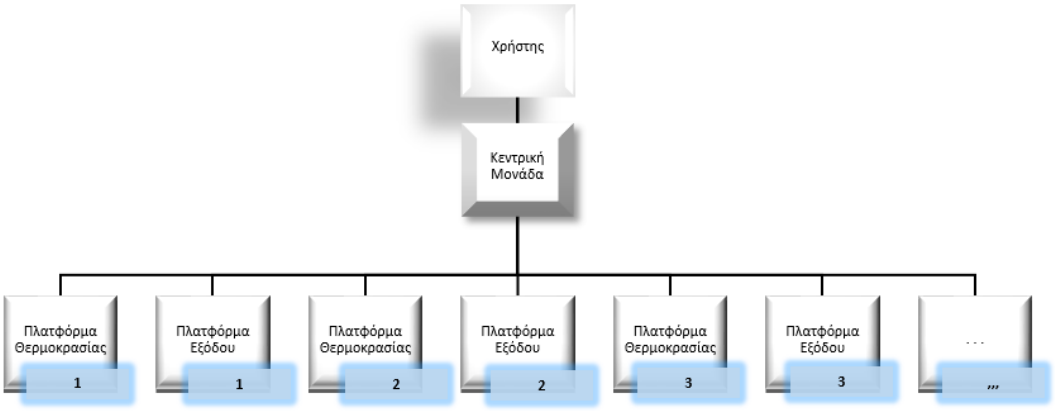
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ

ΘΕΜΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ:

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΠΟΜΑΚΡΥΣΜΕΝΟΥ ΕΛΕΓΧΟΥ ΘΕΡΜΟΚΡΑΣΙΑΣ ΧΩΡΟΥ ΜΕΣΩ ΠΡΟΣΑΡΜΟΖΩΜΕΝΩΝ ΘΕΡΜΙΚΩΝ ΖΩΝΩΝ



ΟΝΟΜΑ ΦΟΙΤΗΤΗ : ΓΚΑΜΠΛΕΤΣΑΣ ΑΝΔΡΕΑΣ - ΕΜΜΑΝΟΥΗΛ

ΕΠΙΒΛΕΠΩΝ:

ΜΙΧΑΗΛ ΠΑΠΟΥΤΣΙΔΑΚΗΣ
ΚΑΘΗΓΗΤΗΣ

ΙΑΝΟΥΑΡΙΟΣ 2022

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η πτυχιακή εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

α/α	ΟΝΟΜΑ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	ΜΙΧΑΗΛ ΠΑΠΟΥΤΣΙΔΑΚΗΣ	ΚΑΘΗΓΗΤΗΣ / ΕΙΣΗΓΗΤΗΣ	Michail Papoutsidakis Digitally signed by Michail Papoutsidakis Date: 2022.03.16 17:17:00 +02'00'
2	ΑΒΡΑΑΜ ΧΑΤΖΟΠΟΥΛΟΣ	ΛΕΚΤΟΡΑΣ ΕΦΑΡΜΟΓΩΝ	Avraam Chatzopoulos Digitally signed by Avraam Chatzopoulos Date: 2022.03.16 17:56:37 +02'00'
3	ΧΡΗΣΤΟΣ ΔΡΟΣΟΣ	ΜΕΛΟΣ ΕΔΙΠ	Christos Drosos Digitally signed by Christos Drosos Date: 2022.03.20 19:19:27 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γκαμπλέτσας Ανδρέας-Εμμανουήλ του Αντωνίου , με αριθμό μητρώου 19389378, φοιτητής του Πανεπιστημίου Δυτικής Αττικής, της Σχολής Μηχανικών, του Τμήματος Βιομηχανικής Σχεδίασης και Παραγωγής, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών/ούσα

ΠΕΡΙΛΗΨΗ

Η τεχνολογία είναι πλέον ένα αναπόσπαστο κομμάτι της ζωής του ανθρώπου, οπού η ραγδαία εξέλιξη τον έξυπνων συστημάτων βρίσκει εφαρμογή σε όλους τους τομείς στην καθημερινότητά του, με απώτερο σκοπό την ασφάλεια, την εξυπηρέτηση και την αποτελεσματικότητα. Στην σύγχρονη εποχή οπού ο χρόνος είναι πολύτιμος, όλα τα βλέμματα είναι στραμμένα στην εξοικονόμηση του χρόνου, κατασκευάζοντας διαφορά μέσα όπως ρομποτικά συστήματα, αυτοματοποιημένες διεργασίες, απομακρυσμένο έλεγχο, ασύρματα συστήματα και αλλά τεχνάσματα για την επίτευξη του στόχου. Στην παρούσα εργασία θα παρουσιαστεί ο σχεδιασμός ενός προηγμένου συστήματος απομακρυσμένης διαχείρισης θερμοκρασίας προσαρμοζόμενων θερμικών ζωνών, το οποίο εμπεριέχει συστήματα αυτοματισμού, ασύρματα δίκτυα, σχεδιασμό επικοινωνιακών συστημάτων αυτοματισμού και ανάλυση τεχνολογικά προηγμένων εφαρμογών. Το σύστημα μέσω μιας κεντρικής μονάδας ελέγχου θα επεμβαίνει στις παραμέτρους της κάθε θερμικής ζώνης ξεχωριστά. Από έναν ηλεκτρονικό υπολογιστή, ένα τάμπλετ, ή ένα έξυπνο κινητό τηλέφωνο, θα είναι δυνατόν να συνδέεται ο χρήστης με την κεντρική μονάδα που θα είναι υπεύθυνη για τις εισόδους – εξόδους του συστήματος, έχοντας έτσι τον έλεγχο της θερμοκρασίας στον εκάστοτε χώρο. Η καινοτομία της ιδέας είναι ότι η κεντρική μονάδα του συστήματος θα είναι εγκατεστημένη σε ειδική πλατφόρμα, η οποία θα έχει κατασκευαστεί για αυτό το σκοπό. Αρχικά θα ξεκινήσει ο σχεδιασμός του συστήματος που θα προσαρμοστούν όλα τα εξαρτήματα, θα μελετηθούν τα χαρακτηριστικά τους και θα επιλεχθούν εκείνα που είναι κατάλληλα. Έπειτα θα πραγματοποιηθεί η σχεδίαση και η σύνδεση για τις πλατφόρμες του συστήματος. Τέλος θα πραγματοποιηθεί η ανάπτυξη του κώδικα που θα τρέχει στις εκάστοτε πλατφόρμες, για να μπορεί να προσαρμοστεί στα δεδομένα του συστήματος και να εξυπηρετήσει με τον καλύτερο τρόπο την εφαρμογή, σε συνδυασμό με την μελέτη για τον απομακρυσμένο έλεγχο της εφαρμογής.

Abstract

Technology has become an integral part in individuals' life, in which the rapid development of smart systems is widely applicable in all different aspects of daily life in order to facilitate, serve and offer quality in everyday reality. At the same time, the quantitative technological development coexists with qualitative technological advancement, offering increasingly more complex, effective and safe applications. In modern age, given that time is valuable, all technological processes aim at saving time, by presenting various possibilities like robotic systems, automated processes, remote control, wireless systems and other technological inventions that achieve time management and saving.

In the present thesis, the design of a sophisticated system of remote temperature management and control for adapted zone heat installations will be presented. This presentation, within the framework of the Master Program, involves the combination and use of automated systems, wireless networks, the design of automated communication systems along with the analysis and study of technologically advanced applications. The system, via a central unit that will classify the different data, will perform temperature control and management of each zone heat installation separately. Through the use of a PC, laptop, tablet, or Smartphone, each user will be able to connect to the central unit that will be managing the system's data (input-output), thus allowing temperature control in specific spaces. The innovation of this concept resides in the fact that the central unit of the system will be installed in a platform, custom-made by myself, to allow different possibilities for its users.

Firstly, the design of the system that will include all the different components and fittings will be designed, adjust and finally select the most optimal and suitable ones. Then, the design and the connection of the system's platforms will be performed and the printing of the system's boards will follow. Finally, the source code for the different platforms will be created in order to adapt to the system's data and contribute in the most efficient way to the application's remote control and administration.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

1.1Εισαγωγή

Αρχικά θα πρέπει να ερευνηθεί ποια είδη αισθητήρων υπάρχουν, καθώς και ποιο από όλα αυτά είναι κατάλληλο για το σύστημά μας. Στη συνέχεια θα πρέπει να προσδιοριστούν τα είδη και οι τύποι των μικροελεγκτών έτσι ώστε να επιλεχθεί ο κατάλληλος με τα κοντινότερα επιθυμητά χαρακτηριστικά που εμπίπτουν με το σύστημα μας. Ακόμα θα πρέπει να διεξαχθεί μελέτη για τον τρόπο επικοινωνίας ανάμεσα σε πλατφόρμες και ποια μέθοδος θα είναι καλύτερη για την ασύρματη, αμφίδρομη επικοινωνία μεταξύ τους. Έπειτα πρέπει να βρεθεί η βέλτιστη λύση για την επικοινωνία ανάμεσα στο σύστημα και το χρήστη. Ένα απαραίτητο κομμάτι το οποίο απαιτεί μεγάλη προσοχή είναι η ηλεκτρική συνδέσει των εξαρτημάτων με απώτερο σκοπό την τύπωση των πλακετών που δομούν το σύστημα. Τέλος θα διεξαχθεί ο προγραμματισμός του συστήματος, για την κάθε μία πλατφόρμα ξεχωριστά.

1.2 Αισθητήρες μέτρησης θερμοκρασίας

1.2.1Θερμίστορς

Αρχή Λειτουργίας

Με την έννοια θερμίστορ εννοούμε αντιστάσεις όπου η τιμή τους μεταβάλλεται ανάλογα με τη θερμοκρασία. Σε σύγκριση με της θερμό - αντιστάσεις, τα θερμίστορ εμφανίζουν μεγαλύτερες μεταβολές της τιμής στην αντίστασή τους. είναι οξείδια των μεταβατικών μετάλλων της σειράς του σιδήρου, για παράδειγμα ο σίδηρος, το χρώμιο, το μαγνήσιο, το νικέλιο και το κοβάλτιο. η εσωτερική αντίσταση τους μεταβάλλεται με ραγδαίο ρυθμό σε σχέση με τη θερμοκρασία, αλλά έχει υψηλά όρια αντοχής. Αυτό έχει σαν αποτέλεσμα να μην έχουν ακρίβεια μέτρησης θερμοκρασίας όπως άλλοι μέθοδοι. Επίσης η μεγάλη μεταβολή της αντίστασης επιτρέπει να χρησιμοποιηθούν σε κυκλώματα ως διακόπτες ή περιοριστές ρεύματος. Τα θερμίστορ είναι ευρέως διαδεδομένα ως μία οικονομική επιλογή για μέτρηση θερμοκρασίας. υπάρχουν δύο είδη θερμίστορ, τα NTC (negative temperature coefficient, αρνητικού θερμοκρασιακού συντελεστή) και τα PTC (positive temperature coefficient, θετικού

θερμοκρασιακού συντελεστή). Στα NTC Αυξάνεται η θερμοκρασία και μειώνεται η αντίσταση, αντίθετα στα PTC Αυξάνεται η θερμοκρασία αλλά αυξάνεται και η αντίσταση.

- Τα θερμίστορς NTC

Τα NTC θερμίστορ παρουσιάζουν μεγάλες μεταβολές αντιστάσεις σε μικρές υφιστάμενες μεταβολές θερμοκρασίας. συνήθως χρησιμοποιούνται για μέτρηση θερμοκρασίας από τους -100 έως τους 300°C. Η μεταβολή της αντίστασης υπολογίζεται από το πηλίκο της τιμής αντίστασης στους 25°Cπρος την αντίσταση στους 125°C. συνήθως η αντίσταση στους 125°C γίνεται 20 έως 40 φορές μικρότερη σε σχέση με τη θερμοκρασία τον 25°C. Οι ανοχές των θερμίστορ είναι αρκετά υψηλότερες σε σχέση με αυτές των θερμοηλεκτρικών αντιστάσεων(5% σε αναλογία με τη θερμοκρασία). Εμφανίζουν υψηλή χρονική σταθερότητα, με αποτέλεσμα στους 100°C και με διάρκεια 1000 ώρες, η τιμή της αντίστασης να αλλάζει 0,1%.Οι συνηθέστερες μορφές πού θα βρούμε στο εμπόριο είναι σε μορφή δίσκου, ράβδου ή κάψουλας. Η έρευνα της αντίστασης από τη θερμοκρασία έχει την γενική μορφή:

$$R_T = R_{T_0}\, e^{B\left(\frac{1}{T}-\frac{1}{T_0}\right)}$$

όπου Τ είναι η θερμοκρασία σε βαθμούς Κέλβιν και Τ0 η θερμοκρασία αναφοράς (298 Κ, που αντιστοιχούν τους 25°C), ενώ RT,RT0είναι οι τιμές της αντίστασης του θερμίστορ NTC. Η παραπάνω σχέση στην πράξη δεν είναι πολύ χρήσιμη, στην ουσία θέλουμε να προσδιορίσουμε την τιμή Τ γνωρίζοντας τις άλλες τρεις ποσότητες. λύνοντας την εξίσωση ως προς Τ προκύπτει ο τύπος του Steinhart:

$$\frac{1}{T} = a + b\,(\ln R) + c\,(\ln R)^3$$

Η θερμοκρασία Τ προκύπτει σε βαθμούς Κέλβιν. Οι ποσότητες α, b και c είναι συντελεστές που αναφέρονται στο συγκεκριμένο θερμίστορ. Για παράδειγμα, σε ένα θερμίστορ, όπως το μοντέλο P1H103T της εταιρείας Therm έχει αντίσταση 10 kΩ ± 2%στους 25° C, οι συντελεστές α, b και c έχουν τις ακόλουθες τιμές:

P1H103T:

$$\alpha = 1.125190920 \cdot 10^{-3}$$

$$b = 2.347363293 \cdot 10^{-4}$$

$$c = 8.551343472 \cdot 10^{-8}$$

Με βάση τις παραπάνω τιμές, η ακρίβεια της θερμοκρασίας από τον τύπο του Steinhart δίνει ακρίβεια καλύτερη από 0.05^{o} C στην περιοχή 0 – 100^{o} C. Ο πίνακας τιμών θερμοκρασίας - αντίστασης του θερμίστορ P1H103T στην παραπάνω περιοχή θερμοκρασιών παρουσιάζεται στον Πίνακα1.

Θερμοκρασία σε (^{o}C)	**Αντίσταση σε (Ω)**
0	32.654
10	19.903
20	12.493
30	8.056
40	5.327
50	3.603
70	1.752
100	680

Πίνακας 1: Τιμές θερμοκρασίας - αντίστασης του θερμίστορ NTC P1H103T της εταιρείας Therm-O-Disk

Τα θερμίστορ όπως και άλλοι αισθητήρες θερμοκρασίας εμφανίζουν το φαινόμενο αυτό - θέρμανσης (self-heating effect), δηλαδή αυξάνεται η θερμοκρασία τους όταν διαρρέονται από ρεύμα, με αναλογία 1^{o}Cανά 7 mW εσωτερικής ηλεκτρικής ισχύς. Τα θερμίστορ που έχουνε μεγάλη επιφάνεια όπως αυτά με σχήμα δίσκου, αποβάλλουν αυτό το φαινόμενο λόγω του ότι επάγουν τη θερμοκρασία από πάνω τους. όσο πιο μικρό είναι το μέγεθος του θερμίστορ τόσο πιο έντονο είναι το φαινόμενο.

- Θερμίστορς PTC

Στα θερμίστορ PTC η αντίστασή τους αυξάνεται με τη αύξηση της θερμοκρασίας. Χρησιμοποιούνται ως περιοριστές ρεύματος σε θερμοκρασίες ανάμεσα 50 έως και 250^{o} C. Η τιμή της αντίστασης των PTC κυμαίνεται από 0,5 έως 20 kΩ σε θερμοκρασία δωματίου 25^{o} C. Επειδή η αύξηση της αντίστασης σε αναλογία με την αύξηση της θερμοκρασίας γίνεται πολύ απότομα, δεν είναι κατάλληλα για εφαρμογές με συνεχή μέτρηση θερμοκρασίας, διότι υπάρχει περίπτωση διακοπής τροφοδοσίας στο κύκλωμα. Για των λόγο αυτό έχουν ονομαστεί θερμοευαίσθητοι ή ηλεκτρικοί διακόπτες, όπου δεν έχουνε κινούμενα μέρη, άρα δεν υπάρχει φθορά, με αποτέλεσμα τη μεγάλη διάρκεια ζωής τους. τα PTC αυτό - θερμαίνονται όταν διαρρέονται από ρεύμα, αυτό είναι ωφέλιμο σε εφαρμογές. οι συνηθέστερες εφαρμογές τον θερμίστορ PTC είναι:

- **Μέτρηση της ροής ρευστών** Εάν ένα PTC βρεθεί σε ακίνητο υγρό ή σε ακίνητο αέρα, δεν αποβάλλει θερμότητα και αυτό – θερμαίνεται γρήγορα. για αυτό το λόγο μεταβαίνει γρήγορα στην κατάσταση υψηλής αντίστασης, όπου μπορεί να διακόψει ένα ηλεκτρονικό κύκλωμα. Με τον τρόπο αυτό μπορεί να ελέγχει μία διαδικασία ψύξης, τοποθετημένο μέσα στο ψυκτικό αέριο ή υγρό. Εάν η ροή μειωθεί, το θερμίστορ αποκτά μεγάλη αντίσταση και ενεργοποιεί το κύκλωμα προειδοποίησης ή το κύκλωμα ελέγχου της ροής..

- **Χρονική καθυστέρηση** Εάν διαπεράσουμε ένα θερμίστορ PTC με συγκεκριμένο ρεύμα, το θερμίστορ θα αυτό - θερμανθεί και σε συγκεκριμένο χρόνο θα οδηγηθεί στην κατάσταση υψηλής αντίστασης, στην οποία μπορεί να ενεργοποιήσει κάποιο κύκλωμα. Άρα μπορούμε να ενεργοποιήσουμε κάποιο κύκλωμα μετά από συγκεκριμένο χρονικό διάστημα, να έχουμε δηλαδή μία επιθυμητή χρονική καθυστέρηση στη λειτουργία του κυκλώματος. Είναι σημαντικό ανάμεσα από κάθε αυτό - θέρμανση να έχει προηγηθεί ψύξη του PTC διότι σε διαφορετική περίπτωση η δεύτερη αυτό - θέρμανση θα έχει διαφορετικό χρόνο μεταβολής τις καταστάσεις.

- **Προστασία μπαταριών** από υπερφόρτιση. Όταν οι μπαταρίες φορτίζονται στην ανωτάτη χωρητικότητα τους, θερμαίνονται. Σε αυτή την περίπτωση θα μπορούσε να τοποθετηθεί ένα θερμίστορ PTC πού θα βρίσκεται σε επαφή με μία από τη

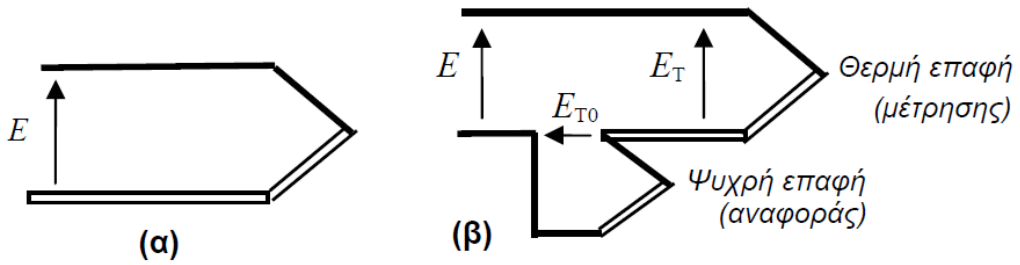
συστοιχία των μπαταριών. Έτσι θα μπορεί να ανιχνεύσει ποτέ φορτίζει πλήρως η συστοιχία από μπαταρίες. Όταν η θερμοκρασία της μπαταρίας αυξηθεί θα αυξηθεί απότομα και η αντίσταση του PTC με αποτέλεσμα την διακοπή τροφοδοσίας της μπαταρίας.

1.2.2 Θερμοζεύγη

Τα θερμοηλεκτρικά ζεύγη ή θερμοζεύγη (thermocouples) αποτελούν ένα διαδεδομένο είδος ανιχνευτών θερμοκρασίας, με υψηλή ακρίβεια και χαμηλό κόστος. Αποτελούνται από δύο διαφορετικά μεταλλίκια σύρματα, τα οποία είναι ενωμένα σε δύο σημεία

Αρχή Λειτουργίας

Τα θερμοζεύγη στηρίζονται στη λειτουργία του φαινομένου του θερμοηλεκτρισμού (thermoelectricity).Όταν δύο διαφορετικά μέταλλα ενωθούν σε ένα σημείο, αναπτύσσεται μία τάση στα άκρα τους η οποία λέγεται θερμοηλεκτρική τάση ή αλλιώς δυναμικό επαφής. Η τάση που αναπτύσσεται στα άκρα εξαρτάται από τη θερμοκρασία. Άρα εάν δύο μέταλλα ενωθούν σε δυο διαφορετικά σημεία τότε θα δημιουργήσουν δύο διαφορετικές θερμοηλεκτρικές τάσεις. Σχήμα 1α απεικονίζεται η δομή του θερμοζεύγους. και στο Σχήμα 1β απεικονίζεται ένα θερμοζεύγος με δύο επαφές, μία ψυχρή και μία θερμή.



