // initialize the tries var
while (client.connect(serverIP, port) == false) {     // check if we're connected with the TCP server
       Serial.print(".");                                   // print dots wile we're waitting for the
connection
```

```
if (tries++ > (FAILSAFEMINS/1000)) {               // if we have over 100 failed tries to connect with
the TCP server
         Serial.println("\nThe server isn't responding");  // print The server isn't responding to the
serial console
         tries = 0;
         digitalWrite(RED_LED, HIGH);
         digitalWrite(GREEN_LED, LOW);
       }                                                   // END - if (tries++ > 100)
       delay(1000);                                        // wait 1 sec before checking for the TCP
server connection status
if ((millis() - noConnectionTime) > FAILSAFEMINS ) failsafe(); // if FAILSAFEMINS passed then enter into
failsafe routine
     }                                                     // END - while (client.connect(serverIP, port)
== false)
     digitalWrite(RED_LED, LOW);                         // connected so red led off
   }                                                     // END - else if (conToServer)
  } else {                                               // when connection with AP lost

    WDTCTL = 0;                                          // zero out the WatchDog Timer so the device
restarts
  }                                                      // END - else if ( WiFi.status() ==
WL_CONNECTED)
if ((millis() - elapsedtime) > MINS) time4update = true; // check if MINS passed and change variable to
true to send a message to the TCP server
}                                                        // END - Main Loop Task

voidprintWifiStatus() {                  // 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

long 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()

voiddeviceStatus() {                      // for testing not used in the program
// LED Status
  dStatus = digitalRead(RED_LED);
  Serial.println(dStatus);
}

voidcreateMsg() {                                 // function for creating the JSON object

  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", macAddr); // Add the macaddress as NodeID
  aJson.addStringToObject(sendJson, "Type", "ControlSensor");// add the node type Sensor or ControlSensor
to the JSON object
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
int h = dht.readHumidity();                          // read and store the humidity from the sensor to
variable
int t = dht.readTemperature();                       // read and store the temperature from the sensor
to variable

if ((t == 0) & (h == 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
  } else {                                              // else if returns are valid
    aJson.addNumberToObject(sendJson, "Humidity", h);    // add to the JSON object the value from the
sensor
    aJson.addNumberToObject(sendJson, "Temperature", t); // add to the JSON object the value from the
sensor
  }                                                     // END - if (isnan(t) || isnan(h))

  readOWTemperature();                                      // Reading temperature from DS18B20 sensor
function and assign it to the global variable ct
  aJson.addNumberToObject(sendJson, "CoolantTemp", ct);      // add to the JSON object the value from the
sensor
  onoff = digitalRead(ONOFF_PIN);                           // check the state of the pin and assign the
value to onoff variable
  aJson.addBooleanToObject(sendJson, "OnOff", onoff);        // add the state of ONOFF_PIN to the JSON
object
  hilow = digitalRead(FAN_PIN);                             // check the state of the pin and assign the
value to hilow variable
  aJson.addBooleanToObject(sendJson, "HiLow", hilow);        // add the state of FAN_PIN to the JSON
object
  winsum = digitalRead(WINSUM_PIN);                         // check the state of the pin and assign the
value to onoff variable
//aJson.addBooleanToObject(sendJson, "WinterSummer", winsum);// add the state of WINSUM_PIN to the JSON
object
  aJson.addNumberToObject(sendJson, "msgCount", msgCount++); // count messages and add to the JSON object
}                                                           // END - createMsg()

voidserverMsg(void) {                                      // function to read the JSON message from
server

if (recvJson != NULL) {                                   // if the message is valid JSON
    aJsonObject* check = aJson.getObjectItem(recvJson, "Check"); // read the check value
if ((check != NULL) & (check->valuebool)) {               // if check is in the message and check = true
      aJsonObject* msgCnt = aJson.getObjectItem(recvJson, "msgCount"); // store to msgCnt the value
received from server
if (msgCnt->valueint == (msgCount-1)) {                // Check for lost messages
        Serial.println("\n*OK*\n");                         // print to serial console OK message
        digitalWrite(GREEN_LED, HIGH);                     // turn on Green Led to indicate success to
the user
      } else { digitalWrite(RED_LED, HIGH); digitalWrite(GREEN_LED, LOW); } // if check has wrong message
count turn on Red Led to inform the user
    } else {                                                // else if check = false
      aJsonObject* on = aJson.getObjectItem(recvJson, "OnOff");  // Read OnOff value from server
      aJsonObject* fan = aJson.getObjectItem(recvJson, "HiLow"); // Read HiLow value from server
      aJsonObject* ws = aJson.getObjectItem(recvJson, "WinterSummer"); // Read WinterSummer from server
if (on != NULL) {                                        // if on value exists in message
//Serial.println(on->valuebool);                        // print it to the serial console
if (on->valuebool) {                                    // if on = true
          digitalWrite(ONOFF_PIN, HIGH);                    // set pin to HIGH
        } else {                                            // else if on = false
          digitalWrite(ONOFF_PIN, LOW);                     // set pin to LOW
        }                                                   // END - if (on->valuebool)
      }                                                     // END - if (on != NULL)
if (fan != NULL) {                                       // if fan value exists in message
//Serial.println(fan->valuebool);                       // print it to serial console
if (fan->valuebool) {                                   // if fan = true
          digitalWrite(FAN_PIN, HIGH);                      // set pin to HIGH
        } else {                                            // else if fan = false
          digitalWrite(FAN_PIN, LOW);                       // set pin to LOW
        }                                                   // END - if (fan->valuebool)
      }                                                     // END - if (fan != NULL)
if (ws != NULL) {                                        // if ws exists in message
//Serial.println(ws->valuebool);                        // print it in serial
if (ws->valuebool) {                                    // if ws = true
          digitalWrite(WINSUM_PIN, HIGH);                   // set pin to HIGH
        } else {                                            // else if ws = false
          digitalWrite(WINSUM_PIN, LOW);                    // set pin to LOW
        }                                                   // END - if (ws->valuebool)
```