Σχήμα 1 (α) Εμφάνιση θερμοηλεκτρικής τάσης όταν δύο μέταλλα ενώνονται.
Σχήμα 1 (β)Εμφάνιση δύο διαφορετικών τάσεων για Δύο διαφορετικές επαφές ψυχρή και θερμή ενός θερμοζεύγους

Είδη θερμοζεύγων

Τα θερμοζεύγη κατασκευάζονται από συγκεκριμένα μέταλλα κράματα μετάλλων, τα οποία αναπτύσσουν θερμοηλεκτρική τάση που μεταβάλλεται σημαντικά με τη θερμοκρασία.

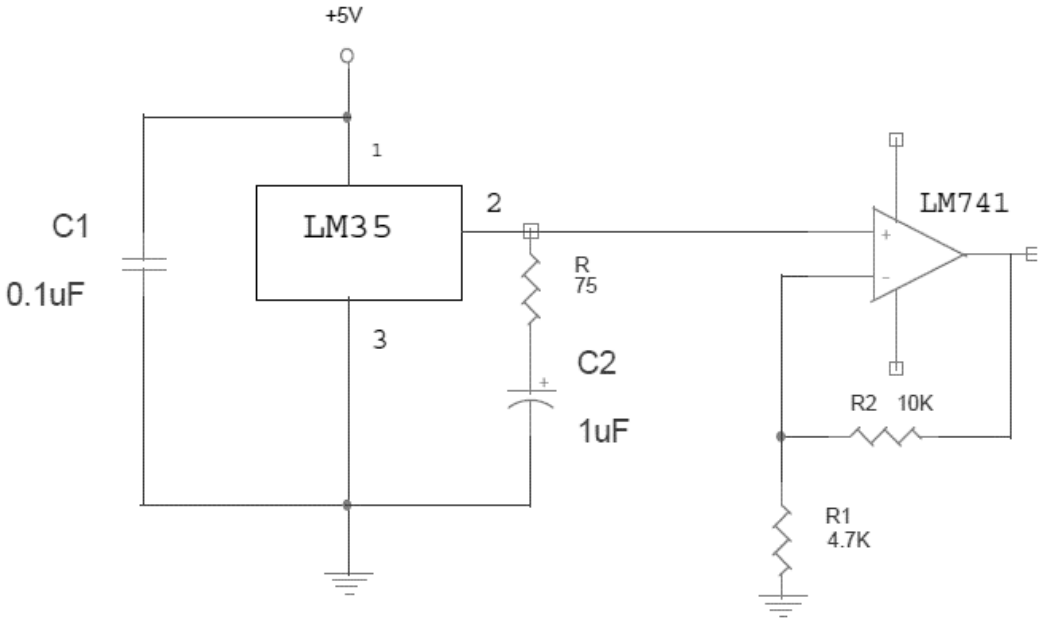
- **Σιδήρου - Κονσταντάνης** (iron-constantan) Οι ακροδέκτες έχουν χρώματα λευκό και κόκκινο. Αναπτύσσει θερμοηλεκτρική τάση 50 μV/^{o}C.
- **Νικελίου/Χρωμίου - Νικελίου/Αλουμινίου** (chromel-alumel) Οι ακροδέκτες έχουν χρώματα κίτρινο και κόκκινο. Αναπτύσσει θερμοηλεκτρική τάση 40 μV/^{o}C.
- **Χαλκού - Κονσταντάνης** (copper-constantan) Οι ακροδέκτες έχουν χρώματα μπλε και κόκκινο. Αναπτύσσει θερμοηλεκτρική τάση 40 μV/^{o}C. Τα είδη των θερμοζευγών που διατίθενται σήμερα στο εμπόριο αναγράφονται στον Πίνακα 2.

Είδος θερμοζεύγους	**θετικό άκρο +**	**θετικό άκρο -**	**περιοχή λειτουργίας**
B	Λευκόχρυσος 30% ρόδιο	Λευκόχρυσος 6% ρόδιο	1370 – 1700 ^{o}C
C	W5Re βολφράμιο 5% Ρηνιο	W5Re βολφράμιο 5% Ρηνιο	1650 – 2315^{o}C
E	Chromel	Κονσταντάνη	95 – 900^{o}C
J	Σίδηρος	Κονσταντάνη	95 – 760^{o}C
K	Chromel	Alumel	95 – 1260^{o}C
N	Nicrosil	Nisil	650 – 1260^{o}C
R	Λευκόχρυσος 13% ρόδιο	Λευκόχρυσος	870 – 1450^{o}C
S	Λευκόχρυσος 10% ρόδιο	Λευκόχρυσος	980 – 1450^{o}C
T	Χάλκος	Κονσταντάνη	-200 –350^{o}C

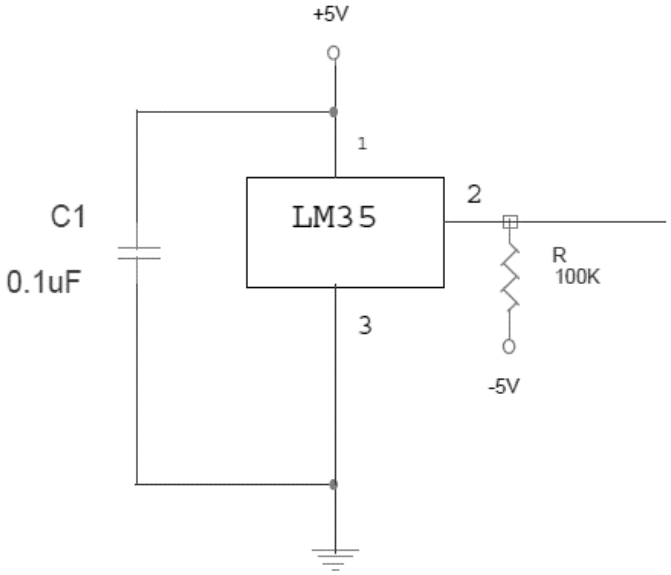
Πίνακας 2: Είδη θερμοζευγών και περιοχές λειτουργίας τους

1.2.3 Το Ολοκληρωμένο Κύκλωμα LM35

Το ολοκληρωμένο κύκλωμα LM35 Κατασκευάστηκε από την εταιρεία National Semiconductor και είναι ένας από τους πιο διαδεδομένους αισθητήρες θερμοκρασίας. Η λειτουργία του στηρίζεται στις ιδιότητες της επαφής pn. είναι γνωστό όταν η επαφή pn είναι πολωμένη ανάστροφα, τότε το ανάστροφο ρεύμα κόρου είναι συνάρτηση της θερμοκρασίας της επαφής. Τα ολοκληρωμένα αισθητήρια εμφανίζουν πολύ καλή γραμμικότητα. Το LM35 μεταβάλλει την έξοδό του κατά 10 mV όταν η θερμοκρασία μεταβάλλεται κατά 1° C. Το κύκλωμα LM35 ανταποκρίνεται σε θερμοκρασίες μεταξύ −55 και +150° C και αποδίδει έξοδο περίπου 10.0 mV ανά βαθμό Κελσίου. Το ρεύμα που το διαρρέει είναι μόλις 60 μA και το κύκλωμα εμφανίζει χαμηλό φαινόμενο αυτοθέρμανσης, περίπου 0.1° C όταν ευρίσκεται σε ακίνητο αέρα. Μπορεί να τροφοδοτηθεί από τάση, μεταξύ +4 και +30 V.



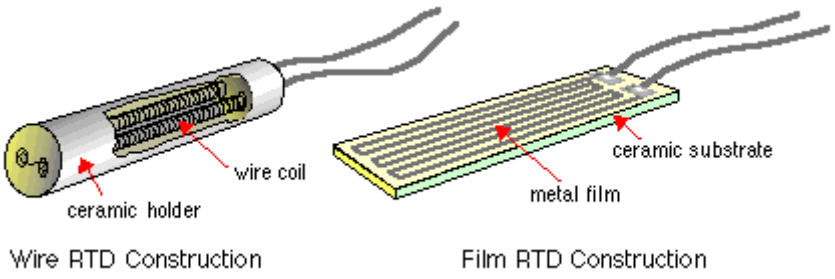
Σχήμα 2Συνδεσμολογία LM35 ώστε να μετρά θερμοκρασίες από 0 έως 150° C.



Σχήμα 3Συνδεσμολογία τουLM35 για μετρήσεις και σε αρνητικές θερμοκρασίες

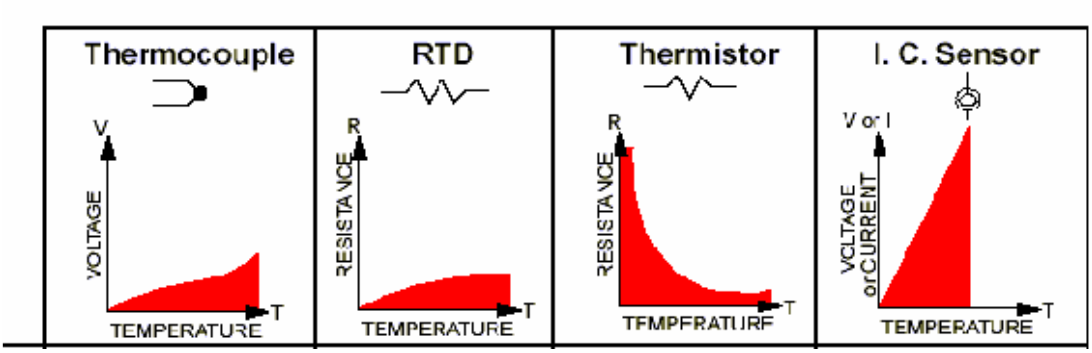
1.2.4 Θερμοηλεκτρικές αντιστάσεις

Είναι γνωστό ότι ηλεκτρική αντίσταση των μετάλλων αυξάνεται καθώς αυξάνεται η θερμοκρασία. Επομένως είναι δυνατό να κατασκευαστεί μία διάταξη μέτρησης της θερμοκρασίας μετρώντας την ηλεκτρική αντίσταση ενός πρότυπου μετάλλου, όταν μεταβάλλεται η θερμοκρασία του. Με αυτό τον τρόπο κατασκευάζονται οι θερμοηλεκτρικές αντιστάσεις. Η μέτρηση της αντίστασης του θερμοηλεκτρικού στοιχείου γίνεται με τη βοήθεια της γέφυρας Wheatstone ή με τη βοήθεια μιας πηγής ρεύματος που διεγείρει τι θερμοηλεκτρική αντίσταση. οι θερμοηλεκτρικές αντιστάσεις έχουν ακρίβεια λιγότερο από 1^{o} C. Συνήθως ένα τυπικό μέταλλο για την κατασκευή των θερμοηλεκτρικών αντιστάσεων είναι η πλατίνα.



Εικόνα 1:Μερικά από τα σχισματικά είδη που υπάρχουν στο εμπορείο

1.2.5 Σύγκριση γραφήματος αισθητηρίων θερμοκρασίας

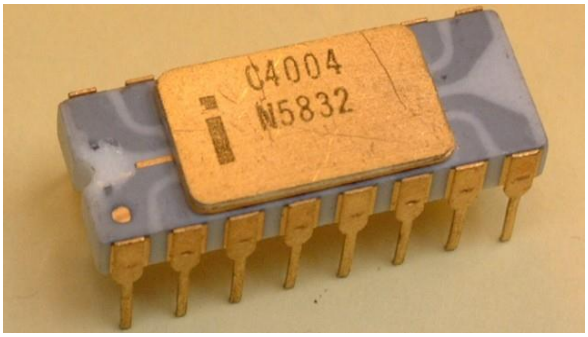


Γράφημα 1: Τα γραφήματα των τεσσάρων αισθητήρων θερμοκρασίας

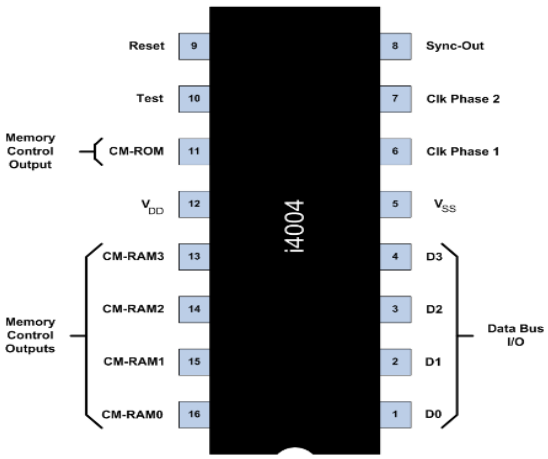
Ανάλογα με τη χρήση που προορίζονται αισθητήρας θερμοκρασίας, αλλά και με γνώμονα την χρονική και θέρμη απόκριση τους, γίνεται η κατάλληλη επιλογή.

1.3 Η τεχνολογία των μικροελεγκτών

Το Νοέμβριο του 1971 κατασκευάστηκε ο πρώτος ολοκληρωμένος μικροελεγκτής στην παγκόσμια ιστορία. Η εταιρεία κατασκευής του πρώτου επεξεργαστή ήταν η Intel, με το μοντέλο 4004, κάποιοι αυτό το μοντέλο το χαρακτήρισαν ‘’όλα σε ένα’’. Λόγω του ότι ο 4004 είχε ενσωμάτωση στο ολοκληρωμένο το κύκλωμα του 2.300 transistors. Χαρακτηριστικά του μικροεπεξεργαστή ήταν, εσωτερικό ρολόι χρονισμού, εσωτερική μνήμη με τη μορφή καταχωρητών (registers), μονάδα αριθμητικών υπολογισμών ALU,4 bit, με ταχύτητα 740 kHz και δυνατότητα ταυτόχρονης επεξεργασίας 46 εντολών.

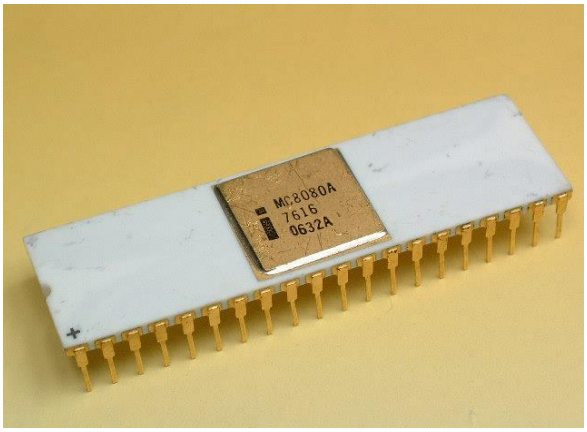


Εικόνα 2: ο μικροεπεξεργαστές Intel 4004



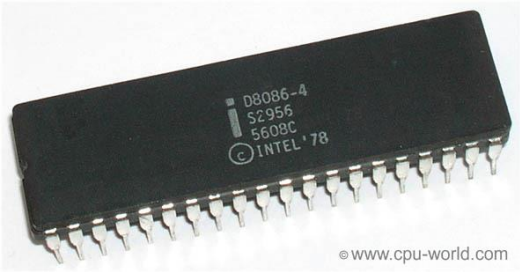
Σχήμα4: ο μικροεπεξεργαστές Intel 4004, αντιστιχεια αριθμιμενον pin και ονομασιας

Το 1974 εμφανίζεται ο πρώτος 8 bit επεξεργαστής της Intel 8080.Ήτανη εξέλιξη του 8008. Αποτελούνταν από 6.000 τρανζίστορ με 2 MHz συχνότητα χρονισμού λειτουργίας. περίπου την ίδια περίοδο η Motorola παρουσίασε τον 6800 που χρησιμοποιήθηκε σε υπολογιστές και σε όλα τα παιχνίδια pinball, όπως επίσης και σε κατασκευές ελέγχου στη βιομηχανία. Έχει 4000 τρανζίστορ, συχνότητα χρονισμού λειτουργίας 1MHz ή 2 MHz, 78 εντολές και 16 bit πλάτος διαύλου διευθύνσεων. Το 1975 η Zilog φτιάχνει τον Z80, έναν μικροεπεξεργαστής 8 bit βασισμένο στον 8080. Η συχνότητα χρονισμού ήταν στα 3,5 MHz με 16 bit πλάτος διαύλου διευθύνσεων, ενώ μπορούσε να ανταποκριθεί σε 64 kilobyte μνήμης.

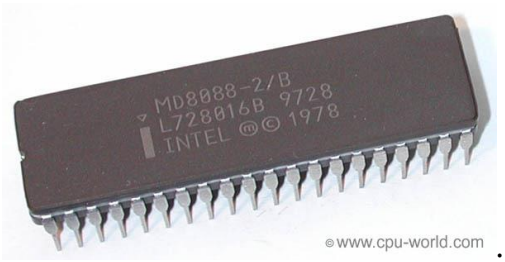


Εικόνα 3: Ο Intel 8086/ 8088

Το 1978 για πρώτη φορά κάνουν την εμφάνισή τους ή 16 bit μικροεπεξεργαστές. Η Intel παρουσιάζει τον 8086/ 8088 με συχνότητα χρονισμού λειτουργίας να έχει φτάσει τα 10 MHz και ενσωματώνει 29.000 τρανζίστορ. Την αντίστοιχη περίοδο η Motorola παρουσίαση τον 68000 με συχνότητα χρονισμού λειτουργίας 8MHz και περιέχει 68.000 τρανζίστορ. Ο μεγάλος του χώρος αποθήκευσης(16 MBytes)σε συνδυασμό με την υψηλή ταχύτητα και το χαμηλό κόστος τον καθιστά πρώτη επιλογή σε πολλές εφαρμογές



Εικόνα 4: Ο Intel 8086



Εικόνα 5: Ο Intel 8088

Το 1982 η Intel δημιουργεί τον 80286 με συχνότητα χρονισμού αρχικά στα 6MHz και αργότερα στα 12,50MHz, Με πάνω από 134.000 εσωτερικά τρανζίστορ και πλάτος διαύλου δεδομένων 16 bit και πλάτος διαύλου διευθύνσεων 24 bit, δυνατότητα αναγνώρισης μνήμης μέχρι 16 Mbytes. Με δυνατότητα να λειτουργεί σε κατάσταση Protected Mode (προστατευμένη κατάσταση λειτουργίας). Την αντίστοιχη χρονική περίοδο η Motorola μετά την επιτυχία της με τον 68.000 προχώρησε στον σχεδιασμό του MC 68010 που είχε ως έξτρα λειτουργία την ύπαρξη εικονικής μνήμης.

Το 1985 έκαναν την πρώτη τους εμφάνιση οι 32bit μικροεπεξεργαστές από την Intel. ο 80486 πού εμπεριέχει 275.000 τρανζίστορ και συχνότητα χρονισμού 33 MHz, με μνήμη που αγγίζει τα 4 Gbytes.

Το 1989 η Intel εμφανίζει τον 32-bit μικροεπεξεργαστή 80486 που εμπεριέχει 1.200.000 τρανζίστορ με συχνότητα χρονισμού λειτουργίας 50MHz.

Το 1993 η Intel παρουσιάζει τον πρώτο μικροεπεξεργαστή intel Pentium της οικογένειας p5 ο οποίος εμπεριέχει 3.100.000 τρανζίστορ, με συχνότητα χρονισμού 66 Hertz. την ίδια χρονική περίοδο η digital παρουσιάζει τον πρώτο επεξεργαστή 64 bit με όνομα Alpha.

Το 1997 η Intel παρουσιάζει τον μικροεπεξεργαστή Pentium 2 τεχνολογία mmx για την υποστήριξη πολυμέσων. έχει 7.500.000 τρανζίστορ και συχνότητα χρονισμού λειτουργίας 300MHz.

Το 1999 η Intel ανακοίνωση των Pentium 3 που εμπεριέχει 9.500.000 τρανζίστορ και συχνότητα χρονισμού λειτουργίας στα 450MHz

Το 2000 έκανε την εμφάνισή του ο Pentium 4 με σχεδίαση σύμφωνα με την μικροαρχιτεκτονική του netbrust. Η συχνότητα χρονισμού Έφτασε το 1 GHz από το σημείο αυτό και ύστερα έως και σήμερα οι μικροεπεξεργαστές έχουν φτάσει σε τεχνολογία με πολλαπλούς πυρήνες και με συχνότητες να αγγίζουν τα 3,5 GHz για κάθε πυρήνα.

Η έννοια μικροεπεξεργαστής αντιπροσωπεύει ένα ολοκληρωμένο κύκλωμα IC που εμπεριέχουν μόνο CPU στο εσωτερικό τους. Παραδείγματος χάρη pentium i, ii, iii, iv της Intel, αυτή η μικροεπεξεργαστές δεν έχουν μνήμη RAM, και άλλα περιφερειακά τσιπ. Για να είναι λειτουργικό ένα τέτοιο σύστημα θα πρέπει να προστεθούν με

εξωτερική σύνδεση αλλά τσιπ μνήμης. Εφαρμογές μικροεπεξεργαστών συναντάμε σε ηλεκτρονικούς υπολογιστές.

Αντίθετα αυτό δεν ισχύει στους μικροελεγκτές. Αυτοί ενσωματώνουν εκτός από τη CPU και άλλες επιπρόσθετες μνήμες RAM και ROM για αυτό και χαρακτηρίζονται ως μικροί υπολογιστές. Οι μικροελεγκτές είναι σχεδιασμένη για να εκτελούν συγκεκριμένες διεργασίες. Αυτό σημαίνει εφαρμογές όπου η σχέση εισροών και εκροών είναι προκαθορισμένη, όπως ποντίκια, πληκτρολόγια, πλυντήριο ρούχων, αυτοκίνητο, κινητό τηλέφωνο κ.λ.π, λόγω του ότι αυτές οι εφαρμογές χρειάζονται μικρούς πόρους για μνήμες RAM και ROM θύρες I/0 μπορούν να ενσωματώσουν οι μικροελεγκτές όλα αυτά μέσα σε ένα μικρό chip με σκοπό το μικρό μέγεθος και το χαμηλό κόστος.

Οι μικροελεγκτές για να μπορούν να επεξεργαστούν αλλά και να αναγνωρίσουν όλα αυτά τα στοιχεία που τους δίνονται ο μόνος τρόπος είναι η γλώσσα προγραμματισμού. Η μόνη γλώσσα προγραμματισμού για τους μικροελεγκτές είναι η γλώσσα μηχανής. με τον όρο γλώσσα μηχανής ονομάζουμε εντολές γραμμένες σε μορφή ακολουθιών bit εκτελέσιμες άμεσα από την κεντρική μονάδα επεξεργασίας. Ονομάζεται γλώσσα μηχανής διότι μέσω αυτής και μόνο αυτής επιτυγχάνεται η επικοινωνία με τον μικροελεγκτή. Προγράμματα που γράφουν σε άλλες γλώσσες προγραμματισμό, για παράδειγμα γλώσσας C ή C++, και επιτυγχάνουν επικοινωνία με τους μικροελεγκτές χρησιμοποιούν converter για να μεταφράσουν τις εντολές σε γλώσσα μηχανής.

1.4Απομακρυσμένος Έλεγχος

Για να επιτευχθεί απομακρυσμένος έλεγχος η γενικότερα η επικοινωνία διαφόρων συσκευών μεταξύ τους, απαιτείται ένας τρόπος επικοινωνίας ο οποίος θα μεταφέρει την εκάστοτε πληροφορία. Στις μέρες μας έχουν αναπτυχθεί πολλές μέθοδοί ασύρματης λειτουργίας. Οι πιο δημοφιλείς είναι το Wi-Fi, Bluetooth, RFSystems, nRF system και το Li-fi

1.4.1Li-Fi

Στις μέρες μας ο άνθρωπος ανακαλύπτει συνεχώς καινούργιες τεχνολογίες και εφαρμογές. Στην προσπάθεια του αυτή ανακαλύφθηκε ότι μέσο μιας φωτεινής πηγής τύπου Led, μπορούμε να διαδώσουμε δεδομένα. Το Li-Fi (Light-Fidelity) χρησιμοποιεί το ηλεκτρομαγνητικό φάσμα για τη μεταφορά δεδομένων. Αντί όμως για ραδιοσήματα, το Li-Fi μεταδίδει δεδομένα σε δυαδικό κώδικα χρησιμοποιώντας την τεχνολογία VLC (Visible Light Communication) που αξιοποιεί το φως από LED. Αυτό έχει ως αποτέλεσμα οι μόνοι δέκτες να είναι αυτοί που είναι πλησιέστεροι στην φωτεινή πηγή, διότι το φώς δεν διαπερνά αντικείμενα και τοίχους. Έτσι μπορούμε να διασφαλίσουμε τα δεδομένα μας από κακοπροαίρετους που βρίσκονται εκτός δωματίου σε περιπτώσεις που είναι αναγκαία η διασφάλιση δεδομένων. Το σύστημα λειτουργεί με αντίστοιχο τρόπο με τον παραδοσιακό κώδικα σημάτων Μορς, χρησιμοποιώντας ορατό φως (VLC) αλλά το οποίο λειτουργεί σε ταχύτητες οι οποίες είναι πολύ υψηλές για να γίνουν αισθητές από το ανθρώπινο μάτι (συχνότητες μεταξύ 400 - 800 THz). Αυτό καθιστά το Li-Fi 100 φόρες ποιο γρήγορο από τον ανταγωνιστή του το Wi-Fi, χωρίς όμως να τον αντικαθιστά πλήρως (τουλάχιστον μέχρι σήμερα).

1.4.2 Wi - fi

Η ονομασία Wi-Fi αναφέρεται για να προσδιορίσει συσκευές WLAN με προδιαγραφή IEEE 802.11 με συχνότητα εκπομπής 2,4 GHz. Γενικότερα η ονομασία Wi-Fi έχει επικρατήσει ως ένας όρος αναφερόμενος γενικότερα στα ασύρματα τοπικά δίκτυα. Η πρώτη εμφάνιση του Wi-Fi έγινε το 1997 όπου στο φυσικό επίπεδο περιλαμβάνει δύο μεθόδους διασποράς φάσματος για μετάδοση στη ζώνη συχνότητας 2,4 GHz αυτή η εκπομπή δεν απαιτεί άδεια. Με ονομασία frequency hopping ήταν η πρώτη μέθοδος λειτουργίας FHss με ρυθμό μετάδοσης έως 1 Mbps Ενώ η δεύτερη λειτούργησε με Direct sequence (DSSS) με ρυθμό μετάδοσης έως 2 Mbps. Το 1999 το 802.11B αύξησε το ρυθμό λειτουργίας στα 11Mbps χρησιμοποιώντας dsss. Το 2000 εμφανίστηκε μία πρωτοπόρα μέθοδος πρόσβασης στο Internet. Ήταν η πρώτη φορά που μία ψηφιακή συσκευή ασύρματης δικτύωσης Wi-Fi και για παράδειγμα ένας ηλεκτρικός υπολογιστής, μπορεί να συνδεθεί στο διαδίκτυο ασύρματα, όταν αυτό βρίσκεται σε ακτίνα κάλυψης ενός ασύρματου δικτύου το οποίο ονομάζεται access

point. Ακόμα μία περιοχή που μπορεί να καλύπτεται από 1 ή περισσότερα ασύρματα σημεία προσβάσεις συνδεμένα μεταξύ τους λέγεται hotspot.

1.4.3 TO RF Home

Μία άλλη εναλλακτική πρόταση για τα δίκτυα και με σχετικά λίγους κόμβους είναι το πρότυπο RF home, το οποίο συνδυάζει αρκετά στοιχεία από το πρότυπο 802.11 βάσει ευρωπαϊκού προτύπου ψηφιακής ασύρματης τηλεφωνίας, δημιουργώντας ένα πιο οικονομικό πακέτο μεταφοράς δεδομένων με ταχύτητα έως 2 Mbps. Αυτή η τεχνολογία έχει εξυπηρετήσει ακομα και την ασύρματη επικοινωνία για πειραματικούς σκοπούς (π.χ πλατφόρμες open τύπου Arduino). Η συχνότητα εκπομπής των δεδομένων είναι τα 2.4 GHz.

1.4.4 TO Nrf

Βασισμένο στον πομποδέκτη RF home το nRF είναι μία πιο βελτιωμένη έκδοση όπου έχει μειωθεί σημαντικά η κατανάλωση του χωρίς να επηρεαστούν η περαιτέρω ικανότητες του. Σαν μία πιο εξελιγμένη έκδοση έχει βρει χώρο στη βιομηχανία, ιατρική και επιστημονική κλίμακα εκπέμποντας στα 2.4 GHz.

1.4.5 TO Bluetooth

Είναι ένα βιομηχανικό πρότυπο το οποίο αφορά ασύρματα δίκτυα συσκευών. πρόκειται για μία τηλεπικοινωνιακή ασύρματη τεχνολογία που αφορά μικρές αποστάσεις, η οποία μπορεί να μεταδώσει δεδομένα μέσω μικροκυμάτων σε ψηφιακές συσκευές. Οι προδιαγραφές το καθορίζουν ως μία τεχνολογία χαμηλής ισχύς και χαμηλού κόστους. Συχνότητα λειτουργίας είναι τα 2,4 GHz έτσι ώστε όλες οι συσκευές που το ενσωματώνουν να μπορούν χωρίς πρόβλημα να επικοινωνούν μεταξύ τους. Για να περιοριστούν όσο το λιγότερο δυνατό οι παρεμβολές από άλλες παρεμφερή συσκευές το Bluetooth ενσωματώνει μία μέθοδο η οποία εκμεταλλεύεται τη διασπορά φάσματος ‘’frequency hopping’’. Αυτό σημαίνει ότι μπορεί να κάνει έως και 1.600 εναλλαγές συχνότητας ανά δευτερόλεπτο. Γενικότερα είναι μία

προδιαγραφή ανοικτή όπου μπορούν οι επιχειρήσεις να υιοθετήσουν ελεύθερα. οι απαιτήσεις των προτύπων έχουν ένα περιορισμένο φάσμα εκπομπής και λήψης όπου επιτρέπει την ανταλλαγή πληροφοριών και επιτυγχάνοντας τις ασύρματες συνδέσεις μεταξύ συσκευών π.χ. κινητά τηλέφωνα.

1.5 ΑΝΑΓΚΑΙΟΤΗΤΑ ΠΡΟΤΕΙΝΟΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ

Έχει μελετηθεί λεπτομερώς η αναγκαιότητα του ανθρώπου για την εξοικονόμηση χρόνου στην καθημερινότητά του. Στην προσπάθειά του να δημιουργήσει εφαρμογές για την εξυπηρέτηση του, με σκοπό την γρηγορότερη και ευκολότερη διαχείριση συσκευών, σε σχέση με τις υφιστάμενες τεχνολογίες, έχει προκύψει αναζήτηση τις οικονομικότερης λύσεις για τις ανάγκες του. Στην παρούσα διατριβή θα παρουσιαστεί, ο σχεδιασμός ενός προηγμένου συστήματος απομακρυσμένης διαχείρισης θερμοκρασίας προσαρμοζόμενων θερμικών ζωνών. Το σύστημα που προτείνεται έχει κύριο γνώμονα την κάλυψη των απαιτήσεων που καλείται να πληροί, αλλά και την οικονομικότερη επιλογή σε σχέση με την αγορά. Αυτό θα επιτευχθεί μέσω της σχεδιαστής και υλοποίησης του συστήματος σε πλατφόρμες, όπου στεγάζονται σε πλακέτες και θα είναι σχεδιασμένες μονάχα για την εξυπηρέτηση αυτής της εφαρμογής.

2 ΚΕΦΑΛΑΙΟ

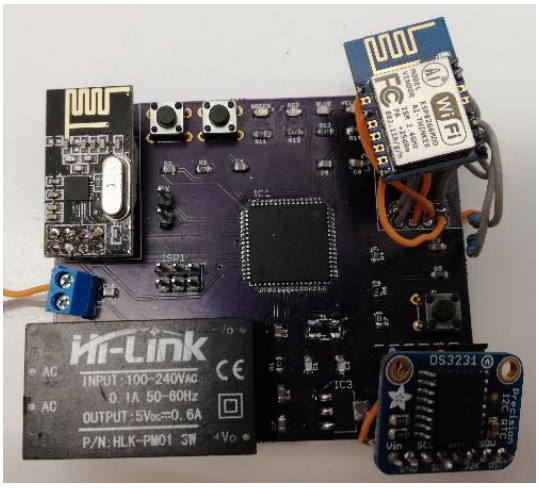
2.1 Το σύστημα

Το σύστημα θα συνδυάζει ασύρματη επικοινωνία, επικοινωνιακά συστήματα αυτοματισμού, ανάλυση τεχνολογικά προηγμένων εφαρμογών, σχεδιασμό και υλοποίηση τον εκάστοτε κυκλωμάτων σε πλακέτα. Μέσω μιας κεντρικής μονάδας όπου θα διαχειρίζεται όλα τα δεδομένα θα πραγματοποιείται ο έλεγχος θερμοκρασίας της εκάστοτε θερμικής Ζώνης ξεχωριστά από έναν, ηλεκτρικό υπολογιστή η τάμπλετ ή κινητό τηλέφωνο, θα είναι δυνατόν να συνδεθεί ο χρήστης με την κεντρική μονάδα που θα είναι υπεύθυνη για τις εισόδους και εξόδους του συστήματος. Η είσοδος του συστήματος θα είναι η πραγματική θερμοκρασία σε online χρόνο για κάθε μία θερμική ζώνη ξεχωριστά. Οι έξοδοι του συστήματος θα είναι συστήματα τα οποία μπορούν να παρέμβουν στην διαδικασία θέρμανσης του εκάστοτε χώρου. Για

παράδειγμα μία ηλεκτροβάνα η οποία απομονώνει το συγκεκριμένο κύκλωμα που επιθυμούμε. Έτσι καταφέρνουμε να έχουμε τον έλεγχο της θερμοκρασίας που έχουμε ορίσει εμείς ανά χώρο. Η καινοτομία της ιδέας είναι ότι η κεντρική μονάδα του συστήματος θα είναι εγκατεστημένοι σε μία πλατφόρμα η οποία βασίζεται στο σκαρίφημα ενός Arduino αλλά με σχεδιασμό και προσαρμογή για τα δικά μας δεδομένα. Συνολικά θα σχεδιαστούν τρεις πλακέτες, Όπου η καθεμία θα αφορά θα έχει προορισμό και προσανατολισμό είχε συγκεκριμένη εφαρμογή.

2.1.1 Κεντρική Μονάδα

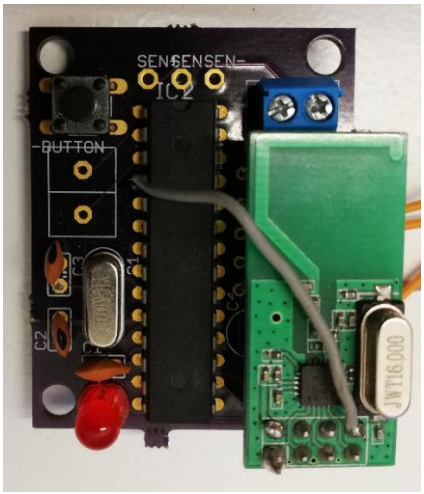
θα είναι υπεύθυνη για το επικοινωνιακό κομμάτι που θα απευθύνεται στο χρήστη και ταυτόχρονα θα μπορεί να επεξεργαστεί και να διαχειριστεί όλα τα δεδομένα του συστήματος με βάση τον προγραμματισμό που θα του έχει δοθεί. Βασικές προϋποθέσεις που πρέπει να υποστηρίζει, είναι μία μονάδα ασύρματης επικοινωνίας ανάμεσα στην κεντρική μονάδα με τις πλατφόρμες, όπου λόγω κόστος, χορό, ευκολίας και ενέργειας, έχει επιλεγεί η μονάδα nRF. Ακόμα ένας μικροελεγκτής ο οποίος θα διαχειρίζεται όλες αυτές απαιτήσεις. Η επικοινωνία ανάμεσα στο σύστημα και τον χρήστη, επιτυγχάνεται μέσω Wi-Fi όπου θα είναι εγκατεστημένο στην κεντρική μονάδα και θα λειτουργεί σαν access point στο οποίο θα μπορεί ο χρήστης να συνδεθεί και να μπορεί να έχει αμφίδρομη επικοινωνία με το σύστημα. Αυτό σημαίνει ότι θα μπορεί να δει την θερμοκρασία της εκάστοτε θερμικές ζώνεις αλλά και να ορίσει τη θερμοκρασία που επιθυμεί σε αυτή.



Εικόνα 6: Κεντρική πλατφόρμα συστήματος

2.1.2 Πλατφόρμα Μέτρησης Θερμοκρασίας

Θα είναι εκείνη η οποία θα δέχεται τα δεδομένα, από τον εκάστοτε αισθητήρας θερμοκρασίας και θα τα μεταβιβάζει στην κεντρική μονάδα σε πραγματικό χρόνο. Ακόμα το κύκλωμα αυτό έχει σχεδιαστεί ώστε να έχει πολύ χαμηλή κατανάλωση για να μπορεί να είναι αυτόνομο ενεργειακά και να μην χρειάζεται ενσύρματη παροχή ενέργειας αλλά να καλύπτει τις ανάγκες του μόνο με μπαταρία. Για το κομμάτι της αμφίδρομης επικοινωνίας με την κεντρική μονάδα έχει επιλεγεί έτσι ώστε να είναι συμβατό και σε αυτή την πλατφόρμα η μονάδα είναι nRF. Επίσης στην πλατφόρμα θα πρέπει να υπάρχει ένας αισθητήρας θερμοκρασίας και ένας μικροελεγκτής ο οποίος θα συλλέγει τα δεδομένα από τον αισθητήρα θερμοκρασίας και μέσω του nRF Θα τα προωθεί στην κεντρική μονάδα.

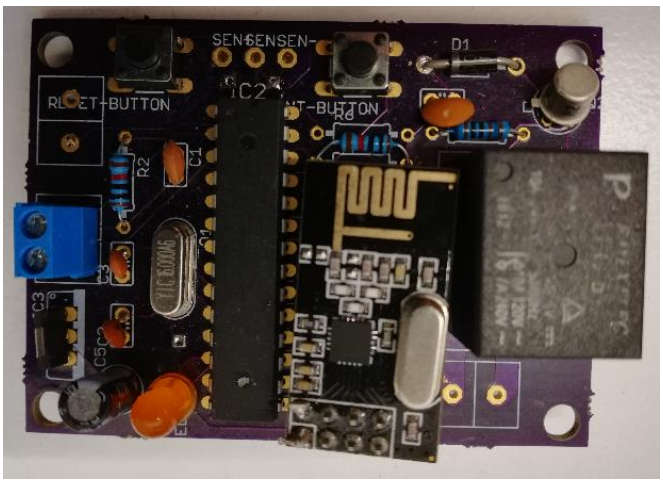


Εικόνα 7: πλατφόρμα εσόδου θερμοκρασίας

2.1.3 Πλατφόρμα επέμβασης σε θερμικές ζώνες

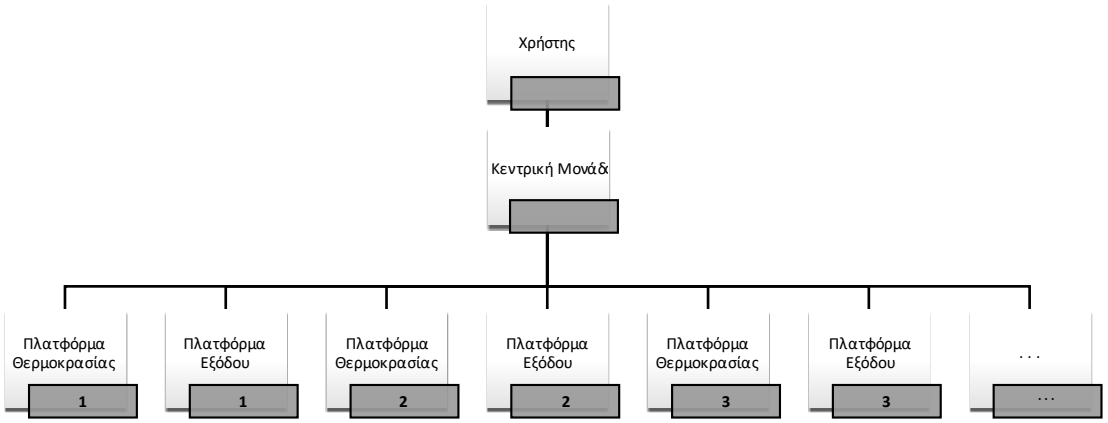
θα δέχεται εντολές από την κεντρική μονάδα και θα είναι υπεύθυνο μέσω τις εξόδους του, η οποία είναι ένας ηλεκτρονόμος, να παρεμβαίνει στην θερμική ζώνη που βρίσκεται στην ίδια Θερμική περιοχή με το κύκλωμα εισόδου θερμοκρασίας. Για την αμφίδρομη επικοινωνία με την κεντρική πλατφόρμα και σε αυτήν την περίπτωση θα είναι η μονάδα nRF. Έξοδος του κυκλώματος θα είναι ένας ηλεκτρονόμος ο οποίος μέσω εξωτερικής πηγής 230volt θα διαχειρίζεται την ηλεκτροβάνα, ή όποιο άλλο μέσο χρειαστεί για την διαχείριση της θερμικής Ζώνης. Επίσης ένας μικροελεγκτής

θα είναι υπεύθυνος για τη μετάβαση των δεδομένων από το nRF μέχρι την όπλιση του ηλεκτρονόμου και την αφοπλίσει του..