```
      }
      time4update = true;                                    // END - if (ws != NULL)
    }                                                        // set variable time4update = true
>valuebool))                                                 // END - if ((check != NULL) & (check-
  }                                                          // END - if (recvJson != NULL)
  free(recvJson);                                            // free the JSON string
  aJson.deleteItem(recvJson);                                // delete the JSON Object
}

voidreadOWTemperature(void){                               //Send a global temperature convert command
  ds.reset();                                               // reset the OneWire driver
  ds.write_byte(0xcc);                                      // was ds.select(work); so request all OW's to do
next command
  ds.write_byte(0x44);                                      // start conversion, with parasite power on at
the end
  delay(1000);                                              // wait 1 sec needed for the conversion
for (i=1; i<ROMmax+1;i++) {
if (ROMtype[i]==0x28) {
      readOW(i);
      saveTemperature(i);
    }
  }
for (i=1;i<ROMmax+1;i++){
if (ROMtype[i]==0x28) {
      foundOW = true;
//Serial.print(result[i]);
      ct = result[i];
    }
  }
  ds.reset();
//delay(500);
}

voidsaveTemperature(uint8_t ROMno){
  int32_t newtemp32;
float i; // uint8_t
  newtemp32=data[1]<<8;
  newtemp32=newtemp32+data[0]>>4;
  result[ROMno]=byte(newtemp32);
  i=(data[0] & 0x0F)* 625/1000;
if (i>=5)  result[ROMno]++;
}

voidreadOW(uint8_t ROMno)
{
   uint8_t i;
   ds.reset();
   ds.select(ROMarray[ROMno]);
   ds.write_byte(0xBE);         // Read Scratchpad
for ( i = 0; i < 9; i++) {   // need 9 bytes
      data[i] = ds.read_byte();
#if TEST
if (data[i]<16) Serial.print("0");
      Serial.print(data[i], HEX);
      Serial.print(" ");
#endif
  }
}

voidfindOW(void)
{
 byte addr[8];
 uint8_t i;
 ROMmax=0;  ////////////////////////////////////////////////////
while (true){  //get all the OW addresses on the buss
   i= ds.search(addr);
if ( i<10) {
//Serial.print("ret=(");
//Serial.print(i);
```

```
//Serial.print(") No more addresses.\n");
      ds.reset_search();
      delay(500);
return;
   }
//Serial.print("R=");
for( i = 0; i < 8; i++) {
if (i==0)  ROMtype[ROMmax+1]=addr[i];  // store the device type

      ROMarray[ROMmax+1][i]=addr[i];

//if (addr[i]<16) Serial.print("0");
//Serial.print(addr[i], HEX);
//Serial.print(" ");
    }
    ROMmax++;
    Serial.print ("(OW");
    Serial.print (ROMmax,HEX);
    Serial.print (") Type=");
    Serial.println (ROMtype[ROMmax],HEX);
 }
}

voidfailsafe() {

  digitalWrite(ONOFF_PIN, HIGH);     // failsafe mode set to HIGH
  digitalWrite(FAN_PIN, HIGH);       // failsafe mode set to HIGH
  digitalWrite(WINSUM_PIN, HIGH);    // failsafe mode set to HIGH
  Serial.println("Fail Safe Mode");

}

/*
void displayOW(void)
{
  uint8_t i;
  Serial.println ("From array");
  for (ROMcount=1; ROMcount<ROMmax+1; ROMcount++) {
    ds.reset();
    for( i = 0; i < 8; i++) {
      if (ROMarray[ROMcount][i]<16) Serial.print("0");
      Serial.print( ROMarray[ROMcount][i], HEX);
Serial.print(" ");
    }
    Serial.println("");
  }
}*/
```