Εικόνα 8: πλατφόρμα εξόδου με τι βοήθεια ηλεκτρονόμου

2.1.4Ιεραρχία συστήματος



Σχήμα 5: Ιεραρχία συστήματος

Στο παραπάνω σχήμα βλέπουμε την ιεραρχία της δομής του συστήματος. Ο χρήστης έχει αμφίδρομη επικοινωνία με την κεντρική μονάδα μέσω της οποίας μπορεί να πληροφορηθεί για την κατάσταση του συστήματος. Μέσω μιας ηλεκτρονικής συσκευής που έχει επιλέξει θα μπορεί να δει τη θερμοκρασία των θερμικών ζωνών

και ποιες από αυτές είναι σε λειτουργία και εάν έχουν πιάσει την επιθυμητή θερμοκρασία που αυτός έχει ορίσει.

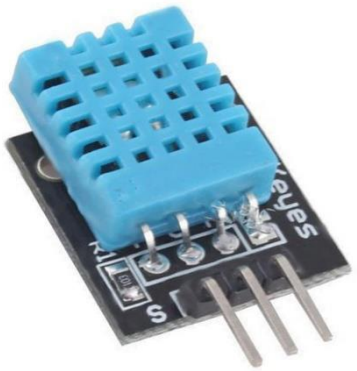
Στη συνέχεια η κεντρική μονάδα επεξεργάζεται όλα αυτά τα δεδομένα και συλλέγει πληροφορίες από τις πλατφόρμες θερμοκρασίας συγκρίνοντάς με το Επιθυμητό αποτέλεσμα που ορίζει ο χρήστη, προβαίνει σε ενέργειες είτε να ενεργοποιήσει κάποια πλατφόρμα εξόδου είτε να την απενεργοποίηση. Βασική προϋπόθεση του συστήματος είναι σε κάθε θερμική ζώνη να υπάρχει τουλάχιστον μία πλατφόρμα θερμοκρασίας και μία πλάτη φόρμα εξόδου. Σε περίπτωση που επιθυμεί παραπάνω απαιτήσεις σε σχέση με τις πλατφόρμες, ο μόνος περιορισμός αφορά τι δυνατότητες της κεντρικής μονάδας, προσεγγιστικά η κεντρική μονάδα μπορεί εύκολα να συμβιώσει με πάνω από 40 βοηθητικές πλατφόρμες.

2.2 Περιγραφή αισθητήρα

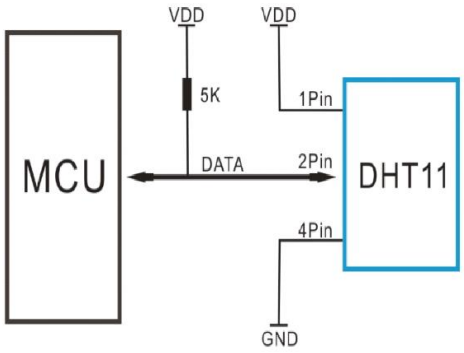
Ο αισθητήρας που χρησιμοποιείται είναι ο DHT11. ο συγκεκριμένος αισθητήρας έχει τη δυνατότητα να μετράει θερμοκρασίες από 0 έως 50 βαθμούς Κελσίου , αλλά και την υγρασία του χώρου. Στην παρούσα εφαρμογή του θα χρησιμοποιείτε μόνο ως αισθητήρας θερμοκρασίας. Τα πιο σημαντικά χαρακτηριστικά του είναι η σταθερότητα του, η σχετικά καλή ακρίβεια και η πλήρης συμβατότητα με το σύστημα μας. Στο παρακάτω Σχήμα 6, θα δούμε ότι ενσωματώνει ένα κύκλωμα προσαρμογής ώστε να μπορεί να συνδεθεί απευθείας στο σύστημα.

Πίνακας 3: Χαρακτηριστικά αισθητήρα θερμοκρασίας του DHT11

Χαρακτηριστικά αισθητήρα θερμοκρασίας DHT11					
Όνομα	Δυνατότητες μέτρησης	Ακρίβεια Υγρασίας	Ακρίβεια θερμοκρασίας	Ανάλυση	Πριν
DHT11	20-90% / RH 0-50 °C	±5%RH	±2°C	1	4



Εικόνα 9: Αισθητήρα θερμοκρασίας DHT11



Σχήμα 6: Ηλεκτρική σύνδεση αισθητήρα θερμοκρασίας DHT11

Βασικά χαρακτηριστικά του αισθητήρα τα οποία μας βοηθάνε στον προγραμματισμό αλλά και στην υλοποίηση του συστήματος. στον πίνακα 9 βλέπουμε όπως δυνατότητες μέτρησης, ακρίβεια και ανάλυση του αισθητήρα.

Στον πίνακα 4 και 5 είναι χαρακτηριστικά τα οποία μας βοηθάνε για τον προγραμματισμό αλλά και της ηλεκτρικές απαιτητής για τι συνδεσιμότητα του αισθητήρα.

Parameter	Min.	Typical	Max.	Unit
Working voltage	3	5	5.5	VDC
Working current（VCC=5V, T=25℃）	0.5	-	2.5	mA
Sampling interval	1	-	-	s

Πίνακας 4: Ηλεκτρικές ανοχές του DHT11

Χαρακτηριστικά αισθητήρα θερμοκρασίας

Όνομα	Δυνατότητες μέτρησης	ακρίβεια Υγρασίας	ακρίβεια θερμοκρασίας	Ανάλυση	Π
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	

Parameter	Min.	Typical	Max.	Unit
Accuracy (25℃)	-	±4	-	%RH
Accuracy (0-50℃)	-	-	±5	%RH
Measurement range (25℃)	20	-	95	%RH
Response time: 1/e（63%）25℃, 1m/s air	6	10	15	s

Πίνακας 5: Ανοχές απόκρισης του DHT11

Χαρακτηριστικά αισθητήρα θερμοκρασίας					
Όνομα	Δυνατότητες μέτρησης	ακρίβεια Υγρασίας	ακρίβεια θερμοκρασίας	Ανάλυση	Πριν
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4

Ο κατασκευαστής του αισθητήρα δίνει σε μορφή open τον κώδικα ο οποίος μπορεί να βοηθήσει για τον προγραμματισμό του αισθητήρα στο σύστημα. Παρακάτω θα δούμε εντολές αλλά και βιβλιοθήκες οι οποίες καλούνται για την αναγνώριση και την αποκωδικοποίηση των δεδομένων του αισθητήρα, έτσι ώστε να είναι αντιληπτές από το σύστημα.

```
//Celsius to Fahrenheit conversion
 double Fahrenheit(double celsius)
 {
   return 1.8 * celsius + 32;
 }
 //Celsius to Kelvin conversion
 double Kelvin(double celsius)
 {
   return celsius + 273.15;
 }
 // dewPoint function NOAA
 // reference: http://wahiduddin.net/calc/density_algorithms.htm
 double dewPoint(double celsius, double humidity)
 {
   double A0= 373.15/(273.15 + celsius);
   double SUM = -7.90298 * (A0-1);
   SUM += 5.02808 * log10(A0);
   SUM += -1.3816e-7 * (pow(10, (11.344*(1-1/A0)))-1) ;
   SUM += 8.1328e-3 * (pow(10,(-3.49149*(A0-1)))-1) ;
   SUM += log10(1013.246);
   double VP = pow(10, SUM-3) * humidity;
   double T = log(VP/0.61078);   // temp var
   return (241.88 * T) / (17.558-T);
 }

 // delta max = 0.6544 wrt dewPoint()
 // 5x faster than dewPoint()
 // reference: http://en.wikipedia.org/wiki/Dew_point
 double dewPointFast(double celsius, double humidity)
 {
   double a = 17.271;
   double b = 237.7;
   double temp = (a * celsius) / (b + celsius) +
log(humidity/100);
   double Td = (b * temp) / (a - temp);
   return Td;
 }
```

```
#include <dht11.h>
dht11 DHT11;
#define DHT11PIN 14
void setup()
{
Serial.begin(115200);
Serial.println("DHT11 TEST PROGRAM ");
Serial.print("LIBRARY VERSION: ");
Serial.println(DHT11LIB_VERSION);
Serial.println();
}
void loop()
{
Serial.println("\n");
int chk = DHT11.read(DHT11PIN);
Serial.print("Read sensor: ");
switch (chk)
{
  case 0: Serial.println("OK"); break;
  case -1: Serial.println("Checksum error"); break;
  case -2: Serial.println("Time out error"); break;
  default: Serial.println("Unknown error"); break;
}

Serial.print("Humidity (%): ");
Serial.println((float)DHT11.humidity, 2);

Serial.print("Temperature (oC): ");
Serial.println((float)DHT11.temperature, 2);

Serial.print("Temperature (oF): ");
Serial.println(Fahrenheit(DHT11.temperature), 2);

Serial.print("Temperature (K): ");
Serial.println(Kelvin(DHT11.temperature), 2);

Serial.print("Dew Point (oC): ");
Serial.println(dewPoint(DHT11.temperature, DHT11.humidity));

Serial.print("Dew PointFast (oC): ");
Serial.println(dewPointFast(DHT11.temperature, DHT11.humidity));
```

```
delay(2000);
 }
```

2.3 Περιγραφή Arduino

2.3.1 Περιγραφή κεντρικής μονάδας

Για να ξεκινήσει ο σχεδιασμός της κεντρικής μονάδας, θα πρέπει να προσδιοριστούν οι απαιτήσεις για τις οποίες προορίζεται. Οι απαιτήσεις που καλείτε να φέρει σε πέρας, έχουν να κάνουν με τη συνδεσιμότητα, όσο με τις άλλες δύο πλατφόρμες (πλατφόρμα εισόδου θερμοκρασίας, πλατφόρμα εξόδου ηλεκτρονόμος), αλλά και με την αμφίδρομη επικοινωνία με το χρήστη. Ακόμα πρέπει να υπάρχει χώρος αποθήκευσης για τον κώδικα και τα δεδομένα, αλλά και επεξεργαστική ισχύ. Θα πρέπει να είναι συμβατό με όλες τις λειτουργίες που απαιτεί η εφαρμογή, όπως να μπορεί να διαχειριστεί τη μονάδα nRF, τη μονάδα wi-fi αλλά να μπορεί να υποστηρίξει κι άλλες εισόδους και εξόδους για μελλοντικές βελτιώσεις και εάν απαιτηθεί από το χειριστή μία επιμέρους οθόνη και τη διαχείριση όλου του συστήματος.

Με βάση τα παραπάνω έχει επιλεχθεί ο μικροελεγκτής ολοκληρωμένου κυκλώματος AVR της εταιρείας microchip (ATMEL) μοντέλο atMega2561-16au. Για την αμφίδρομη επικοινωνία ανάμεσα σε πλατφόρμες έχει επιλεχθεί η μονάδα nRF 24L01. Γιατί τη αμφίδρομη επικοινωνία ανάμεσά της κεντρικής μονάδας και του χρήστη είναι η μονάδα WiFi ESP8266MOD.

2.3.1.1 Περιγραφή atMega2561-16au

ATMega 2561-16au είναι ο αντικαταστάτης του προκατόχου του 2560 ο οποίος χρησιμοποιείται στο Arduino Mega. Διαθέτει συνολικά 64 PIN πράγμα που σημαίνει ότι έχει πολλές επιλογές συνδεσιμότητας. Είναι ένας από τους πιο διαδεδομένους μικροελεγκτές ο οποίος χρησιμοποιείται για ευρύ εφαρμογές. Το μικρό του μέγεθος είναι ωφέλιμο για κατασκευές που απαιτούν μικρό χώρο.

Εικόνα 10: Ο μικροεπεξεργαστής ATMega 2561-16au

Στον παρακάτω πίνακα 6 θα δούμε τα τεχνικά χαρακτηριστικά του ATMega 2561-16au. Είναι απαραίτητο πριν από τον σχεδιασμό του κυκλώματος να μελετηθούν πολύ καλά τα τεχνικά χαρακτηριστικά των εξαρτημάτων έτσι ώστε να υπάρχει εξασφάλιση συμβατότητας και συνδεσιμότητας.

Specifications	
Product Attribute	Attribute Value
Manufacturer:	Microchip
Product Category:	8-bit Microcontrollers - MCU
RoHS:	Details
Mounting Style:	SMD/SMT
Package/Case:	TQFP-64
Series:	ATmega256x
Core:	AVR
Data Bus Width:	8 bit
Maximum Clock Frequency:	16 MHz
Program Memory Size:	256 kB
Data RAM Size:	8 kB
Operating Supply Voltage:	4.5 V to 5.5 V
Minimum Operating Temperature:	- 40 C

Maximum Operating Temperature:	+ 85 C
Interface Type:	2-Wire, SPI, USART
Packaging:	Tray
Height:	1 mm
Length:	14 mm
Product:	MCU
Program Memory Type:	Flash
Width:	14 mm
Brand:	Microchip Technology / Atmel
Data RAM Type:	SRAM
Data ROM Size:	4 kB
Data ROM Type:	EEPROM
Moisture Sensitive:	Yes
Number of Timers/Counters:	6 Timer
Processor Series:	megaAVR
Product Type:	8-bit Microcontrollers - MCU
Factory Pack Quantity:	90
Subcategory:	Microcontrollers - MCU
Supply Voltage - Max:	5.5 V
Supply Voltage - Min:	4.5 V
Unit Weight:	360,600 mg

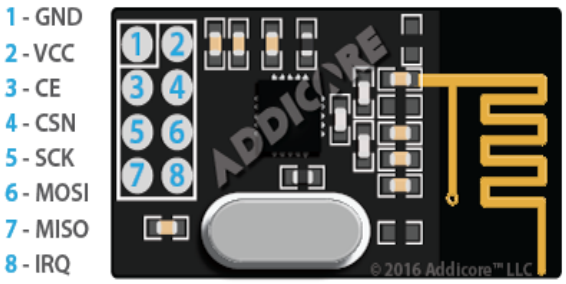
Πίνακας 6: Χαρακτηριστικά του μικροεπεξεργαστής ATMega 2561-16au

2.3.1.2nRF 24L01

Το nRF 24L01 είναι το πιο διαδεδομένο μέσω ασύρματης αμφίδρομης επικοινωνίας. Βασικά του χαρακτηριστικά είναι η μικρή κατανάλωση ενέργειας ο μικρός χώρος που καταβάλλει και η εύκολη συνδεσιμότητα του πλατφόρμες τύπου Arduino αλλά και γενικά με μικροεπεξεργαστές. Η συχνότητα μετάδοσης των δεδομένων είναι τα 2,4 GHz Με 125 κανάλια εναλλαγής.

Εικόνα 11:Το nrf 24L01

Στο παρακάτω Σχήμα 7 απεικονίζονται ταpin συνδεσιμότητας του nRF 24L01αλλά και ο προσανατολισμός τους στην πλακάτα.



Σχήμα 7: Η ονομασία των pin του nrf 24L01

Στον παρακάτω πίνακα 7, αναγράφονται ταpin και επεξηγείται τι ακριβώς είναι το καθένα, ποιο από αυτά είμαι τροφοδοσία και ποιο επικοινωνία με το σύστημα μας.

Χαρακτηριστικά nRF 24L01			
Pin Number	Pin Name	Abbreviation	Function

1	Ground	Ground	Connected to the Ground of the system
2	Vcc	Power	Powers the module using 3.3V
3	CE	Chip Enable	Used to enable SPI communication
4	CSN	Ship Select Not	This pin has to be kept high always, else it will disable the SPI
5	SCK	Serial Clock	Provides the clock pulse using which the SPI communication works
6	MOSI	Master Out Slave In	Connected to MOSI pin of MCU, for the module to receive data from the MCU
7	MISO	Master In Slave Out	Connected to MISO pin of MCU, for the module to send data from the MCU
8	IRQ	Interrupt	It is an active low pin and is used only if interrupt is required

Πίνακας 7: Η ονομασία των pin του nrf 24L01

Στον παρακάτω πίνακα 8, δίνονται όλα τα βασικά χαρακτηριστικά του nRF που είναι χρήσιμα για την ένταξη του στο σύστημα μας.

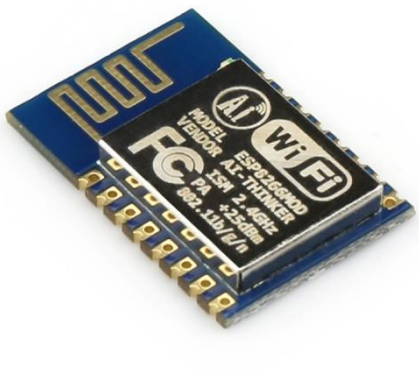
nRF24L01 Features	
RF transceiver Module	2.4GHz
Operating Voltage	3.3V
Nominal current	50mA

Range	50 – 200 feet
Operating current	250mA (maximum)
Communication Protocol	SPI
Baud Rate	250 kbps - 2 Mbps
Channel Range	125
Maximum Pipelines/node	6
Low cost wireless solution	

Πίνακας8: Τα χαρακτηριστικά του nrf 24L01

2.3.1.3 Περιγραφή ESP8266MOD

Το ESP8266MOD είναι ένα wi-fi μέσω μεταδόσεις το οποίο έχει επικρατήσει λόγω του μικρού όγκου και του μικρού κόστους αλλά και της συμβατότητας με πλατφόρμες όπως το Arduino αλλά και γενικότερα με μικροεπεξεργαστές



Εικόνα 12: Το ESP8266MOD

Τα βασικά χαρακτηριστικά του ESP8266MOD τα οποία μας βοηθούν για την συμβατότητα του με το σύστημα μας και αν αυτό πληροί όλες τις προϋποθέσεις.

Features:

• 802.11 b/g/n

• Integrated low power 32-bit MCU

• Integrated 10-bit ADC

• Integrated TCP/IP protocol stack

• Integrated TR switch, balun, LNA, power amplifier and matching network

• Integrated PLL, regulators, and power management units

• Supports antenna diversity

• WiFi 2.4 GHz, support WPA/WPA2

• Support STA/AP/STA+AP operation modes

• Support Smart Link Function for both Android and iOS devices

• SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO

• STBC, 1x1 MIMO, 2x1 MIMO

• A-MPDU & A-MSDU aggregation & 0.4s guard interval

• Deep sleep power <10uA, Power down leakage current < 5uA

• Wake up and transmit packets in < 2ms

• Standby power consumption of < 1.0mW (DTIM3)

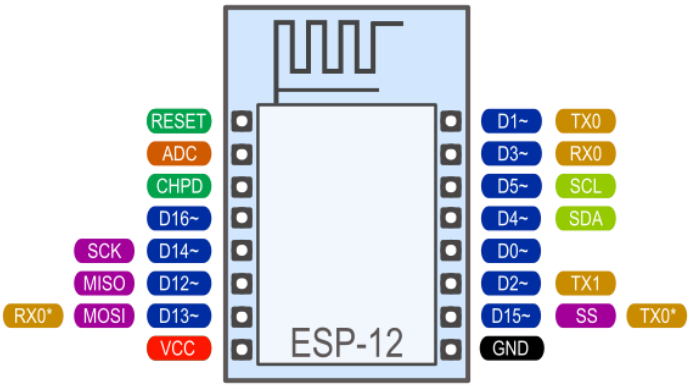
• +20 dBm output power in 802.11b mode

• Operating temperature range -40C ~ 125C

• FCC, CE, TELEC, WiFi Alliance, and SRRC certified

Operating Voltage 3.0~3.6V

Operating Current Average value: 80mA



Σχήμα 8: Η ονομασία των pintou ESP8266MOD

2.3.2 Περιγραφή πλακέτας εισόδου θερμοκρασίας

2.3.2.1 Περιγραφή μικροεπεξεργαστή **ATMega 168-20p**

Για τι πλατφόρμα εισόδου θερμοκρασίας έχει επιλεχθεί ο μικροεπεξεργαστής ATMega 168-20p της εταιρείας ATMEL. Είναι αντιστοίχως μικροεπεξεργαστής σαν αυτόν που χρησιμοποιεί η πλατφόρμα Arduino.

Εικόνα 13: Ο μικροελεγκτής ATMega 168-20p

Στον παρακάτω πίνακα 9 θα δούμε τα χαρακτηριστικά του μικροεπεξεργαστή αλλά και όλες τις ηλεκτρικές ανοχές στης οποίες που μπορεί να εργαστεί.

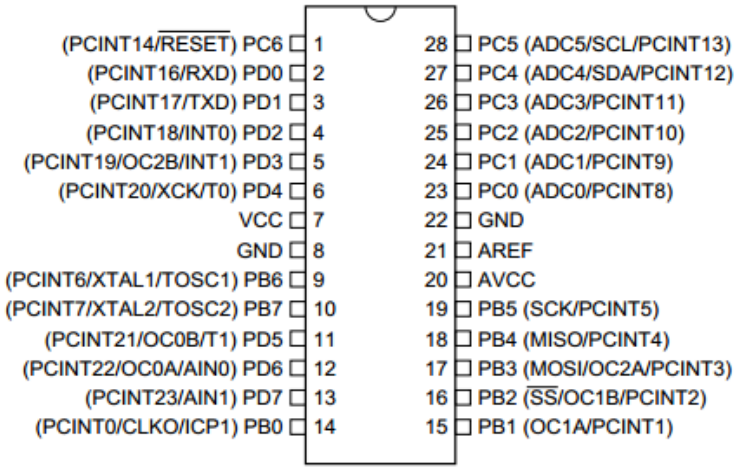
atmega168-20p Specifications	
Physical	
Case/Package	PDIP

Contact Plating	Tin
Mount	Through Hole
Number of Pins	28
Technical	
Access Time	20 µs
Address Bus Width	8 b
Core Architecture	AVR
Data Bus Width	8 b
Density	16 kB
Frequency	20 MHz
Interface	I2C , SPI , UART , USART , Serial , 2-Wire
Max Frequency	20 MHz
Max Operating Temperature	85 °C
Max Supply Voltage	5.5 V
Memory Size	16 kB
Memory Type	FLASH , EEPROM
Min Operating Temperature	-40 °C
Min Supply Voltage	2.7 V
Number of ADC Channels	6
Number of Bits	8
Number of I/Os	23
Number of I2C Channels	1
Number of PWM Channels	6
Number of Programmable I/O	23
Number of SPI Channels	1
Number of Timers/Counters	3
Number of UART Channels	0
Number of USB Channels	0
Oscillator Type	Internal
Peripherals	Brown-out Detect/Reset , POR , PWM , WDT
RAM Size	1 kB
Watchdog Timer	Yes
Dimensions	
Height	4.572 mm
Length	34.8 mm

Width	7.49 mm
Compliance	
Lead Free	Lead Free
REACH SVHC	No SVHC
Radiation Hardening	No
RoHS	Non-Compliant

Πίνακας 9: Χαρακτηριστικά του μικροελεγκτης ATMega 168-20p

Στο παρακάτω σχήμα παρουσιάζεται το σκαρίφημα του επεξεργαστή με την ακριβή αντιστοιχία τον pin σε σχέση σε σχέση με την ονομασία τους.



Σχήμα 9: Η ονομασία των pin του μικροελεγκτις ATMega 168-20p

2.3.2.2 Περιγραφή nRF 24L01

Το nRF 24L01 είναι ίδιο με αυτό που αναλύθηκε σε προηγούμενη ενότητα (Βλέπε περιγραφή κεντρικής πλατφόρμας)

2.3.2.3Περιγραφή ESP8266MOD

Το ESP8266MOD είναι ίδιο με αυτό που αναλύθηκε σε προηγούμενη ενότητα (Βλέπε περιγραφή κεντρικής πλατφόρμας)

2.3.3 Περιγραφή πλακέτας εξόδου

2.3.3.1 Περιγραφή μικροεπεξεργαστή ATMega 168-20p

Το ATMega 168-20p είναι ίδιο με αυτό που αναλύθηκε σε προηγούμενη ενότητα (Βλέπε περιγραφή Πλακέτα σε έξοδο)

2.3.3.2 Περιγραφή nRF 24L01

Το nRF 24L01 είναι ίδιο με αυτό που αναλύθηκε σε προηγούμενη ενότητα (Βλέπε περιγραφή κεντρικής πλατφόρμας)

2.3.3.3 περιγραφή ESP8266MOD

Το ESP8266MOD είναι ίδιο με αυτό που αναλύθηκε σε προηγούμενη ενότητα (Βλέπε περιγραφή κεντρικής πλατφόρμας)

2.3.3.4 ηλεκτρονόμος

Ο ηλεκτρονόμος που επιλεχθείτε είναι SPDT 5V 10A της εταιρίας RAYEX ELECTRONIC .

Χαρακτηριστικά

Manufacturer: RAYEX ELECTRONIC

Relay type: electromagnetic

Contacts configuration: SPDT

Rated coil voltage: 5V DC

AC contacts rating: 10 A / 120 VAC

DC contacts rating: 10 A / 24 VDC

Contact current max.: 10A

Switched voltage: max 240V AC, max 100V DC

Relay variant: miniature

Mounting: PCB

Operate time: 8ms

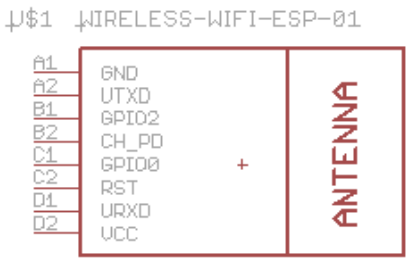
Dimensions: 19.5 x 15.8 x 15mm

2.4.1 Περιγραφή συνδέσεων κεντρικές μονάδας

Συλλέγοντας όλες τις πληροφορίες που αναφέραμε παραπάνω γιατί συνδεσιμότητα αλλά και τη συμβατότητα των εξαρτημάτων θα προβούμε στον σχεδιασμό της κεντρικής μονάδας. Κύρια προτεραιότητα είναι η ορθολογική συνδεσμολογία των εξαρτημάτων. Θα επιλέξουμε ένα πρόγραμμα το οποίο θα μας παρέχει το κατάλληλο περιβάλλον για τον σχεδιασμό της κεντρικής μονάδας, με δυνατότητα να μπορούμε να μεταβιβάσουμε το σχέδιό μας σε ηλεκτρονική πλακέτα, χρησιμοποιούμε ένα από τα πιο σύγχρονα προγράμματα, το Eagle 6.6.6 Professional το οποίο είναι πλήρης είναι συμβατό με εταιρείες τυπώσεις ηλεκτρικών κυκλωμάτων. Με τη χρήση αυτού πετυχαίνουμε την ορθολογική συνδεσμολογία των εξαρτημάτων.

Αρχικά εισάγουμε τα κύρια εξαρτήματα που αποτελούν την κεντρική μονάδα. Με ιδιαίτερη προσοχή στην αντιστοιχία του πραγματικού εξαρτήματος σε σχέση με αυτού του προγράμματος. Παρακάτω απεικονίζονται σε μορφή symmetric εξαρτήματα.

- EAGLE Schematic του Wi Fi ESP8266MOD



Σχήμα10: Schematic του Wi Fi ESP8266MOD

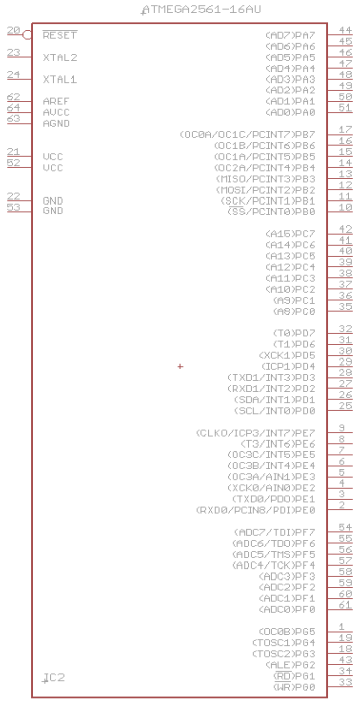
- EAGLE Schematic του nRF 24L01

Σχήμα11: Schematic του nRF 24L01

- EAGLE Schematic του POWER SUPPLY

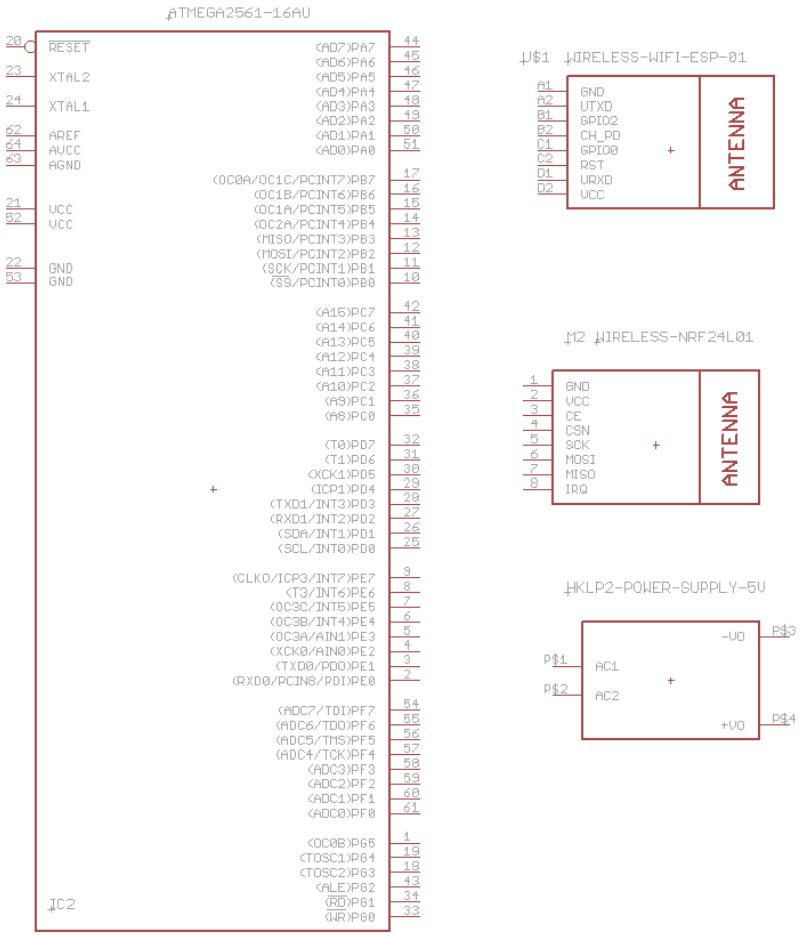
Σχήμα12: Schematic του POWER SUPPLY

- EAGLE Schematic του atMega2561-16au



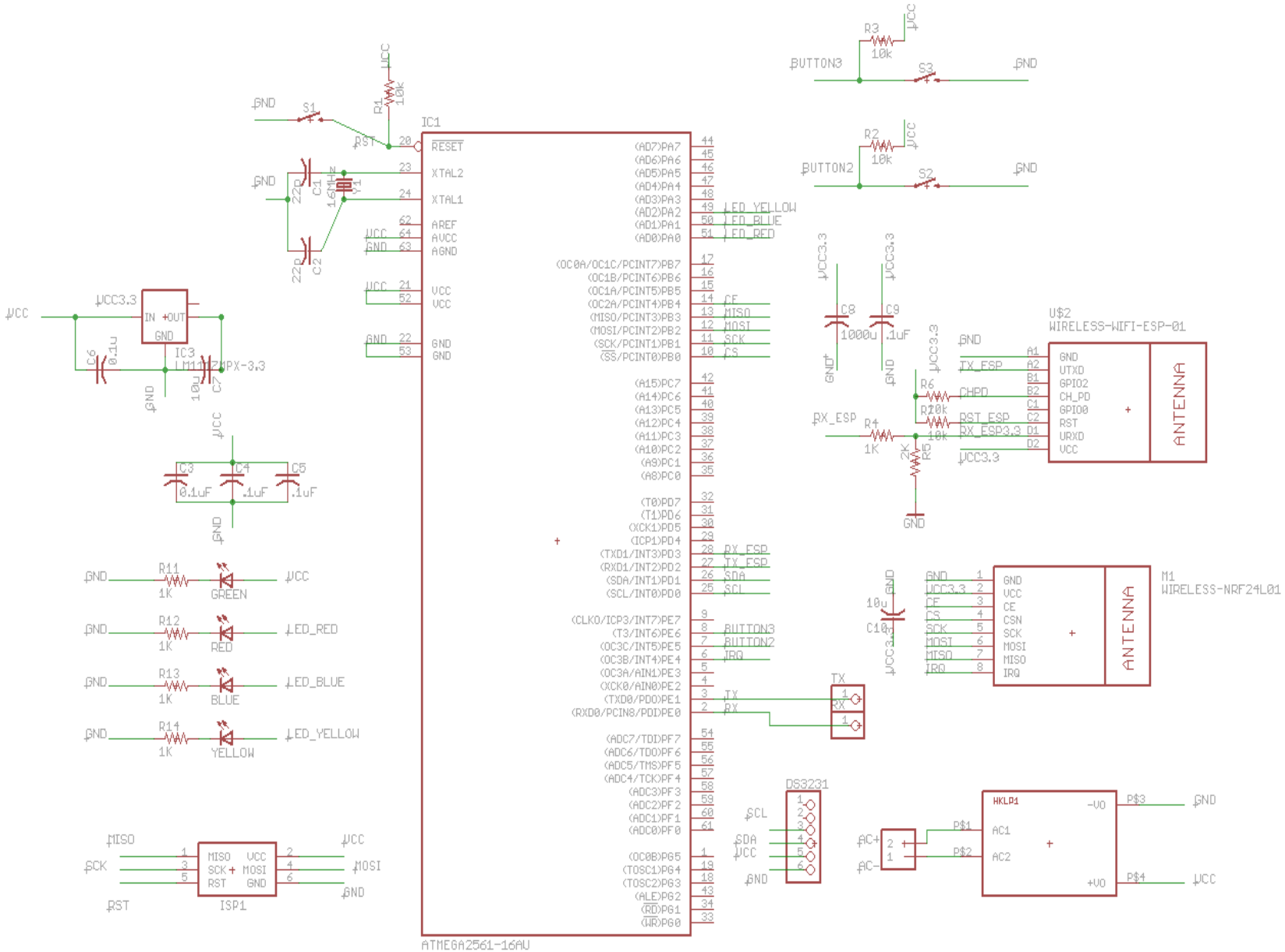
Σχήμα 13: Schematic του atMega2561-16au

Στο παρακάτω Σχήμα14, παρατηρούμε ότι έχουμε εισάγει όλα τα βασικά εξαρτήματα και είμαστε έτοιμοι να ξεκινήσουμε τη διαδικασία συνδεσμολογίας μεταξύ τους, πάντα με βάση τα χαρακτηριστικά και τις παραμέτρους που έχει ορίσει ο κατασκευαστής



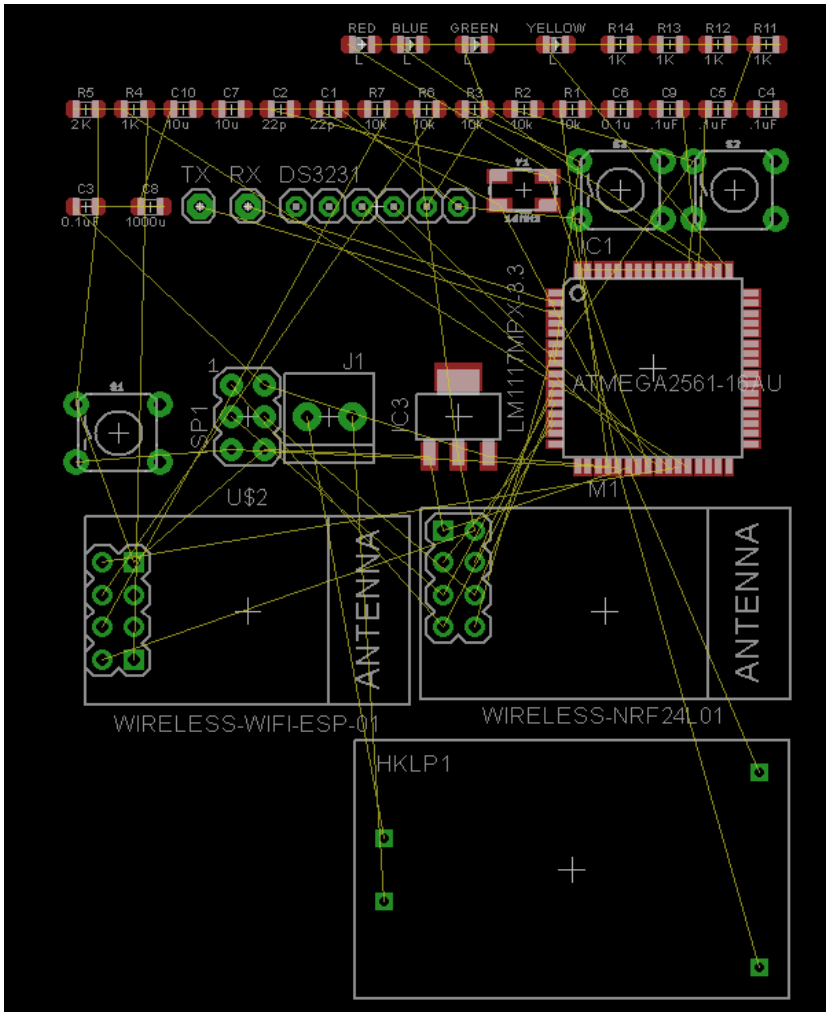
Σχήμα 14: Τα βασικότερα Schematic για την κεντρική πλακάτα

Όπως φαίνεται στο Σχήμα 15, παρακάτω έχει ολοκληρωθεί η συνδεσμολογία των εξαρτημάτων με τη μέθοδο, που έχει τη δυνατότητα το πρόγραμμα, ονομασίας στα δύο άκρα των αγώγιμων αγωγών επιτυγχάνοντας ένα πιο καθαρό σχέδιο. Επίσης έχουν εισαχθεί και άλλα βοηθητικά κυκλώματα όπως, κύκλωμα χρονισμού του μικροελεγκτή με χρήση κρυστάλλου, βοηθητικό κύκλωμα σταθεροποίησης αλλά και κύκλωμα για την πτώση τάσης στα 3,3 volt όπου λειτουργεί όλη η πλατφόρμα και τέλος προσθήκη σε διάφορα PINσυνδεσιμότητας.


IC1
ATMEGA2561-16AU
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
(AD7)PA7
(AD6)PA6
(AD5)PA5
(AD4)PA4
(AD3)PA3
(AD2)PA2
(AD1)PA1
(AD0)PA0
LED_YELLOW
LED_BLUE
LED_RED
(OC0A/OC1C/PCINT7)PB7
(OC1B/PCINT6)PB6
(OC1A/PCINT5)PB5
(OC2A/PCINT4)PB4
(MISO/PCINT3)PB3
(MOSI/PCINT2)PB2
(SCK/PCINT1)PB1
(SS/PCINT0)PB0
CE
MISO
MOSI
SCK
CS
(A15)PC7
(A14)PC6
(A13)PC5
(A12)PC4
(A11)PC3
(A10)PC2
(A9)PC1
(A8)PC0
(T0)PD7
(T1)PD6
(XCK1)PD5
(ICP1)PD4
(TXD1/INT3)PD3
(RXD1/INT2)PD2
(SDA/INT1)PD1
(SCL/INT0)PD0
RX_ESP
TX_ESP
SDA
SCL
(CLKO/ICP3/INT7)PE7
(T3/INT6)PE6
(OC3C/INT5)PE5
(OC3B/INT4)PE4
(OC3A/AIN1)PE3
(XCK0/AIN0)PE2
(TXD0/PDO)PE1
(RXD0/PCIN8/PDI)PE0
BUTTON3
BUTTON2
IRQ
TX
RX
(ADC7/TDI)PF7
(ADC6/TDO)PF6
(ADC5/TMS)PF5
(ADC4/TCK)PF4
(ADC3)PF3
(ADC2)PF2
(ADC1)PF1
(ADC0)PF0
(OC0B)PG5
(TOSC1)PG4
(TOSC2)PG3
(ALE)PG2
(RD)PG1
(WR)PG0
S1
RST
R1
10k
C1
22p
16MHz
Y1
C2
22p
S2
R2
10k
BUTTON2
S3
R3
10k
BUTTON3
IC3
LM1117-3.3
VCC3.3
C6
0.1u
C7
10u
C3
0.1uF
C4
.1uF
C5
.1uF
C8
1000u
C9
.1uF
U$2
WIRELESS-WIFI-ESP-01
GND
UTXD
GPIO2
CH_PD
GPIO0
RST
URXD
VCC
ANTENNA
TX_ESP
CHPD
RST_ESP
RX_ESP3.3
R4
1K
R5
2K
R6
10k
R7
10k
M1
WIRELESS-NRF24L01
CE
CSN
SCK
MOSI
MISO
IRQ
C10
10u
R11
1K
GREEN
R12
1K
RED
R13
1K
BLUE
R14
1K
YELLOW
ISP1
MISO
SCK
RST
VCC
MOSI
GND
TX
RX
DS3231
SCL
SDA
VCC
GND
HKLP1
AC+
AC-
AC1
AC2
-VO
+VO
P$1
P$2
P$3
P$4

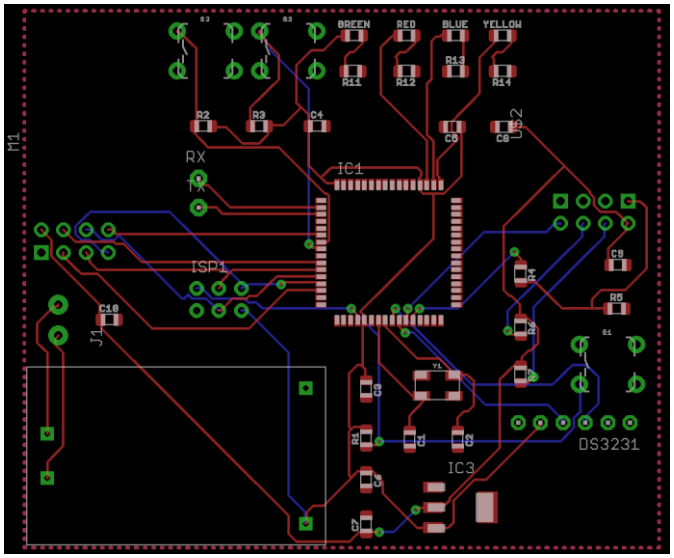
Σχήμα 15: σύνδεση υλικών στο Schematic

Μετά την ολοκλήρωση του σχεδιασμού του κυκλώματος σε μορφή EAGLE Schematic, θα μεταβούμε στο στάδιο του EAGLE Board. Όπως θα δούμε στο παρακάτω Σχήμα 16, απεικονίζεται σε πραγματικό μέγεθος αλλά και σε πραγματικό χώρο η πλακέτα της κεντρικής μονάδας. Σε αυτό το στάδιο το πρόγραμμα έχει κάνει μία τυχαία τοποθέτηση των υλικών όπου παρατηρείται η διασταύρωση αγώγιμων δρόμων(κίτρινες διασυνδέσεις) όπου σίγουρα δεν είναι αυτό που επιθυμούμε. Σε αυτό το σημείο καλούμαστε να τοποθετήσουμε τα υλικά με τη δομή που εμείς επιθυμούμε πάνω στην πλακέτα λαμβάνοντας πάντα υπόψη μας τις αλληλεπιδράσεις μεταξύ των υλικών. Επίσης έχουμε τη δυνατότητα να γίνει τυπώσει Πλακέτας διπλής όψης. αυτό σημαίνει ότι έχουμε μικρότερο χώρο αλλά και μεγαλύτερη ευελιξία στον σχηματισμό της πλακέτας.



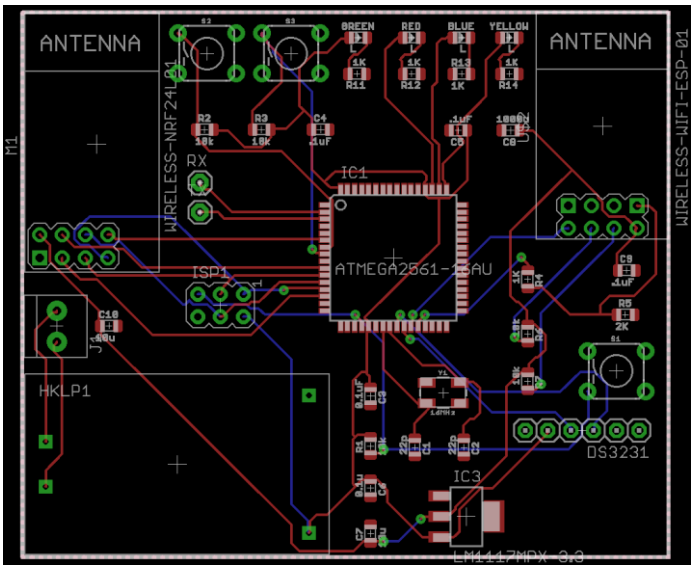
Σχήμα 16: Τυχαία τοποθέτηση εξαρτημάτων στο EAGLE Board

Λαμβάνοντας υπόψη όλες τις παραπάνω πληροφορίες έχει ολοκληρωθεί η διαδικασία όπου απαιτείται για το τελικό αποτέλεσμα των αγώγιμων διαδρομών της πλακέτας. Στο παρακάτω Σχήμα 17, φαίνεται η δομή και ο προσανατολισμός των εξαρτημάτων σε μία πλακέτα διπλής όψης. οι μπλε αγώγιμες διαδρομές είναι η πίσω πλευρά της πλακέτας ενώ η κόκκινη αγώγιμη διαδρομή είναι η πάνω πλευρά της πλακέτας όπου τοποθετούνται και τα κυρίως υλικά.



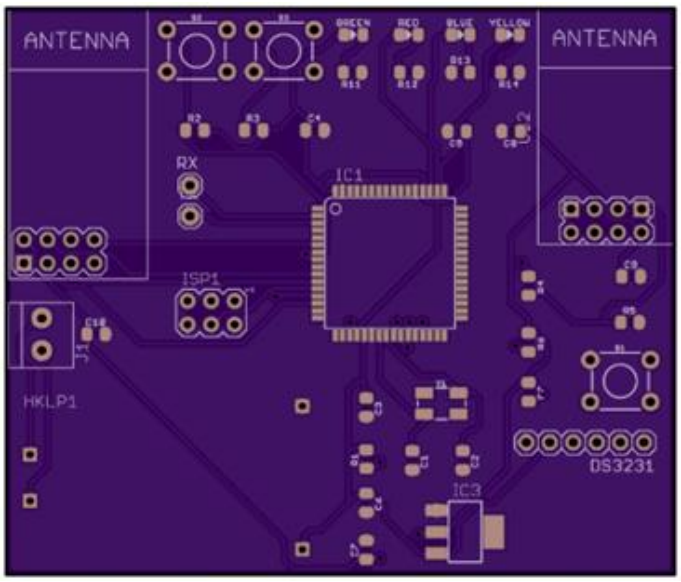
Σχήμα 17:Ολοκλήρωση δομής πλακέτας στο EAGLE Board

Στο παρακάτω Σχήμα 18 βλέπουμε την προσομοίωση και των υλικών με θεωρητική τοποθέτηση τον ANT1 ESP8266MOD και nRF24L01.



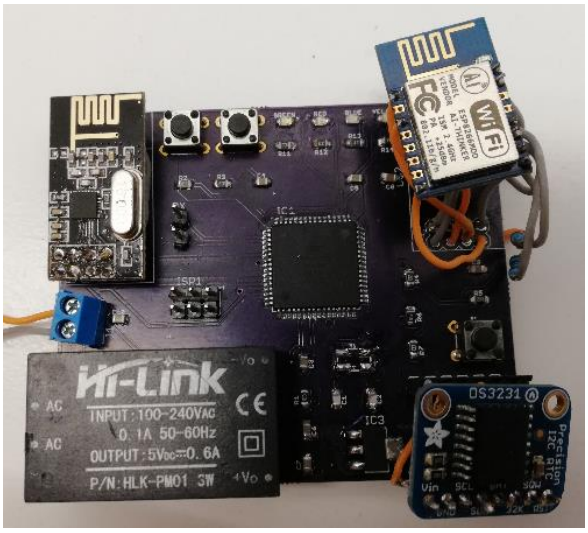
Σχήμα 18:χάραξη εξαρτημάτων RFστο EAGLE Board

Έπειτα απεικονίζεται η πραγματική πλακέτα όπου έχει τυπωθεί σε εξειδικευμένη μονάδα τυπώσεις ηλεκτρικών πλακετών, με την ιδιαιτερότητα ειδικής επίστρωσης για την προστασία των αγώγιμων δρόμων. Σε αυτό το στάδιο πριν την τοποθέτηση των υλικών με τη βοήθεια ενός ωμόμετρου θα γίνει επαλήθευση της αγωγιμότητα για κάθε γραμμής του κυκλώματος, αλλά και για τυχόν ανεπιθύμητο βραχυκύκλωμα με άλλους αγωγούς.



Σχήμα 19: Η πραγματική πλακέτα κεντρικής πλατφόρμας

Στην παρακάτω εικόνα 14 έχει γίνει η τοποθέτηση όλων των υλικών και έχει γίνει η δοκιμή και όλα τα τεστ για τη σωστή λειτουργία.

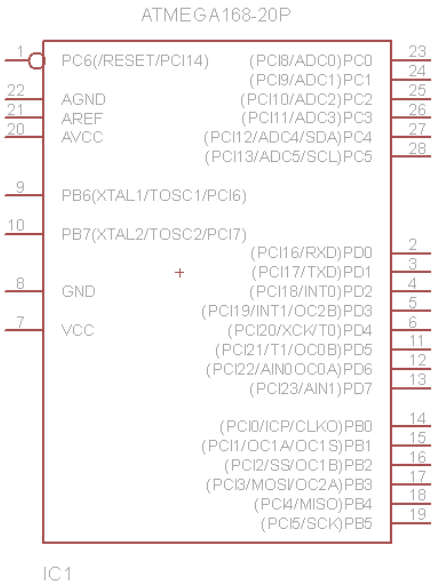


Εικόνα 14: Η πραγματική πλακέτα κεντρικής πλατφόρμας με τοποθέτηση υλικών

2.4.2 Πλακέτα αισθητήρα θερμοκρασίας

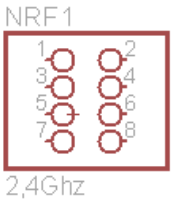
Με την ίδια μεθοδολογία Όπως και στην προηγούμενη ενότητα, Έτσι και εδώ χρησιμοποιούμε το ίδιο πρόγραμμα σχεδιασμού για τί σχεδίαση της πλατφόρμας. Αρχικά εισάγουμε τα κύρια εξαρτήματα που αποτελούν την κεντρική μονάδα. με ιδιαίτερη προσοχή στην αντιστοιχία του πραγματικού εξαρτήματος σε σχέση με αυτού του προγράμματος. παρακάτω απεικονίζονται σε μορφή symmetric εξαρτήματα.

- EAGLE Schematic του ATMega 168-20p



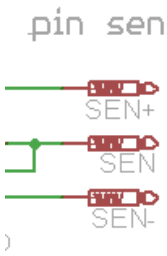
Σχήμα20: Schematic του ATMega 168-20p

- EAGLE Schematic του nRF 24L01



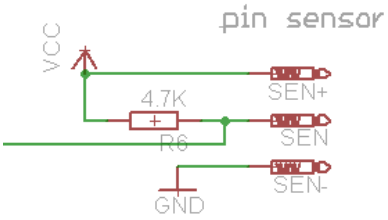
Σχήμα21: Schematic του nRF 24L01

- EAGLES chematic του αισθητήρα θερμοκρασίας DHT11



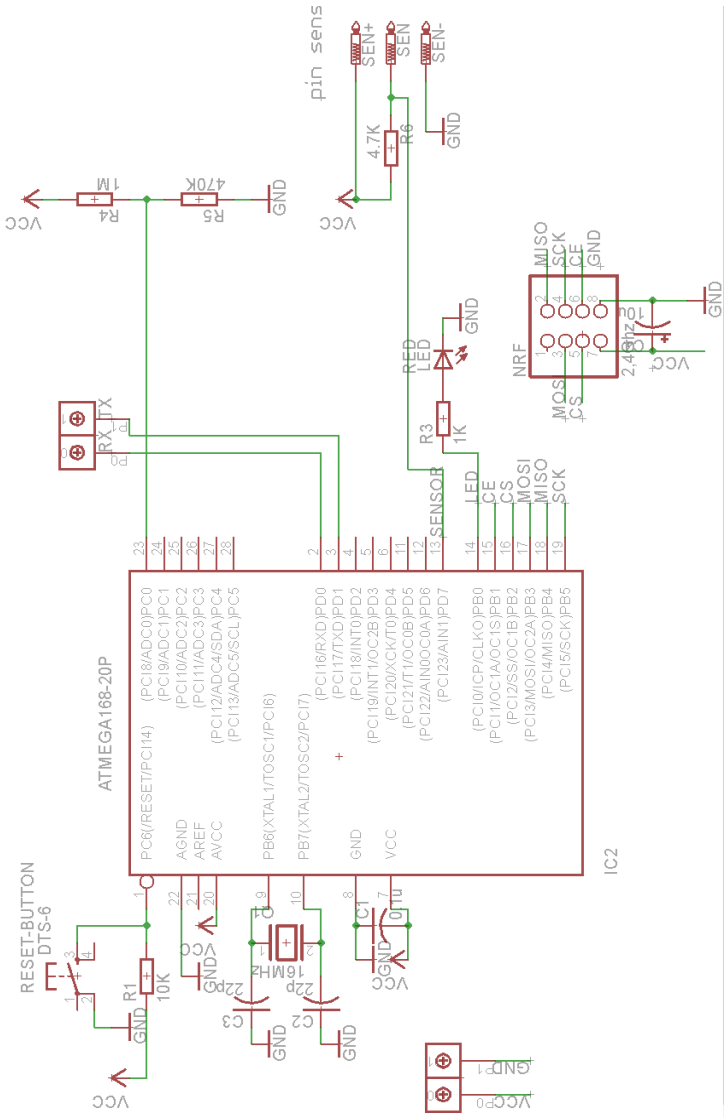
Σχήμα 22: Schematic του DHT11

- EAGLES chematic του κυκλώματος προσαρμογής του αισθητήρα DHT11



Σχήμα 23: Ηλεκτρική προσαρμογή Schematic του DHT11

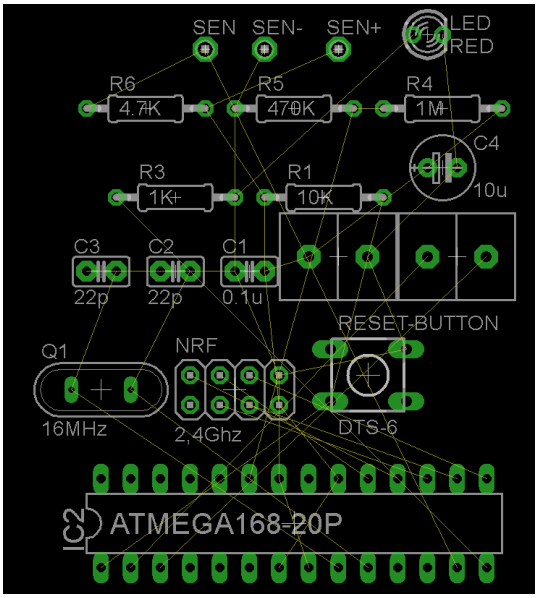
Όπως φαίνεται στο Σχήμα 24, παρακάτω έχει ολοκληρωθεί η συνδεσμολογία των εξαρτημάτων με τη μέθοδο, που έχει τη δυνατότητα το πρόγραμμα, ονομασίας στα δύο άκρα των αγώγιμων αγωγών επιτυγχάνοντας ένα πιο καθαρό σχέδιο. επίσης έχουν εισαχθεί και άλλα βοηθητικά κυκλώματα όπως, κύκλωμα χρονισμού του μικροελεγκτή με χρήση κρυστάλλου, βοηθητικό κύκλωμα σταθεροποίησης αλλά και προσθήκη PIN συνδεσιμότητας.



Σχήμα 24: Ολοκλήρωση ηλεκτρικής σύνδεσης πλακέτας εισόδου θερμοκρασίας

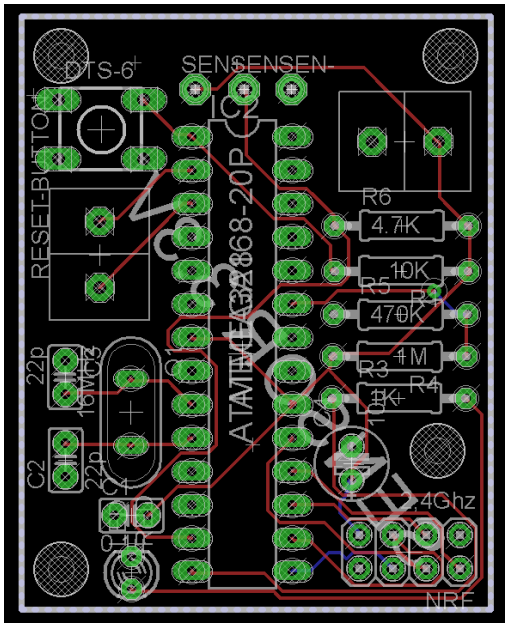
Μετά την ολοκλήρωση του σχεδιασμού του κυκλώματος σε μορφή EAGLE Schematic, θα μεταβούμε στο στάδιο του EAGLE Board. Όπως θα δούμε στο

παρακάτω Σχήμα 25, απεικονίζεται σε πραγματικό μέγεθος αλλά και σε πραγματικό χώρο η πλακέτα εισόδου θερμοκρασίας. Σε αυτό το στάδιο το πρόγραμμα έχει κάνει μία τυχαία τοποθέτηση των υλικών όπου παρατηρείται η διασταύρωση αγώγιμων δρόμων(κίτρινες διασυνδέσεις) όπου σίγουρα δεν είναι αυτό που επιθυμούμε. Σε αυτό το σημείο καλούμαστε να τοποθετήσουμε τα υλικά με τη δομή που εμείς επιθυμούμε πάνω στην πλακέτα λαμβάνοντας πάντα υπόψη μας τις αλληλεπιδράσεις μεταξύ των υλικών. Επίσης έχουμε τη δυνατότητα να γίνει τυπώσει Πλακέτας διπλής όψης. αυτό σημαίνει ότι έχουμε μικρότερο χώρο αλλά και μεγαλύτερη ευελιξία στον σχηματισμό της πλακέτας.



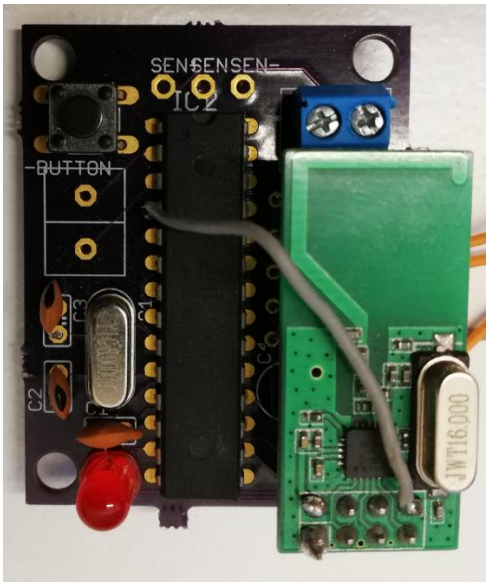
Σχήμα 25: Τυχαία τοποθέτηση εξαρτημάτων στο EAGLE Board

Λαμβάνοντας υπόψη όλες τις παραπάνω πληροφορίες έχει ολοκληρωθεί η διαδικασία όπου απαιτείται για το τελικό αποτέλεσμα των αγώγιμων διαδρομών της πλακέτας. Στο παρακάτω Σχήμα 26, φαίνεται η δομή και ο προσανατολισμός των εξαρτημάτων σε μία πλακέτα διπλής όψης. Οι μπλε αγώγιμες διαδρομές είναι η πίσω πλευρά της πλακέτας ενώ η κόκκινη αγώγιμη διαδρομή είναι η πάνω πλευρά της πλακέτας όπου τοποθετούνται και τα κυρίως υλικά.



Σχήμα 26: Ολοκλήρωση δομής πλακέτας στο EAGLE Board

Στην παρακάτω εικόνα 15 έχει γίνει η τοποθέτηση όλων των υλικών και έχει γίνει η δοκιμή και όλα τα τεστ για τη σωστή λειτουργία.



Εικόνα 15: Η πραγματική πλακέτα της πλατφόρμας εισόδου θερμοκρασίας με τοποθέτηση υλικών

2.4.3 Περιγραφή πλακέτας εξόδου (με χρήση ηλεκτρονόμου)

Με την ίδια μεθοδολογία όπως και στην προηγούμενη ενότητα, έτσι και εδώ χρησιμοποιούμε το ίδιο πρόγραμμα σχεδιασμού γιατί σχεδιάζει της πλατφόρμας.

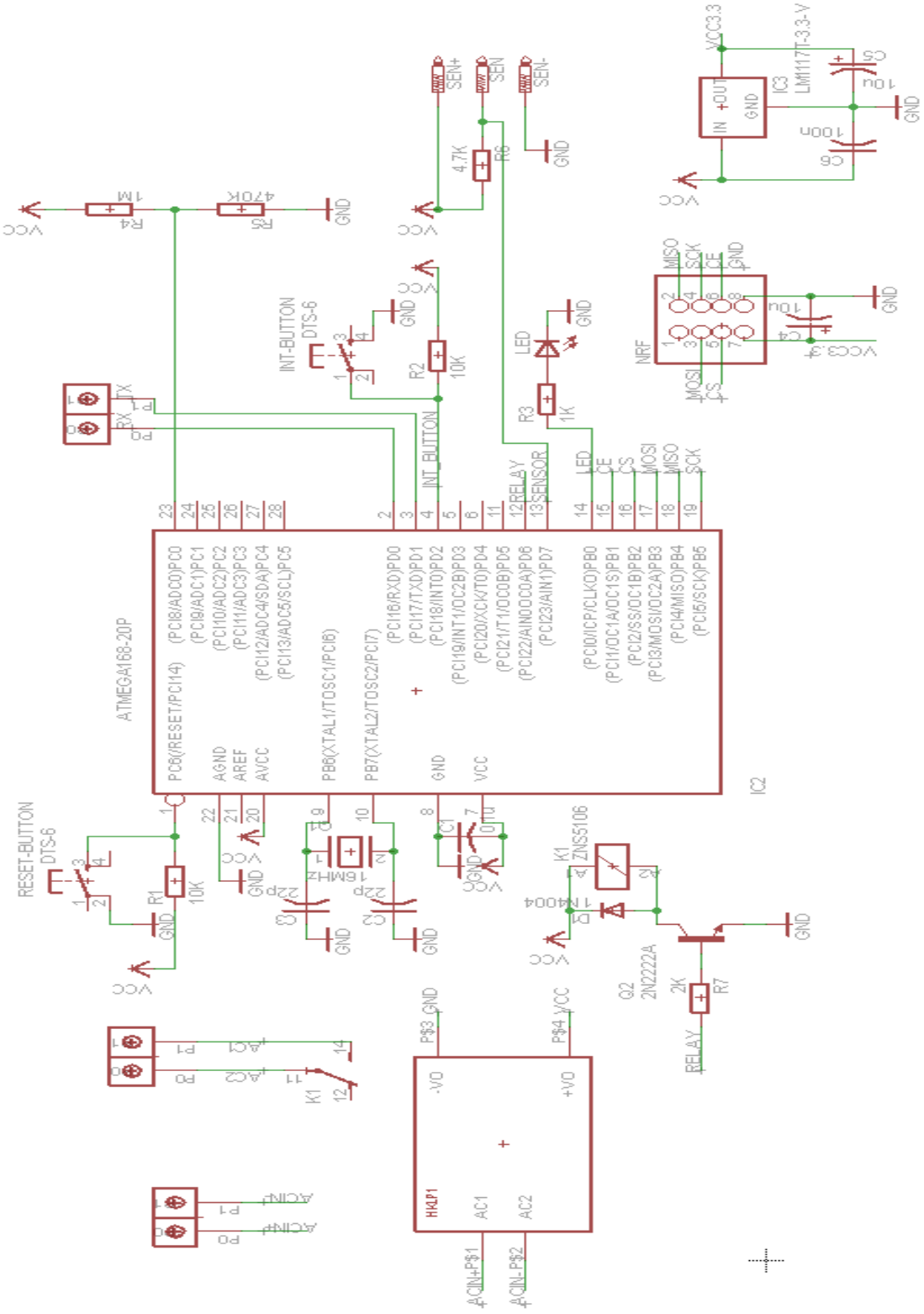
Αρχικά εισάγουμε τα κύρια εξαρτήματα που αποτελούν την κεντρική μονάδα. Με ιδιαίτερη προσοχή στην αντιστοιχία του πραγματικού εξαρτήματος σε σχέση με αυτού του προγράμματος. Παρακάτω απεικονίζονται σε μορφή symmetric εξαρτήματα.

EAGLE Schematic του ηλεκτρονόμου

Σχήμα 27: Schematic του ηλεκτρονόμου

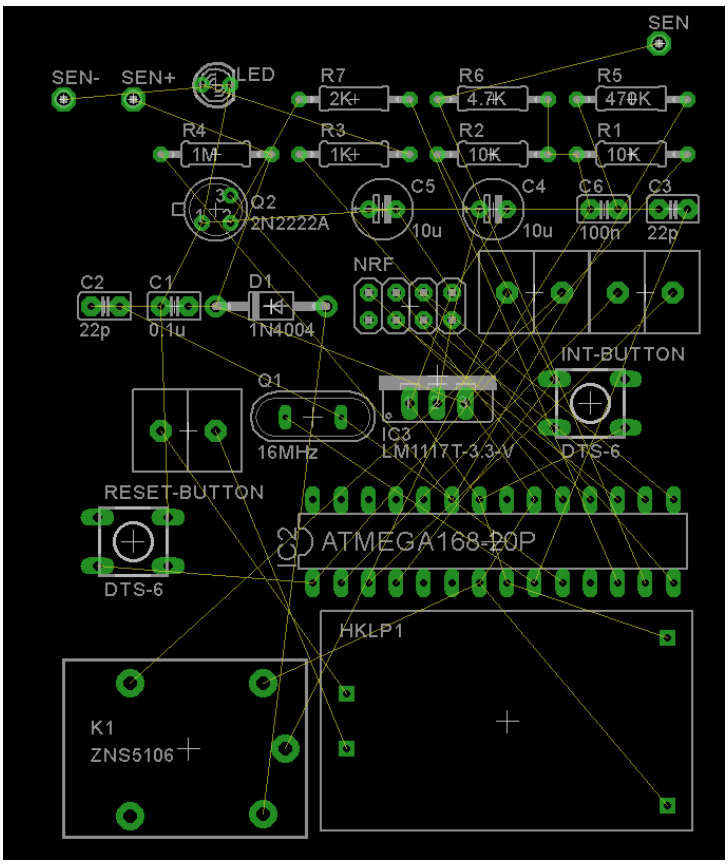
Τα ποιο Βασικά εξαρτήματα μήπως μικροεπεξεργαστή ATMega 168-20p, nRF 24L01, Wi Fi ESP8266MOD Είναι ίδια με αυτά της προηγούμενης ενότητας. Επίσης για την έξοδο χρησιμοποιείτε ο ηλεκτρονόμος SPDT 5V 10A της εταιρίας RAYEX ELECTRONIC.

Όπως φαίνεται στο Σχήμα 28 παρακάτω έχει ολοκληρωθεί η συνδεσμολογία των εξαρτημάτων με τη μέθοδο που έχει τη δυνατότητα το πρόγραμμα, ονομασίας στα δύο άκρα των αγώγιμων αγωγών επιτυγχάνοντας ένα πιο καθαρό σχέδιο. Επίσης έχουν εισαχθεί και άλλα βοηθητικά κυκλώματα όπως, κύκλωμα χρονισμού του μικροελεγκτή με χρήση κρυστάλλου, βοηθητικό κύκλωμα σταθεροποίησης αλλά και προσθήκη PIN συνδεσιμότητας.


ATMEGA168-20P
IC2
RESET-BUTTON
DTS-6
INT-BUTTON
DTS-6
R1
10K
R2
10K
R3
1K
R4
1M
R5
470K
R6
4.7K
R7
2K
LED
RELAY
ZNS5106
1N4004
Q2
2N2222A
IC3
LM1117T-3.3-V
VCC3.3
NRF
MISO
SCK
CE
GND
MOSI
CS
SEN+
SEN-
SEN
INT_BUTTON
SENSOR
RELAY
RX_TX
HKJ1
AC1
AC2
-VO
+VO
P$3_GND
P$4_VCC
ACIN+P$1
ACIN-P$2
16MHZ
100n
10u

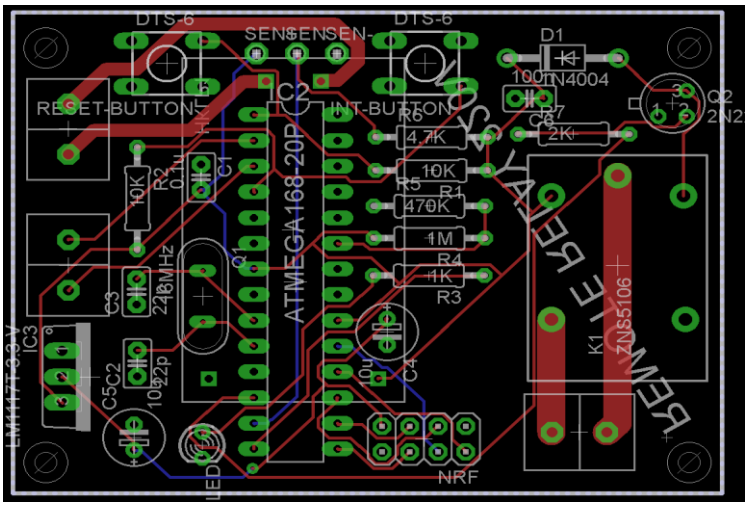
Σχήμα 28: Ολοκλήρωση ηλεκτρικής σύνδεσης πλακέτας εξόδου με τη βοήθεια ηλεκτρονόμο.

Μετά την ολοκλήρωση του σχεδιασμού του κυκλώματος σε μορφή EAGLE Schematic, θα μεταβούμε στο στάδιο του EAGLE Board. Όπως θα δούμε στο παρακάτω Σχήμα 29, απεικονίζεται σε πραγματικό μέγεθος αλλά και σε πραγματικό χώρο η πλακέτα εισόδου θερμοκρασίας. Σε αυτό το στάδιο το πρόγραμμα έχει κάνει μία τυχαία τοποθέτηση των υλικών όπου παρατηρείται η διασταύρωση αγώγιμων δρόμων (κίτρινες διασυνδέσεις) όπου σίγουρα δεν είναι αυτό που επιθυμούμε. Σε αυτό το σημείο καλούμαστε να τοποθετήσουμε τα υλικά με τη δομή που εμείς επιθυμούμε πάνω στην πλακέτα λαμβάνοντας πάντα υπόψη μας τις αλληλεπιδράσεις μεταξύ των υλικών. Επίσης έχουμε τη δυνατότητα να γίνει τυπώσει πλακέτας διπλής όψης. Αυτό σημαίνει ότι έχουμε μικρότερο χώρο αλλά και μεγαλύτερη ευελιξία στον σχηματισμό της πλακέτας.



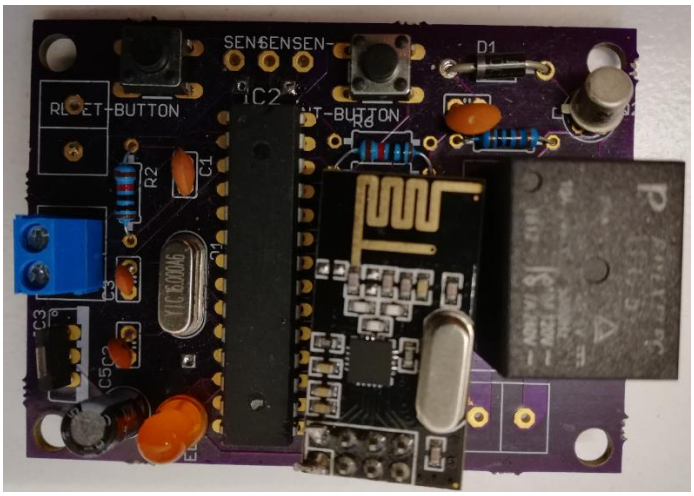
Σχήμα 29: Τυχαία τοποθέτηση εξαρτημάτων στο EAGLE Board

Λαμβάνοντας υπόψη όλες τις παραπάνω πληροφορίες έχει ολοκληρωθεί η διαδικασία όπου απαιτείται για το τελικό αποτέλεσμα των αγώγιμων διαδρομών της πλακέτας. Στο παρακάτω Σχήμα 30, φαίνεται η δομή και ο προσανατολισμός των εξαρτημάτων σε μία πλακέτα διπλής όψης. Οι μπλε αγώγιμες διαδρομές είναι η πίσω πλευρά της πλακέτας ενώ η κόκκινη αγώγιμη διαδρομή είναι η πάνω πλευρά της πλακέτας όπου τοποθετούνται και τα κυρίως υλικά.



Σχήμα 30: Ολοκλήρωση δομής πλακέτας στο EAGLE Board

Στην παρακάτω εικόνα 16 έχει γίνει η τοποθέτηση όλων των υλικών και έχει γίνει η δοκιμή και όλα τα τεστ για τη σωστή λειτουργία.



Εικόνα 16: Η πραγματική πλακέτα της πλατφόρμας εξόδου με τοποθέτηση υλικών

2.5 Κώδικας συστήματος

Σε αυτό το σημείο θα διεξαχθεί ο προγραμματισμός του συστήματος, για την κάθε μία πλατφόρμα ξεχωριστά. Η ανάπτυξη του κώδικα θα πραγματοποιηθεί σε γλώσσα C++. Μετά την ανάπτυξη του κώδικα, με τι βοήθεια της πλατφόρμας ARDUINO αλλά και του software ARDUINO 1.8.5, θα μεταβιβάσουμε των κώδικα στης πλατφόρμες.

2.5.1 Κώδικας για Κεντρική πλατφόρμα

```
#definebit_get(p,m) ((p) & (m))
#define bit_set(p,m) ((p) |= (m))
#define bit_clear(p,m) ((p) &= ~(m))
#define bit_flip(p,m) ((p) ^= (m))
#define bit_write(c,p,m) (c ? bit_set(p,m) : bit_clear(p,m))
#define BIT(x) (0x01 << (x))
#define LONGBIT(x) ((unsigned long)0x00000001 << (x))


//#include <SPI.h>
//#include <nRF24L01.h>
#include <RF24.h>
//#include <RF24_config.h>
#include <ArduinoJson.h>
#include <MemoryFree.h>
#include <avr/pgmspace.h>
#include "printf.h"
#include <DS3231.h>

#define mySerial Serial2
#define bufSize 500
#define bufDateSize 64
```

```
#define bufProgmemSize 64

#define RST 15
#define BUTTON4   BIT(5)     //PG5 -- 4
#define BUTTON5   BIT(3)     //PE3 -- 5
#define BUTTON6   BIT(3)     //PH3 -- 6
#define BUTTON7   BIT(4)     //PH4 -- 7

#define LED_BLUE   BIT(5)     //PH5    --digitalpin8
#define LED_RED    BIT(4)     //PE4  --digitalpin2
#define LED_YELLOW BIT(5)   //PE5     --digitalpin3
#define INTERVAL_ESP 60000
#define TIMEOUT_NRF 2000
#define DEBUG true


//--------------------------------PROGMEM-----------------------------------------
enum {RESET,MODE,CWJAP2,CWJAPP,CWSAP2,CWSAP,CIPMUX,
CIPSERVER,CIPCLOSE1,CIPSTART,CIPSEND,
POST1,POST2,POST3,POST4,POST5,CIPCLOSE2,SERVER,PATH
};
const char reset[]     PROGMEM = "AT+RST";
const char mode[]      PROGMEM = "AT+CWMODE_DEF=3";
const char cwjap2[]    PROGMEM = "AT+CWJAP_DEF?";
const char cwjapp[]    PROGMEM = "AT+CWJAP_DEF=\"";
const char cwsap2[]    PROGMEM = "AT+CWSAP_DEF?";
const char cwsap[]     PROGMEM = "AT+CWSAP_DEF=\"espkotso\",\"\",1,0";
const char cipmux[]    PROGMEM = "AT+CIPMUX=1";
const char cipserver[] PROGMEM = "AT+CIPSERVER=1,80";
const char cipclose1[] PROGMEM = "AT+CIPCLOSE=5";
const char cipstart[]  PROGMEM = "AT+CIPSTART=\"TCP\",\"wp-admin.space\",80";
const char cipsend[]   PROGMEM = "AT+CIPSEND=";
const char post1[]     PROGMEM = "POST /";
const char post2[]     PROGMEM = " HTTP/1.1\r\nHost: ";
```

```
const char post3[]     PROGMEM = "\r\nContent-Type: application/";
const char post4[]     PROGMEM = "json\r\nContent-Length: ";
const char post5[]     PROGMEM = "\r\n\r\n";
const char cipclose2[] PROGMEM = "AT+CIPCLOSE";
const char server[]    PROGMEM = "wp-admin.space";
const char path[]      PROGMEM = "growUnit/postGrowData";
const char* const StringArray[] PROGMEM = {reset,mode,cwjap2,cwjapp,cwsap2,cwsap,
cipmux,cipserver,cipclose1,cipstart,cipsend,post1,post2,post3,post4,post5,cipclose2,server,path};

//-----------------------------------ESP-----------------------------------
char buf[bufSize];
char bufDate[bufDateSize];
char bufProgmem[bufProgmemSize];

String CWJAP;
String JSON;
char ssid[25]={0};
char pass[20]={0};

byte error[11] = {0,0,0,0,0,0,0,0,0,0,0};

float room1[7]={0,0,0,0,0,0,0};
float sensor1[5]={0,0,0,0,0};
float sensor2[5]={0,0,0,0,0};
float sensor3[5]={0,0,0,0,0};
float sensor4[5]={1,0,0,0,0};
float sensor5[5]={0,0,0,0,0};

float lamp[4]={0,0,0,0};
float hum[4]={0,0,0,0};
float fan1[4]={0,0,0,0};
float fan2[4]={0,0,0,0};
float water[5]={0,0,0,0,0};
float fert[6]={0,0,0,0,0,0};
```

```
unsigned long sendloop;

unsigned long sendtimer;

boolean syncFlag=true;

boolean firstFlag=true;

//--------------------------------------NRF-----------------------------------

RF24 radio(49,53);

const uint64_t SensorAdd[]={0xA8A8E1F0C6LL, 0xA8A8E1F0A5LL, 0xA8A8E1F0B7LL, 0xA8A8E1F0D6LL, 0xA8A8E1F0E7LL};  // 0:Node1 26s, 1:....., 2:Node3 58s

const uint64_t RelaysAdd[]={0xA8A8D2F0C6LL, 0xA8A8D2F0A5LL, 0xA8A8D2F0B7LL, 0xA8A8D2F0D6LL, 0xA8A8D2F0E7LL};  // 0:livingRoom(node 1), 1:office(Node 2),...

uint64_t tempAdd;

boolean flagbutton = false;

//--------------------------------------DS3231-----------------------------------

DS3231  rtc(SDA, SCL);

Time  t;

//------if(DEBUG)----------

int sensorfreq[3]={26,0,58};

unsigned long nodetimer[4];

unsigned long start[4];    //start[0] for button,[1] for node1, [2] for node2...

unsigned long fin[4];

char fail1=0;              //wifi        2

char fail2=0;              //cipstart    7

char fail3=0;              //cipsend     8

char fail4=0;              //sendRequest  9

char fail5=0;              //cipclose    10

char fail6=0;              //sensor1

char fail7=0;              //sensor3

int startup=0;

String failures;

//-------------------------------------------------------------------------------



void setup()

{
```

```
  DDRH |= LED_BLUE;
  DDRE |= LED_RED;
  DDRE |= LED_YELLOW;
  DDRG &= ~BUTTON4;
  DDRE &= ~BUTTON5;
  DDRH &= ~BUTTON6;
  DDRH &= ~BUTTON7;

  Serial.begin(9600);
  mySerial.begin(9600);
  rtc.begin();
  delay(20);
  radio_init();
  delay(50);
  resetESP();
}

//--------------------------------------------------------------------
void loop(){


  if (radio.available()){
    char packet[4];
    if(radio.read(&packet, sizeof(packet))){
  Serial.print(F("SENSOR#: "));
  Serial.print(packet[0]);

  Serial.print(F("Bat(V):  "));
  Serial.println(packet[1]);
  Serial.print(F("Sens:   "));
  Serial.println(packet[2]);
  Serial.print(F("Err(s):  "));
  Serial.println(packet[3]);
  Serial.println(F("\n"));}
```

```
  }



//radio_listening();

if(millis()-sendloop >INTERVAL_ESP && firstFlag==true){
  Serial.print(F("RAM-->radiocheck=")); Serial.println(freeMemory());
  radio_check();
  Serial.print(F("RAMradiocheck--sendTCP=")); Serial.println(freeMemory());
  sendTcpESP();
  }



//--------------BUTTONS-----------------------------------

if(!bit_get(PING,BUTTON4) && firstFlag==true){
  radio_check();
  sendTcpESP();
 }


if(millis()-start[0] >500)flagbutton=false;

if(!bit_get(PINE,BUTTON5) && flagbutton==false){
  start[0]=millis();
  resetESP();
  flagbutton=true;}

if(!bit_get(PINH,BUTTON6) && flagbutton==false){
  start[0]=millis();
  tempAdd=0xA8A8D2F0C6LL;
  radio_send();
```

```
  debug();
  flagbutton=true;}

if(!bit_get(PINH,BUTTON7) && flagbutton==false){
  firstConnectionESP();
  flagbutton=true;}

}//-----------------------------------------------------------------------

void sendTcpESP(){
  bit_set(PORTH,LED_BLUE);
  createJSON();
  String POSTtemp;
  String CIPSENDtemp;
  POSTtemp = getStringMEM(POST1);
  POSTtemp+= getStringMEM(PATH);
  POSTtemp+= getStringMEM(POST2);
  POSTtemp+= getStringMEM(SERVER);
  POSTtemp+= getStringMEM(POST3);
  POSTtemp+= getStringMEM(POST4);
  POSTtemp+= JSON.length();
  POSTtemp+= getStringMEM(POST5);
  POSTtemp+= JSON;
  POSTtemp+= getStringMEM(POST5);
  CIPSENDtemp= getStringMEM(CIPSEND);
  CIPSENDtemp+= POSTtemp.length();
  CIPSENDtemp+= "\r\n";
 Serial.print(F("RAMposttemp--sendATtcp=")); Serial.println(freeMemory());
  sendloop=millis();

if(!SendAT(getStringMEM(CIPSTART),"OK",5000)) {error[7]=1;}
if(!SendAT(CIPSENDtemp,"OK",5000))                                    {error[8]=1;}
//if(!SendREQUEST(CIPSENDtemp,"OK",">",3000))    {error[8]=1;}
if(syncFlag==false){
  if(!SendREQUEST(POSTtemp,"SEND OK","}}",4000)){error[9]=1;}}
```

```
if(syncFlag==true){
  if(!SyncTimeESP(POSTtemp,"GMT","}}",4000))  {error[9]=1;} SyncClock();}
if(!SendAT(getStringMEM(CIPCLOSE2),"OK",3000)){error[10]=1;}
decodeJSON();
Serial.println(JSON);
Serial.print("Timer:");Serial.println(millis()-sendloop);
bit_clear(PORTH,LED_BLUE);
debug();
}

//if(!SendATdual(getStringMEM(CWJAP2),ssid,"OK",2000)){error[2]=1;}
//if(!SendAT(getStringMEM(CWSAP),"OK",3000)) error[3]=1;

void firstConnectionESP(){


firstFlag=false;
if(!SendATdual(getStringMEM(CWSAP2),"espkotso","OK",3000)) error[3]=1;
if(!SendAT(getStringMEM(CIPMUX),"OK",3000)) error[4]=1;
if(!SendAT(getStringMEM(CIPSERVER),"OK",3000))error[5]=1;

WAITforECHO("HTTP/", 30000);
if(!SendAT(getStringMEM(CIPCLOSE1),"OK",3000)){error[6]=1; debug; return;}


//if(find HTTP,ssid,pss at WAITforECHO){store ssid; CWJAP=...;}
storeSSID();
if(!SendAT(CWJAP,"OK",30000)) {error[2]=1; debug; return;}
 else{firstFlag=true;}

debug();
resetESP();
}
```

```
void resetESP(){
 if(!SendATdual(getStringMEM(RESET), "Ready","WIFI GOT IP",10000)) error[0]=1;
 //SendAT("ATE0","OK",2000);
 debug();
 }

void hardresetESP(){
  digitalWrite(RST,LOW);
  digitalWrite(LED_RED,HIGH);
  delay(100);
  digitalWrite(RST,HIGH);
  digitalWrite(LED_RED,LOW);
  debug();
}
//-------------------------------------------------------------------------------

void SyncClock(){
   uint8_t hours = ((bufDate[23]-48)*10) + (bufDate[24]-48);
   uint8_t mins =  ((bufDate[26]-48)*10) + (bufDate[27]-48);
   uint8_t secs =  ((bufDate[29]-48)*10) + (bufDate[30]-48);
   rtc.setTime(hours, mins,secs);
   //rtc.setDOW(SUNDAY);
   //rtc.setDate(7, 1, 2018);
   syncFlag=false; }


void storeSSID(){
char * pch; int j=0; int i=0; int k=0; boolean flag1=true; boolean flag2=false;
pch=strchr(buf,'/');
while(i<sizeof(buf)){
i++;
if(pch[i]=='/'){            // if finds '/'
  if(pch[++i]=='~'){
```

```
  flag1=false;
  flag2=true;
  i++;
  }else i--;}
if(pch[i]==' '){              // if finds ' '
 if(pch[++i]=='H'){
   i+=4;
   if(pch[i]=='/'){
   flag2=false;
 }else i-=5;
}else i--;}
if(flag2)
 pass[k++]=pch[i];
if(flag1)
 ssid[j++]=pch[i];}

CWJAP = getStringMEM(CWJAPP);
CWJAP += ssid;
CWJAP += "\",\"";
CWJAP += pass;
CWJAP += "\"";

}

void createJSON(){
  StaticJsonBuffer<500> jBuffer;      //DynamicJsonBuffer jBuffer;
  JsonObject& UNIT=jBuffer.createObject();
  JsonArray& AAABBBAAADDD=UNIT.createNestedArray("AAABBBAAADDD");
  JsonObject& R1=AAABBBAAADDD.createNestedObject().createNestedObject("R1");
  JsonObject& S1=AAABBBAAADDD.createNestedObject().createNestedObject("S1");
  JsonObject& S2=AAABBBAAADDD.createNestedObject().createNestedObject("S2");
  JsonObject& S3=AAABBBAAADDD.createNestedObject().createNestedObject("S3");
  JsonObject& S4=AAABBBAAADDD.createNestedObject().createNestedObject("S4");
  JsonObject& S5=AAABBBAAADDD.createNestedObject().createNestedObject("S5");
```

```
R1["A"]= room1[0]; R1["B"]= room1[1]; R1["C"]= room1[2]; R1["D"]= room1[3];

R1["E"]= room1[4]; R1["F"]= room1[5]; R1["G"]= room1[6];

S1["A"]= sensor1[0]; S1["B"]= sensor1[1]; S1["C"]= sensor1[2]; S1["D"]= sensor1[3]; S1["E"]= sensor1[4];

S2["A"]= sensor2[0]; S2["B"]= sensor2[1]; S2["C"]= sensor2[2]; S2["D"]= sensor2[3]; S2["E"]= sensor2[4];

S3["A"]= sensor3[0]; S3["B"]= sensor3[1]; S3["C"]= sensor3[2]; S3["D"]= sensor3[3]; S3["E"]= sensor3[4];

S4["A"]= sensor4[0]; S4["B"]= sensor4[1]; S4["C"]= sensor4[2]; S4["D"]= sensor4[3]; S4["E"]= sensor4[4];

S5["A"]= sensor5[0]; S5["B"]= sensor5[1]; S5["C"]= sensor5[2]; S5["D"]= sensor5[3]; S5["E"]= sensor5[4];

  JSON="";

  UNIT.printTo(JSON);       //UNIT.prettyPrintTo(Serial);

  }



void decodeJSON(){

StaticJsonBuffer<500> jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(buf);

JsonObject& L = root["L"];

JsonObject& H = root["H"];

JsonObject& F1 = root["F1"];

JsonObject& F2 = root["F2"];

JsonObject& W = root["W"];

JsonArray& W_FE = W["FE"];


lamp[0] = L["SON"];

lamp[1] = L["SOF"];

lamp[2] = L["A"];

lamp[3] = L["RNG"];


hum[0] = H["SON"];

hum[1] = H["SOF"];

hum[2] = H["TG"];

hum[3] = H["A"];
```

```
fan1[0] = F1["SON"];
fan1[1] = F1["SOF"];
fan1[2] = F1["TG"];
fan1[3] = F1["A"];

fan2[0] = F2["SON"];
fan2[1] = F2["SOF"];
fan2[2] = F2["TG"];
fan2[3] = F2["A"];

water[0] = W["SON"];
water[1] = W["Q"];
water[2] = W["TG"];
water[3] = W["BA"];
water[4] = W["A"];
fert[0] = W_FE[0];
fert[1] = W_FE[1];
fert[2] = W_FE[2];
fert[3] = W_FE[3];
fert[4] = W_FE[4];
fert[5] = W_FE[5];
}

//-----------------------------------------------NRF-----------------------------

void radio_init(){
  radio.begin();
  printf_begin();
 // radio.setChannel(108);
  radio.openReadingPipe(0,SensorAdd[0]);
  radio.openReadingPipe(1,SensorAdd[1]);
  radio.openReadingPipe(2,SensorAdd[2]);
  radio.openReadingPipe(3,SensorAdd[3]);
```

```
 radio.openReadingPipe(4,SensorAdd[4]);
 radio.openReadingPipe(5,SensorAdd[5]);
 radio.startListening();
}

void radio_listening(){
 if (radio.available()){   digitalWrite(LED_YELLOW,HIGH);
   float packet[5]={0,0,0,0,0}; byte id;
   if(radio.read(&packet, sizeof(packet))){
   byte i=0; id=int(packet[0]);
   switch (id){
     case 1:
     for(i=0;i<5;i++)sensor1[i]=packet[i];
     break;
     case 2:
     for(i=0;i<5;i++)sensor2[i]=packet[i];
     break;
     case 3:
     for(i=0;i<5;i++)sensor3[i]=packet[i];
     break;
   }
   start[id]=millis()-fin[id];
   fin[id]=millis();
   nodetimer[id]=float(start[id]/1000.00);
   digitalWrite(LED_YELLOW,LOW);

if(DEBUG){
 Serial.print(F("SENSOR#: "));
 Serial.print(packet[0]);
 Serial.print(F("   Prev:  "));
 Serial.println(float(start[id]/1000.00));
 Serial.print(F("Bat(V):  "));
 Serial.println(packet[1]);
 Serial.print(F("Sens:   "));
```

```
  Serial.println(packet[2]);
  Serial.print(F("Err(s):  "));
  Serial.println(packet[3]);
  Serial.println(F("\n"));}
    }else {if(DEBUG){Serial.println(F("Didn't catch it!"));}}
  }
}


void radio_send(){
  boolean flagSendmsg = false; byte errorNRF=0;
  //int msg[] = {111};
  char msg[] = "111";
  radio.stopListening();
  radio.openWritingPipe(tempAdd);
  flagSendmsg=false;
  start[0]=millis();
  while(flagSendmsg==false){
      if(radio.write(&msg, sizeof(msg))){
        if(DEBUG)Serial.println(F("SUCCESS"));
      flagSendmsg=true;
      }else {if(DEBUG)Serial.println(F("Keep trying..."));}
      if(millis()-start[0]>=TIMEOUT_NRF){
        flagSendmsg=true; errorNRF=1;
        if(DEBUG)Serial.println(F("ERROR!"));}
  }
  radio.startListening();
}


void radio_check(){
  startup++;
for(int i=0;i<3;i++){
  if(nodetimer[i+1] > (sensorfreq[i]-2) && nodetimer[i+1] < (sensorfreq[i]+2)){
```

```
   Serial.print(F("Sensor "));
   Serial.print(i+1);
   Serial.println(F(" is OK"));
  }else {
     Serial.print(F("Something is wrong with sensor "));
     Serial.println(i+1);
     if(startup>3){
      if(i+1 == 1){fail6++;} //   failures+=rtc.getTimeStr(); failures+="    ";}
      if(i+1 == 3){fail7++;} //  failures+=rtc.getTimeStr(); failures+="    ";}
     }
}}




}
//--------------------------------VARIOUS------------------------------

boolean SendAT(String cmd,char const *match, long timeout){
int i=0; byte c; boolean flag=true,flagOK=false; char buf[400];
mySerial.println(cmd); long start = millis();
while(millis()- start < timeout){
  while(mySerial.available()>0){
    c=mySerial.read();
     switch (c){
      case '\n':
      Serial.write(c);
      i=0;
      if (strcasestr(buf,match)){
         flagOK=true;}
      break;
      default:
      Serial.write(c);
      buf[i] = c;  i++;  buf[i]= 0;
      break;}
```

```
   if(flagOK==true)
   return flagOK;
  }} return flagOK;}


boolean SendATdual(String cmd,char const *match,char const *match2, long timeout){  //uses local buf[]
int i=0; byte c; boolean flag=true,flagOK=false,flagOK2=false; char buf[400];
mySerial.println(cmd); long start = millis();
while(millis()- start < timeout){
  while(mySerial.available()>0){
   c=mySerial.read();
     switch (c){
      case '\n':
      Serial.write(c);
      i=0;
      if(flag==true){
      if(strcasestr(buf,match)){flagOK=true;flag=false;}}      //check for "COSMOTE"
      if(strcasestr(buf,match2)) flagOK2=true;                  // check for "OK"
      break;
      default:
      Serial.write(c);
      buf[i] = c; i++; buf[i]= 0;
      break;}
    if(flagOK2==true){
    return flagOK;}
  }}return flagOK;}


boolean SendREQUEST(String cmd,char const *match,char const *match2, long timeout){    //uses global buf[]
int i=0; byte c; boolean flag=true,flagOK=false,flagOK2=false;
mySerial.println(cmd); long start = millis();
while(millis()- start < timeout){
  while(mySerial.available()>0){
   c=mySerial.read();
```

```
    switch (c){
     case '\n':
      Serial.write(c);
      i=0;
     if(flag==true){if(strcasestr(buf,match))  {flagOK=true; flag=false;} }
     if(flag==false){if(strcasestr(buf,match2)){flagOK2=true;}}
     break;
     default:
      Serial.write(c);
      buf[i] = c;  i++; buf[i]= 0;
     break;
    }if(flagOK2)return flagOK;
    }}return flagOK;}



boolean SyncTimeESP(String cmd,char const *match,char const *match2, long timeout){  //uses global buf[]
int i=0; byte c; boolean flag=true,flagOK=false,flagOK2=false;
mySerial.println(cmd); long start = millis();
while(millis()- start < timeout){
  while(mySerial.available()>0){
   c=mySerial.read();
     switch (c){
      case '\n':
       Serial.write(c);
      if(flag==true){
        if(strcasestr(buf,match)){
          flagOK=true; flag=false;
          for(int j=0;j<i;j++)bufDate[j]=buf[j];}}
      if(flag==false){if(strcasestr(buf,match2)){flagOK2=true;}}
      i=0;
      break;
      default:
       Serial.write(c);
       buf[i] = c;  i++; buf[i]= 0;
```

```
      break;
    }if(flagOK2)return flagOK;
    }} return flagOK;}


boolean WAITforECHO(char const *match, long timeout){       //uses global buf[]
int i=0; boolean flag=true; boolean flagOK=false; byte c; long start = millis();
while(millis()- start < timeout){
  while(mySerial.available()>0){
   c=mySerial.read();
     switch (c){
      case '\n':
      Serial.write(c);
      i=0;
      flag=true;
      if (strcasestr(buf,match) && flag==true){
          flagOK=true;
          flag=false;}
      break;
      default:
      Serial.write(c);
        if(flag==true){
        buf[i] = c; i++; buf[i]= 0;}
      break;}}} return flagOK;}


char* getStringMEM(int i){
  strcpy_P(bufProgmem, (char*)pgm_read_word(&(StringArray[i])));
  return bufProgmem;}

void printerrors(){
  for(int i=0;i<11;i++){
  Serial.print(error[i]);
  Serial.print("    "); }
```

```
  if(error[2]==1) {fail1++;  failures+=rtc.getTimeStr(); failures+="    ";}
  if(error[7]==1) {fail2++;  failures+=rtc.getTimeStr(); failures+="    ";}
  if(error[8]==1) {fail3++;  failures+=rtc.getTimeStr(); failures+="    ";}
  if(error[9]==1) {fail4++;  failures+=rtc.getTimeStr(); failures+="    ";}
  if(error[10]==1){fail5++;   failures+=rtc.getTimeStr(); failures+="    ";}

room1[0]=fail1;room1[1]=fail2;room1[2]=fail3;room1[3]=fail4;room1[4]=fail5;room1[5]=fail6;room
1[6]=fail7;
  Serial.print("\nFailures:  ");
  Serial.println(failures);}


void debug(){
    t = rtc.getTime();
    Serial.println(rtc.getTimeStr());
    Serial.print(F("RAM=")); Serial.println(freeMemory());
    printerrors();
    clearerrors();}


void clearerrors(){
  for(int i=0;i<11;i++){error[i]=0;}}


  void clearbuf(){
  while(mySerial.available()>0){
  digitalWrite(LED_RED,HIGH);
  byte c = mySerial.read();
  Serial.write(c);
  }}
//if(!SendAT(getStringMEM(MODE),"OK",2000))      error[1]=1;
//t=rtc.getTime();
//if(t.min%10 >= 0 &&  t.min%10 < 6) digitalWrite(LED_RED,HIGH);
//else digitalWrite(LED_RED,LOW);
//Serial.print(radio.getCRCLength());
//Serial.print(radio.getDataRate());
//radio.printDetails();
```

2.5.2 Κώδικας για Πλατφόρμα εισόδου θερμοκρασίας

```
void sending(){
  radio.powerUp(); delay(2);
  flag=false; start=millis();
  while(flag==false){
      if(radio.write(&packet, sizeof(packet))){
      if(DEBUG){Serial.println("SUCCESS"); delay(15);}
      flag=true; error=0; packet[3] = error;
      }else{ if(DEBUG){Serial.println("Keep trying...");delay(15);}}

      if(millis()-start>= TIMEOUT){
        flag=true; error++;
        packet[3] = error;
        if(DEBUG){led();  Serial.println(error);}
      }
    }
  if(error>5){shutdownflag=true; led();}
  radio.powerDown(); delay(2);
  }
//-------------------------------------------------------------
//-------------------------------------------------------------
void digitalInterrupt(){

}
//-------------------------------------------------------------
void led(){
  digitalWrite(LED,HIGH);
  delay(200);
  digitalWrite(LED,LOW);
  delay(200);
  digitalWrite(LED,HIGH);
  delay(200);
```

```
  digitalWrite(LED,LOW);
}
//-------------------------------------------------------
void sleep(){
  for(int i=0; i<SLEEP;  i++){
  MCUCR |= (3 << 5);
  MCUCR = (MCUCR & ~(1 << 5)) | (1 << 6);
  __asm__  __volatile__("sleep");
}
}

ISR(WDT_vect){
}
```

2.5.3 Κώδικας για Πλατφόρμα Εξόδου ηλεκτρονομος

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>

#define DEBUG true
#define RELAY 8
#define LED_PIN 6

#define NODE 1      // 1:living room, 2: office,
int msg[1];
char msg2[4];
RF24 radio(9,10);
//const uint64_t rAddress[]={0x7878787878LL, 0xE8E8F0F0E1LL, 0xE8E8F0F0E2LL, 0xE8E8F0F0E3LL, 0xE8E8F0F0E4LL, 0xE8E8F0F0E5LL };
//const uint64_t TXAddress[]={0xA1A2A3A4A1, 0xA1A2A3A4A2LL, 0xA1A2A3A4A3LL, 0xA1A2A3A4A4LL,};
const uint64_t RelaysAdd[]={0xA8A8D2F0C6LL, 0xA8A8D2F0A5LL, 0xA8A8D2F0B7LL, 0xA8A8D2F0D6LL, 0xA8A8D2F0E7LL};
```

```
boolean STATE=false;

int flag=0;
int INT=0;
unsigned long start;
unsigned long fin;

void setup() {
 pinMode(LED_PIN, OUTPUT);
 Serial.begin(9600);
 radio.begin();
 radio.setChannel(108);
 radio.openReadingPipe(1,RelaysAdd[NODE-1]);
 radio.startListening();
 msg[0] = 111;
 attachInterrupt(0, digitalInterrupt, FALLING);
 delay(50);
}

void loop() {
  listening();

  if(INT==1){
    Serial.println("hi");
    STATE=!STATE;
    start=0;
    digitalWrite(LED_PIN,STATE);
    digitalWrite(RELAY,STATE);
    delay(400);
    INT=0;
  }
```

```
}
//-------------------------------------------------------------
void listening(){

  if(radio.available()){
     radio.read(msg,1);     //TESTTTTT    (msg,1)
     //Serial.println(msg2);
     if (msg[0] == 111){              //TESTTTTTTTTTTTTTTTTTTTTTTTTTTT!!!!!!!!!!!!!!!
     //if(msg2[0] == 'O'){
      STATE=!STATE;
      start=0;
      digitalWrite(LED_PIN,STATE);
      digitalWrite(RELAY,STATE);
          if(DEBUG){Serial.println("\nCHANGE STATE - SUCCESS");
          Serial.println(STATE);}
      delay(400);
      }else{
          if(DEBUG){Serial.println("FAILED...");
          start++;
          Serial.println(start);
          delay(1);} }
}
}

//-------------------------------------------------------------
void digitalInterrupt(){
  INT=1;
}
ISR(WDT_vect){
}
```

Κεφάλαιο 3

3.1 Συμπεράσματα - μελλοντικές επεκτάσεις

Για την υλοποίηση του συστήματος χρησιμοποιήθηκαν κάποια υλικά τα οποία επιλέχθηκαν με βάση τα χαρακτηριστικά τους. Για τη σωστή επιλογή τους μελετήθηκε ηλεκτρική του σύνδεσμο τι τα αλλά και σε κάποιες περιπτώσεις η συμβατότητα μεταξύ τους. Γενικότερα θα μπορούσε να υλοποιηθεί το σύστημα και με άλλα παρόμοια υλικά. Για παράδειγμα ο αισθητήρας θερμοκρασίας θα μπορούσε να ήταν ένα θερμίστορ, στη θέση του Αισθητήρα θερμοκρασίας DHT11 που επιλέχθηκε για το σύστημά μας. Ακόμα στη θέση του nRF 24L01 θα μπορούσα να ήταν μία μονάδα αμφίδρομης επικοινωνίας RF 2,4Ghz.

Το σύστημα λόγω ότι στεγάζεται σε μία πλατφόρμα με πολλές δυνατότητες, Μπορεί μελλοντικά να επεκταθεί και να μπορεί να διαχειριστεί παραπάνω λειτουργίες και όχι μόνο θερμικές ζώνες. Για παράδειγμα θα μπορούσε να διαχειριστεί και τον φωτισμό ενός συνόλου. Αυτό γίνεται εφικτό αλλάζοντας κάποια κομμάτια στον κώδικα προγραμματισμού έτσι ώστε σαν είσοδος του συστήματος να μην είναι ο θερμοστάτης, αλλά ένα πλήκτρο στην εφαρμογή του κινητού τηλεφώνου ή του tablet ή ενός ηλεκτρονικού υπολογιστεί του χρήστη.

Βιβλιογραφία

[1]. http://meleththrio.teicm.gr/xmlui/bitstream/handle/123456789/26/Kef2_Sensors_Petal_Kalom.pdf;jsessionid=1F833CA751B19DF8EA2296168D9524B9?sequence=2

[2]. http://www.hep.upatras.gr/class/download/ais_mik_sis_sil_ded/sensornotes.pdf

[3]. http://www.capgo.com/Resources/Temperature/RTDs/RTD.html

[4]. http://slideplayer.gr/slide/11141618/#

[5]. http://nefeli.lib.teicrete.gr/browse/stef/hle/2007/MarakisGiannis,AlevyzakisAristotelis/attached-document/2007Marakis.pdf

[6]. https://15epalthessalonikis.files.wordpress.com/2016/03/what-is-a-microcontroller.pdf

[7]. http://apothetirio.teiep.gr/xmlui/bitstream/handle/123456789/4753/%20%20%20%20%20%20%20%20%20%20%20%20%20%20-%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20.pdf?sequence=1

[8]. https://www.tme.eu/gr/Document/3f52390268b1fd1b557e00064ba55f74/ATmega640-128x-256x-DTE.pdf

[9]. https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf