ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Κατανεμημένο Ασυρματικό Σύστημα Ελέγχου Θέρμανσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Π. Πεϊτσίνης

Επιβλέπων: Συκάς Ευστάθιος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Κατανεμημένο Ασυρματικό Σύστημα Ελέγχου Θέρμανσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Π. Πεϊτσίνης

Επιβλέπων: Συκάς Ευστάθιος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Ιούνιο του 2013.

.....	.....	.....
Ε. Συκάς	Μ. Θεολόγου	Γ. Στασινόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2013

.....

Νικόλαος Π. Πεϊτσίνης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της διπλωματικής εργασίας ήταν η κατασκευή ενός αυτοματοποιημένου συστήματος ελέγχου θερμοκρασίας των χώρων ενός σπιτιού με βάση τις αρχές της αυτονομίας θέρμανσης. Η κατασκευή περιλαμβάνει μια ομάδα μικροελεγκτών (κόμβων) οι οποίοι επικοινωνούν ασύρματα ανταλλάσοντας δεδομένα θερμοκρασιών και ελέγχοντας ως παράδειγμα δυο χώρους.

Συγκεκριμένα, οι τέσσερεις πλακέτες που κατασκευάστηκαν περιλαμβάνουν δύο (2) πανομοιότυπες πλακέτες εγκατεστημένες σε θερμαντικά σώματα, οι οποίες περιέχουν ηλεκτρονικά θερμόμετρα και leds που προσομοιώνουν τον έλεγχο ενός actuator για τη ροή του θερμαντικού μέσου στα σώματα. Μια (1) πλακέτα έξυπνου θερμοστάτη η οποία εμφανίζει τις τρέχουσες τιμές θερμοκρασίας των χώρων σε μία οθόνη LCD και επιτρέπει τον έλεγχο και την αλλαγή των τρεχουσών θερμοκρασιών με λειτουργία αφής. Μια (1) πλακέτα η οποία επιτρέπει τη σειριακή και στη συνέχεια ασύρματη επικοινωνία ενός PC με τον θερμοστάτη χώρων, και επιτρέπει τον έλεγχο των θερμοκρασιών και μέσω του PC και κατ'επέκταση διαδικτυακά. Η επικοινωνία του PC με την κατασκευή γίνεται μέσω ενός Java Applet που περιλαμβάνει κουμπιά για τον έλεγο και terminal για την παρουσίαση των θερμοκρασιών.

Με τη μεθοδολογία αυτή παρουσιάζεται ένα πλήρως ασύρματο σύστημα αυτονομίας στη θέρμανση μιας κατοικίας, στην οποία ο χρήστης έχει γνώση και τον πλήρη έλεγχο των θερμοκρασιών του κάθε χώρου της κατοικίας.

Η κατασκεύη αυτή αποδεικνύει ότι μπορούν άμεσα και με χαμηλό κόστος να υλοποιηθούν συστήματα που να συνδυάζουν την αύξηση της άνεσης, την μείωση του κόστους θέρμανσης και την βελτίωση του περιβαλλοντικού αποτυπώματος.

Λέξεις κλειδιά

Θέρμανση, αυτονομία, arduino, έξυπνο σπίτι, ασύρματα, δίκτυο RF, θερμόμετρο, DS18B20, NRF24L01, NRF24, αισθητήρες, actuator, java, μικροελεγκτής, atmega, touchscreen

Abstract

The scope of this thesis was the development of an automated model of temperature control in a house based on the principles of thermal autonomy. The structure contains a group of micro-controllers (nodes) which communicate wirelessly, exchanging thermal data and allowing the control of two room temperatures.

Specifically, the four boards consist of: two (2) identical boards to be installed in the thermal bodies, containing electronic thermometers and leds simulating the operating of an actuator controlling the heating medium flow in the thermal bodies. One (1) “smart” thermostat board which displays the current temperature in the rooms on an LCD screen and allows control over them via the touch screen. One (1) board allowing the serial and consecutively the wireless communication of a PC with the thermostat and allows the thermal control over it via PC. The computer communicates with the structure via a Java Applet consisting of buttons for the control of the temperatures and a terminal which prints the current data.

Using this methodology we have a fully wireless autonomous thermal system, providing the user with full thermal control over every separate thermal body (room) of the installation.

KeyWords

Heating, autonomy, arduino, smart home, wireless, RF network, electronics, thermometer, DS18B20, NRF24L01, NRF24, sensors, actuator, java, micro-controller, atmega, touch screen

Στον πατέρα μου

Περιεχόμενα

1.1 Αντικειμενικός σκοπός

Αντικειμενικός σκοπός της παρούσας διπλωματικής εργασίας αποτελεί η προσπάθεια κατασκευής ενός σύγχρονου μοντέλου ελέγχου θέρμανσης εσωτερικών χώρων σύμφωνα με τις τεχνολογικές επιταγές της εποχής, αλλά ταυτόχρονα χαμηλού κόστους. Οι βαθμοί αυτονομίας στη θέρμανση εσωτερικών χώρων έχουν φτάσει στο μέγιστο και η αγορά του εξωτερικού έχει ήδη ανοιχτεί σε "έξυπνες" κατασκευές θερμοστατών και συστημάτων θέρμανσης, οι οποίες όμως έχουν ακόμα υψηλό κόστος για τον πελάτη.

Η κατασκευή απαιτούσε την δημιουργία ενός έξυπνου θερμοστάτη ο οποίος να είναι προσβάσιμος από τον χρήστη με δυο τρόπους, μέσω μιας ψηφιακής οθόνης αλλά και μέσω ενός Η/Υ. Ο χώρος που θεωρήσαμε ως μοντέλο για την ανάπτυξη της κατασκευής είναι ένα κτίριο με δυο δωμάτια, τα οποία περιέχουν ένα σώμα καλοριφέρ σε κάθε δωμάτιο.

Για την αποφυγή τοποθέτησης επιπλέον καλωδίων στους χώρους και προς διευκόλυνση της εγκατάστασής του , επιλέχθηκε να χρησιμοποιηθεί ασύρματη επικοινωνία ανάμεσα στις διαφορετικές συνιστώσες της κατασκευής. Επειδή τα δεδομένα που ανταλλάσσονται μεταξύ των εξαρτημάτων είναι low-level, δε χρειάστηκε να χρησιμοποιήσουμε ασύρματο δίκτυο wifi, αντιθέτως επιλέξαμε τη χρήση δικτύου RF (radio frequency), το οποίο έχει πιο χαμηλού επιπέδου προγραμματισμό, ενώ το hardware είναι σημαντικά φθηνότερο.

1.2 Δομή της διπλωματικής εργασίας

Στο δεύτερο κεφάλαιο γίνεται μια μικρή τεχνική αναδρομή στα συστήματα θέρμανσης, από τις αρχές σχεδόν του προηγούμενου αιώνα, μέχρι σήμερα. Δίνεται μεγάλη έμφαση στα τελευταία μεγάλα τεχνολογικά βήματα στις κατασκευές των συστημάτων θέρμανσης και στις τεχνικές βελτιστοποίησής τους.

Στο τρίτο κεφάλαιο παρουσιάζεται το πρόβλημα που καλούμαστε να λύσουμε, καθώς

και μερικές από τις ιδέες που προτάθηκαν μέχρι να καταλήξουμε στην παρούσα εργασία.

Στο τέταρτο κεφάλαιο παρουσιάζονται οι γενικές γραμμές στις οποίες κινήθηκε η κατασκευή και το πλάνο πάνω στο οποίο βασίστηκε, αναλύονται οι επιμέρους συνιστώσες και η μεταξύ τους διασύνδεση, καθώς και η αρχική σχεδίαση στην πορεία επίλυσης του προβλήματος που τέθηκε.

Στο πέμπτο κεφάλαιο παρουσιάζεται εκτενώς το hardware που χρησιμοποιήθηκε, καθώς και η πορεία κατασκευής των επιμέρους τμημάτων της εργασίας και όσων υλικών χρειάστηκαν.

Στο έκτο κεφάλαιο εξετάζουμε τη σύνδεση κώδικα-hardware των επιμέρους τμημάτων και βλέπουμε τις λειτουργίες τους ξεχωριστά, καθώς και διάφορα προβλήματα που εμφανίστηκαν στην πορεία. Εξηγούνται μερικές βασικές λειτουργίες του κώδικα.

Στο έβδομο κεφάλαιο εμφανίζονται εικόνες από τη λειτουργία της κατασκευής με κάποια σημαντικά κομμάτια του κώδικα που επιτελούν κάποιες από τις επιμέρους λειτουργίες.

Στο όγδοο κεφάλαιο αναφέρονται διάφορα σχόλια για την κατασκευή και τα συμπεράσματα που προκύπτουν από αυτή τη διαδικασία.

Στο παράρτημα παρατίθενται οι πλήρεις κώδικες που γράφτηκαν.

1.3 Ευχαριστίες

Ένα μεγάλο ευχαριστώ στον υπεύθυνο της εργασίας υποψ. Διδάκτορα Απόστολο Κοτοπούλη για όση βοήθεια μου προσέφερε και κυρίως για την επιμονή του να συνεχίσω ακόμα και όταν πίστευα ότι δεν μπορούσα. Ευχαριστώ και στον Ζακ Σολομωνίδη για όση βοήθεια μου προσέφερε στην αρχή. Τέλος, στον πατέρα μου που προθυμοποιήθηκε να βοηθήσει με κάθε δυνατό τρόπο, ειδικά στο ξεκίνημα.

Τα συστήματα θέρμανσης είναι μηχανισμοί διατήρησης της θερμοκρασίας σε ένα προκαθορισμένο επίπεδο, χρησιμοποιώντας θερμική ενέργεια σε ένα σπίτι, γραφείο ή άλλο κτήριο. Είναι συχνά κομμάτι ενός συστήματος HVAC (heating, ventilation & air-conditioning system). Ένα σύστημα θέρμανσης μπορεί να είναι κεντρικό ή διανεμημένο.

2.1 Κεντρικό σύστημα θέρμανσης

Το κεντρικό σύστημα θέρμανσης προσφέρει θερμότητα σε ολόκληρο το εσωτερικό ενός κτιρίου (ή σε κομμάτι του κτιρίου) από ένα σημείο (λεβητοστάσιο), σε πολλούς χώρους. Η κεντρική θέρμανση διαφέρει από την τοπική θέρμανση, στο γεγονός ότι η παραγωγή θερμότητας γίνεται σε ένα συγκεκριμένο χώρο, όπως το λεβητοστάσιο σε ένα σπίτι ή το μηχανοστάσιο σε ένα μεγάλο κτίριο. Η πιο συνηθισμένη μέθοδος παραγωγής θερμότητας περιλαμβάνει την καύση πετρελαίου σε έναν λέβητα. Η παραγόμενη θερμότητα στη συνέχεια διανέμεται συνήθως στη μορφή θερμού νερού μέσω σωληνώσεων. Πολλά κτίρια χρησιμοποιούν συστήματα που λειτουργούν με ηλιακή ενέργεια για να μεταφέρουν τη θερμότητα μέσω θερμού νερού.

Μετά το πέρας του Δευτέρου Παγκοσμίου Πολέμου, τα περισσότερα νέα σπίτια περιλάμβαναν κεντρικό σύστημα θέρμανσης.

2.2 Χαρακτηριστικά

Το κεντρικό σύστημα θέρμανσης σε σπίτια (πολυκατοικίες) αποτελείται από κάποια διακριτά κομμάτια: **τον λέβητα**, που βρίσκεται συνήθως στο λεβητοστάσιο, στα υπόγεια των κτιρίων, **τα θερμαντικά σώματα** (καλοριφέρ), που βρίσκονται διασπαρμένα στα διάφορα δωμάτια ενός διαμερίσματος, **τους σωλήνες** που μεταφέρουν τη θερμότητα στα θερμαντικά σώματα και **τον θερμοστάτη χώρου**, ο οποίος ρυθμίζει αυτόματα τη λειτουργία του συστήματος, διατηρώντας τη θερμοκρασία στα επιθυμητά επίπεδα.

2.2.1 Ο Λέβητας

Λέβητας ονομάζεται κάθε κλειστή μεταλλική συσκευή εντός της οποίας νερό θερμαίνεται με τη βοήθεια θερμότητας. Η θερμότητα παράγεται από την καύση πετρελαίου (ή και ξύλου), με το οποίο τροφοδοτείται, στο εργαζόμενο μέσο που ανακυκλοφορεί μέσα σε σωληνώσεις και μεταφέρει τη θερμότητα αυτή στα θερμαντικά σώματα.

Οι λέβητες διακρίνονται γενικά σύμφωνα με το υλικό κατασκευής τους σε χυτοσίδηρους και χαλύβδινους. Οι πρώτοι αντέχουν καλύτερα στη διάβρωση, μπορούν να επιδεχθούν προσθήκες στοιχείων και χρειάζονται μικρότερες ποσότητες νερού κατά τη λειτουργία τους. Οι χαλύβδινοι έχουν μικρότερο βάρος και αντέχουν καλύτερα στις πιέσεις και στις απότομες αλλαγές θερμοκρασίας. Οι διαστάσεις τους προσαρμόζονται καλύτερα στις διάφορες απαιτήσεις και έχουν χαμηλό κόστος.

2.2.2 Θερμαντικά Σώματα

Τα **θερμαντικά σώματα** είναι εναλλάκτες θερμότητας, που χρησιμοποιούνται για να μεταφέρουν θερμική ενέργεια από ένα μέσο σε ένα άλλο, με σκοπό τη θέρμανση (ή και την ψύξη). Είναι πηγές θερμότητας σε ένα περιβάλλον, αλλά μπορεί να χρησιμοποιούνται γενικότερα όχι μόνο για τη θέρμανση ενός χώρου, αλλά και για τη ψύξη του υγρού που τους παρέχεται, όπως για παράδειγμα στη ψύξη μηχανών.

Τα θερμαντικά σώματα που λειτουργούν με νερό αποτελούνται από ένα κλειστό μεταλλικό δοχείο. Καθώς μεταφέρεται ενέργεια από το νερό στο θερμαντικό σώμα, το νερό

ψύχεται και η ενέργειά του εναποτίθεται στον περιβάλλοντα χώρο.

2.2.2.1 Τύποι θερμαντικών σωμάτων

-Σώματα ΑΚΑΝ ή κλασικά

Είναι τα σώματα που κυριάρχησαν για πολλές δεκαετίες στις εγκαταστάσεις κεντρικής θέρμανσης. Θερμαίνουν περίπου κατά 70 – 80 % με μεταφορά και 20 – 30 % με ακτινοβολία. Η χωρητικότητά τους σε νερό είναι από τις μεγαλύτερες που συναντάμε σε θερμαντικά σώματα και αυτό έχει σαν αποτέλεσμα να αργούν σχετικά να ζεσταθούν, όπως αργούν και να κρυώσουν.

Αρχικά σαν υλικό κατασκευής τους χρησιμοποιείτο ο χυτοσίδηρος, και κατόπιν ο χάλυβας. Κατασκευάζονται σε φέτες οι οποίες ενώνονται μεταξύ τους. Η φέτα στα χαλύβδινα αποτελείται από δύο συγκολλημένα ημικελύφη διαμορφωμένα σε πρέσα κατάλληλα ώστε να δημιουργούνται εσωτερικοί αγωγοί με μορφή στήλης, από τους οποίους θα περνά το ζεστό νερό. Το πάχος της λαμαρίνας τους είναι 0,8 – 1,00 mm.

Οι φέτες κατασκευάζονται σε απόλυτα τυποποιημένες διαστάσεις σε τέσσερα ύψη και τρία πάχη. Όταν στη φέτα διαμορφώνονται δύο στήλες νερού μιλάμε για δίστηλο σώμα και συμβολίζεται με το λατινικό αριθμό ΙΙ, όταν έχουμε τρεις στήλες νερού μιλάμε για τρίστηλο σώμα με συμβολισμό ΙΙΙ, και με τέσσερις στήλες νερού έχουμε τετράστηλο σώμα με συμβολισμό IV. Το πάχος της κάθε φέτας είναι: ΙΙ=85mm, ΙΙΙ=135mm, IV=195mm. Το πλάτος της κάθε φέτας είναι 38mm. Η ισχύς των σωμάτων δίνεται για θερμοκρασία εισόδου του νερού στο σώμα tv=90^{o}C, θερμοκρασία εξόδου tr=70^{o}C και θερμοκρασία χώρου tx=20 ^{o}C.

-Σώματα πάνελ

Κατασκευάζονται από χαλύβδινα ελάσματα πάχους 1,1 - 1,25 mm, διαμορφωμένα σε πρέσα και συγκολλημένα μεταξύ τους ώστε να δημιουργείται μια υδροφόρα πλάκα. Το κάθε σώμα αποτελείται από μία έως τρεις υδροφόρες πλάκες. Μεταξύ των πλακών συνήθως τοποθετείται μαίανδρος συγκολλημένος στις πλάκες για αύξηση της θερμαινόμενης επιφάνειας. Στο επάνω μέρος και στα πλάγια τοποθετούνται καλύμματα για αισθητικούς λόγους. Διατίθενται βαμμένα ηλεκτροστατικά σε χρώμα λευκό και κάποιες φορές, κατόπιν παραγγελίας, και σε άλλα χρώματα.

Τα σώματα πάνελ περιγράφονται με τρεις αριθμούς. Ο πρώτος αριθμός είναι διψήφιος, το πρώτο νούμερό του αναφέρεται στο πλήθος των πλακών και το δεύτερο στο πλήθος των μαιάνδρων. Όταν μιλάμε για σώμα π.χ. 22, εννοούμε ότι το σώμα αυτό έχει δύο πλάκες και δύο μαιάνδρους. Ο δεύτερος αριθμός του σώματος μας δίνει το συνολικό ύψος του σώματος σε χιλιοστά του μέτρου (όχι την απόσταση των κέντρων σύνδεσης όπως στα ΑΚΑΝ). Ο τρίτος αριθμός μας δίνει το μήκος του σώματος σε χιλιοστά του μέτρου.

-Κονβέκτορες (convectors)

Τα σώματα αυτά κατασκευάζονται από χάλκινες σωλήνες, πάνω στις οποίες είναι προσαρμοσμένα πτερύγια αλουμινίου για αύξηση της θερμαινόμενης επιφάνειας. Η καλή επαφή χαλκοσωλήνα και φύλλου αλουμινίου εξασφαλίζεται με μηχανική εκτόνωση του χαλκού. Περιφερειακά το σώμα περιβάλλεται από βαμμένα καλύμματα, ενώ στο πάνω μέρος φέρει σχάρα για την ελεύθερη διέλευση του θερμού αέρα. Τα περιφερειακά καλύμματα συντελούν και στην αύξηση της ταχύτητας του αέρα μέσα από το σώμα, βοηθώντας έτσι στην αύξηση της απόδοσης θερμότητας .

Τα σώματα αυτά θερμαίνουν περίπου κατά 95% με μεταφορά και μόλις κατά 5% με ακτινοβολία. Η πίεση λειτουργίας τους μπορεί να φθάσει στα 20 bar. Η στήριξή τους μπορεί να γίνει στον τοίχο αλλά και στο δάπεδο με τα κατάλληλα στηρίγματα. Επίσης μπορούν να τοποθετηθούν σε κανάλι καταλλήλων διαστάσεων, διαμορφωμένο μέσα στο δάπεδο.

-Άλλα είδη σωμάτων

Τα σώματα αλουμινίου είναι κομψά σώματα, με μικρό όμως μερίδιο της αγοράς λόγω της υψηλής τους τιμής. Κατασκευάζονται σε φέτες με χύτευση κραμάτων αλουμινίου σε καλούπια. Βάφονται ηλεκτροστατικά και έχουν λεία επιφάνεια. Οι φέτες ενώνονται μεταξύ τους με χαλύβδινα νίπελ. Κατασκευάζονται σε διάφορα ύψη, ενώ το μήκος τους καθορίζεται από τον αριθμό των φετών.

Τα σώματα runtal χρησιμοποιούνται πλέον ελάχιστα, καθώς έχουν εκτοπιστεί σε μεγάλο βαθμό από τα πάνελ. Αποτελούνται από χαλύβδινους πεπλατυσμένους οριζόντιους σωλήνες, συνδεδεμένους στα άκρα τους με κατακόρυφους συλλέκτες.

Οι fan convectors φέρουν, επιπλέον των απλών convectors, ανεμιστήρα στο κάτω μέρος τους για δυναμική εκφόρτιση του θερμαντικού στοιχείου. Τα πτερύγια τώρα είναι τοποθετημένα πιο πυκνά, μια και δε στηριζόμαστε στη φυσική κυκλοφορία του αέρα, η οποία

θα παρεμποδιζότανε με μια τέτοια κατασκευή. Με τα fan convectors επιτυγχάνουμε μεγάλη θερμική ισχύ από μικρό μέγεθος σώματος.

Τα σώματα fan coils είναι μια επέκταση της ιδέας των fan convectors με τη διαφορά ότι τα fan coils δουλεύουν τόσο με ζεστό νερό για τη θέρμανση του χώρου, όσο και με κρύο νερό για την ψύξη του.

Τα σώματα λουτρού εμφανίστηκαν σαν σώματα ειδικών διαστάσεων για τοποθέτηση σε λουτρά τα οποία είναι συνήθως περιορισμένων διαστάσεων.

2.2.3 Σύστημα Μεταφοράς

Το **σύστημα μεταφοράς** χωρίζεται σε δυο κατηγορίες: μονοσωλήνιο και δισωλήνιο. Δισωλήνιο είναι το σύστημα όπου ξεκινάνε δυο σωλήνες για κάθε σώμα, και καταλήγουν η μία στον συλλέκτη προσαγωγής και η άλλη στον συλλέκτη επιστροφής. Στο σύστημα αυτό τα σώματα είναι συνδεδεμένα παράλληλα. Μονοσωλήνιο είναι το σύστημα στο οποίο όλα τα σώματα σε κάθε εγκατάσταση είναι συνδεδεμένα σε σειρά.

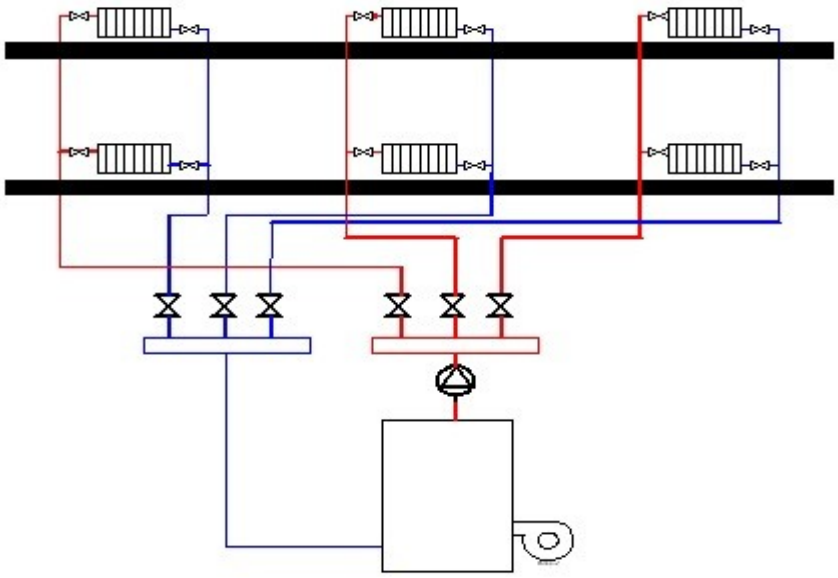
2.2.3.1 Το Δισωλήνιο Σύστημα

Στο **δισωλήνιο σύστημα** απαιτούνται δύο σωλήνες ξεχωριστοί για την τροφοδοσία κάθε συστήματος (προσαγωγή-επιστροφή). Συνήθως, αυτό το σύστημα στη φυσιολογική του μορφή αποτελείται από ένα οριζόντιο δίκτυο σωληνώσεων (διπλών) , το οποίο εκτείνεται από το λεβητοστάσιο ως τα άκρα της πλάκας της οικοδομής και συνεχίζει κατακόρυφο πλέον, με μορφή στηλών και τροφοδοτεί τα σώματα των ορόφων. Γενικά σε ένα δισωλήνιο σύστημα έχουμε πολλές στήλες (διπλές), οι οποίες βρίσκονται σε παράλληλη σύνδεση μεταξύ τους, ενώ σε κάθε στήλη έχουμε πολλά σώματα, που βρίσκονται σε παράλληλη σύνδεση μεταξύ τους.Αυτό είναι και το βασικό χαρακτηριστικό του δισωλήνιου συστήματος θέρμανσης.

Ένα δεύτερο χαρακτηριστικό του είναι ότι λόγω της παράλληλης σύνδεσης των στηλών και των σωμάτων έχουμε θεωρητικά ίδια θερμοκρασία σε όλα τα σώματα , όταν φυσικά είναι μονωμένες οι σωληνώσεις σε όλη τους τη διαδρομή. Ένα τρίτο χαρακτηριστικό αφορά την ταχύτητα του νερού στο δισωλήνιο, η οποία είναι χαμηλή και δεν πρέπει να ξεπεράσει τα 0,6m/sec. Το φαινόμενο αυτό οφείλεται στη σχετικά άγρια εσωτερική επιφάνεια

των συμβατικών σωλήνων που συνήθως χρησιμοποιούνται σε αυτό το σύστημα.

Υπάρχει, βέβαια, και το δισωλήνιο σύστημα που είναι γνωστό στην πράξη με το όνομα **ομπρέλα**. Το σύστημα αυτό συνήθως απευθύνεται σε οικοδομές έτοιμες ή σε σπίτια που ήδη κατοικούνται, όπου δεν έχει γίνει πρόβλεψη για κεντρική θέρμανση και θέλουμε να τα θερμάνουμε εκ των υστέρων. Το σύστημα αυτό αποτελείται από ένα οριζόντιο δίκτυο σωληνώσεων (διπλών), το οποίο απλώνεται στη συμβολή των κατακόρυφων τοίχων με την οροφή και πριν από κάθε σώμα δημιουργείται κατακόρυφη διακλάδωση που έχει κατεύθυνση από πάνω προς τα κάτω για την τροφοδοσία του σώματος.



Εικόνα 2.1: Σχεδιάγραμμα Δισωλήνιου Συστήματος

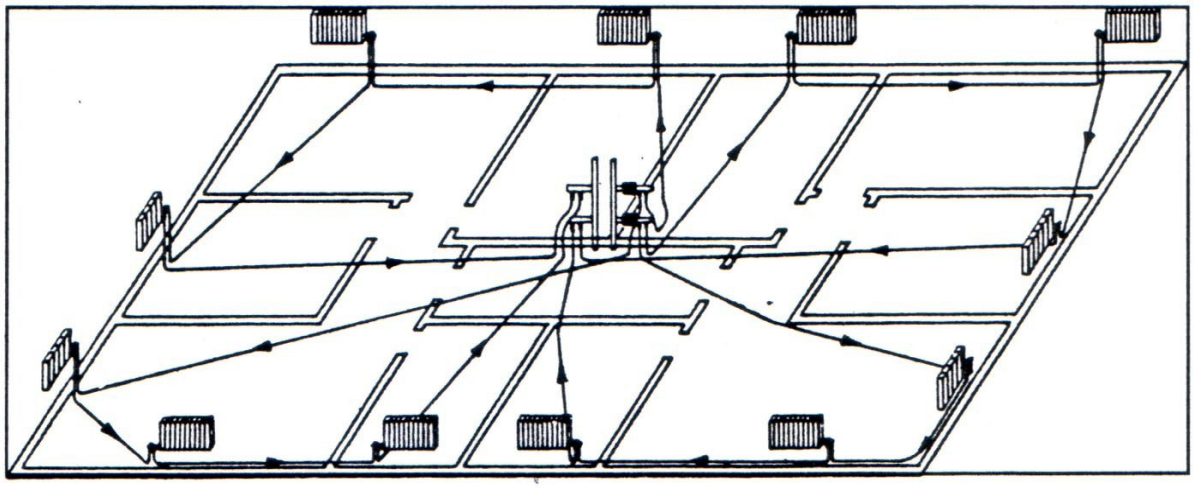
2.2.3.2 Το Μονοσωλήνιο Σύστημα

Στο **μονοσωλήνιο σύστημα** υπάρχει μία κεντρική στήλη (ζεύγος σωλήνων προσαγωγής -επιστροφής) που ξεκινά από το λεβητοστάσιο και με κατεύθυνση από κάτω προς τα πάνω, τροφοδοτεί τον συλλέκτη προσαγωγής με ζεστό νερό από τον λέβητα και παραλαμβάνει από τον συλλέκτη επιστροφής νερό χαμηλότερης θερμοκρασίας ($\Delta T=20^{o}C$), που έχει κάνει τη "διαδρομή" του στα θερμαντικά σώματα του σπιτιού, και το επιστρέφει στον λέβητα.

Η κατακόρυφη στήλη τοποθετείται κάπου στο κέντρο της οικοδομής π.χ. στο κλιμακοστάσιο ή μέσα στο φωταγωγό ή και σε οποιαδήποτε άλλη θέση η οποία θα εξυπηρετεί τη λειτουργικότητα του συστήματος και δεν θα δημιουργεί αισθητικά ή λειτουργικά

προβλήματα στους χώρους της πολυκατοικίας. Για λόγους οικονομίας και αισθητικής συνηθίζεται η θερμομόνωση της κατακόρυφης στήλης και η κάλυψή της με γυψοσανίδα ή νευρομετάλ και σοβά. Η κατακόρυφη κεντρική στήλη διέρχεται από όλους τους ορόφους. Σε κάθε όροφο ανάλογα με το μέγεθός του και τον αριθμό διαμερισμάτων, τοποθετούνται ένας ή περισσότεροι συλλέκτες (προσαγωγής-επιστροφής) σε ύψος 30-50εκ. από το δάπεδο του ορόφου.

Το μονοσωλήνιο σύστημα χαρακτηρίζεται από κυκλώματα-βρόχους μέχρι 3-4 θερμαντικά σώματα το καθένα. Από τον συλλέκτη προσαγωγής ξεκινάει ένας σωλήνας με προορισμό το πρώτο σώμα. Εκεί, ο διακόπτης του πρώτου σώματος στέλνει μία ποσότητα του νερού στο σώμα (όπου περνώντας μέσα από τις πτυχώσεις του,αποδίδεται σε θερμότητα στο περιβάλλον). Την υπόλοιπη ποσότητα ζεστού νερού, αφού αναμιχθεί με το νερό επιστροφής του πρώτου σώματος, την κάνει bypass κατευθείαν στο επόμενο σώμα. Δηλαδή, οι 2 ποσότητες νερού, η μία με θερμοκρασία περίπου ίση με της κεντρικής στήλης και η άλλη μειωμένη εξαιτίας της διέλευσης μέσα από το πρώτο σώμα κατευθύνονται στο δεύτερο σώμα, όπου γίνεται ακριβώς η ίδια διαδικασία μέχρι και το τελευταίο σώμα του συγκεκριμένου κυκλώματος,από το οποίο επιστρέφει το νερό στον συλλέκτη επιστροφής.



Σχηματική διάταξη οριζόντιου δικτύου σωληνώσεων σε εγκατάσταση μονοσωλήνιου συστήματος από την agpa

Εικόνα 2.2: Σχεδιάγραμμα Μονοσωλήνιου Συστήματος

Όπως καταλαβαίνουμε στο μονοσωλήνιο σύστημα, τα σώματα είναι συνδεδεμένα σε σείρα. Χαρακτηριστικό του συστήματος είναι ότι τα σώματα που είναι δεύτερα, τρίτα και

τελευταία στη σειρά σε ένα βρόγχο, πρέπει να προσαυξάνονται ως προς το μέγεθός τους για έχουν την επιθυμητή απόδοση, καθώς το νερό που φτάνει σε αυτά είναι χαμηλότερης θερμοκρασίας.

Σε κάθε υποδοχή των συλλεκτών (προσαγωγή-επιστροφή) τοποθετείται από μία ρυθμιστική βαλβίδα η οποία διευκολύνει την εξισορρόπηση των πιέσεων κάθε κυκλώματος ή ακόμα και την τέλεια απομόνωσή του αν παραστεί ανάγκη, επιτρέποντας τη λειτουργία του υπόλοιπου διαμερίσματος και της πολυκατοικίας έως ότου βρεθεί μάστορας να επέμβει. Η ρυθμιστική βαλβίδα φέρει ειδικό σύνδεσμο (αυτοεκτονούμενο ρακόρ) ο οποίος συνδέει τον ειδικό σωλήνα που τροφοδοτεί τα θερμαντικά σώματα. Ο σωλήνας τροφοδοτεί "εν σειρά" τα σώματα κάθε διαμερίσματος με ένα ή περισσότερα κυκλώματα ο αριθμός των οποίων εξαρτάται από το μέγεθος και τις απώλειες που εμφανίζει το ίδιο το διαμέρισμα.

Ο χρησιμοποιούμενος σωλήνας είναι ειδικής κατασκευής. Αποτελείται από δύο ομόκεντρους σωλήνες κατασκευασμένους από ανθεκτικό, σε υψηλές θερμοκρασίες, πιέσεις και μηχανικές καταπονήσεις πλαστικό υλικό. Ο εξωτερικός σωλήνας προστατεύει τον εσωτερικό και ταυτόχρονα δίνει τη δυνατότητα αντικατάστασης του κύριου εσωτερικού σωλήνα, σε περίπτωση βλάβης, χωρίς την ταλαιπωρία και τα έξοδα οικοδομικών επισκευών.

Το υλικό από το οποίο είναι κατασκευασμένοι οι σωλήνες, αποκλείει τα φαινόμενα της οξείδωσης και της ηλεκτρόλυσης, και συνεπώς τα προβλήματα που παρουσιάζονται σε εγκαταστάσεις με σωλήνες κατασκευασμένου από χάλυβα ή χαλκό. Έτσι οι σωλήνες, οι λέβητες και τα θερμαντικά σώματα, είναι απρόσβλητα από τη συνηθισμένη σε τέτοιες εγκαταστάσεις διάβρωση. Ο σωλήνας απλώνεται πάνω στο δάπεδο και στερεώνεται με πλαστικά ή μεταλλικά κολλάρα ανά ένα έως δύο μέτρα. Δεν απαιτούνται κλίσεις, ούτε υπάρχουν προβλήματα συστοδιαστολών, αφού οι μικρές συστοδιαστολές που υπάρχουν απορροφούνται απ' τον εξωτερικό προστατευτικό σωλήνα.

Τα θερμαντικά σώματα τροφοδοτούνται με ειδικούς διακόπτες, οι οποίοι επιτρέπουν σε μέρος μόνο του διερχομένου νερού να τροφοδοτήσει το σώμα ενώ το υπόλοιπο αναμειγνύεται με αυτό που εξέρχεται απ' το θερμαντικό σώμα με αποτέλεσμα την σχεδόν ταυτόχρονη ενεργοποίηση όλων των θερμαντικών σωμάτων. Επίσης είναι κατάλληλα κατασκευασμένοι έτσι ώστε όταν διακοπεί η παροχή προς ένα σώμα, αφενός μεν να επιτρέπου τη διέλευση του νερού για να τροφοδοτούνται τα επόμενα σώματα αφ' ετέρου δε την πλήρη στεγανοποίηση των εξόδων τους προς το σώμα για να μπορεί αυτό να αφαιρεθεί.

Το μονοσωλήνιο σύστημα θέρμανσης επιτρέπει ακόμα με μικρό πρόσθετο κόστος και με την βοήθεια ηλεκτροκίνητης βάνας, θερμοστάτη χώρου και μετρητή την

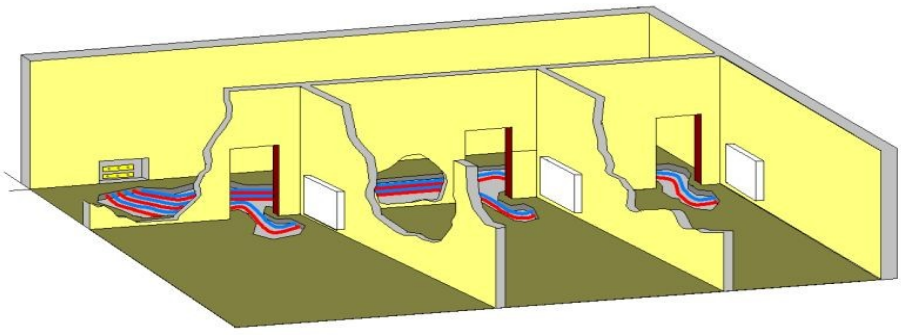
ανεξαρτητοποίηση της θέρμανσης κάθε διαμερίσματος από κάθε άποψη (επιθυμητής θερμοκρασίας χώρου, χρόνου λειτουργίας, κατανάλωσης κ.λ.π.).

Το μονοσωλήνιο σύστημα θέρμανσης αποδείχθηκε ως η πλέον βέλτιστη οικονομικά και κατασκευαστικά λύση. Η καθιέρωση του μονοσωληνίου συστήματος προέκυψε από τη δυνατότητα ανεξάρτητης θέρμανσης των διαμερισμάτων του κτιρίου με ένα λέβητα, καυστήρα, κυκλοφορητή και ένα μόνο ζεύγος κεντρικών στηλών (προσαγωγή-επιστροφή) καθώς και συλλέκτες σε κάθε διαμέρισμα στους οποίους συνδέονται οι προσαγωγές και επιστροφές των θερμαντικών επιφανειών.

Η εγκατάσταση του λεβητοστασίου δεν παρουσιάζει βασικές διαφορές από αυτή του παλιού κλασσικού δισωληνίου συστήματος. Το μονοσωλήνιο σύστημα λύνει και το βασικό πρόβλημα των πολλών κατακόρυφων στηλών που αναγκαστικά εγκαθίστανται στο κλασσικό σύστημα κεντρικής θέρμανσης που είναι αντιλειτουργικές.

2.2.3.3 Το Ακτινωτό Δισωλήνιο Σύστημα

Το **ακτινωτό δισωλήνιο** είναι ένα σύστημα που έχει κοινά σημεία και με το δισωλήνιο και με το μονοσωλήνιο και ονομάζεται έτσι ,επειδή η σύνδεση των σωμάτων με την κεντρική στήλη γίνεται ακτινωτά. Υπάρχει μία κεντρική στήλη με συλλέκτες διανομής νερού σε κάθε όροφο, όπως στο μονοσωλήνιο με τη διαφορά ότι σε αυτό κάθε βρόγχος (κύκλωμα) τροφοδοτεί ένα μόνο σώμα.



Εικόνα 2.3: Σχεδιάγραμμα Ακτινωτού Δισωλήνιου Συστήματος

Τα σώματα σε αυτό το σύστημα έχουν θεωρητικά την ίδια θερμοκρασία και βέβαια, επειδή οι χρησιμοποιούμενοι σωλήνες τροφοδοσίας έχουν ιδιαίτερα λεία εσωτερική

επιφάνεια, δίνουν τη δυνατότητα στο νερό να έχει ταχύτητα από 0,6-1,2m/sec. Στις οριζόντιες διαδρομές δεν απαιτούνται κλίσεις. Το σύστημα είναι σαν ένα δισωλήνιο, το οποίο έχει μία μόνο κατακόρυφη στήλη .

Το βασικό του πλεονέκτημα είναι ότι συνδυάζει μεγάλη ταχύτητα νερού και σταθερή και υψηλή θερμοκρασία σωμάτων (δε χρειάζονται προσαυξήσεις στην επιφάνεια των σωμάτων). Το βασικό του μειονέκτημα είναι ότι απαιτεί πολλά μέτρα σωλήνα (60-70% περισσότερα από το μονοσωλήνιο), άρα και υψηλότερο κόστος κατασκευής.

2.2.4 Διακόπτες Σωμάτων

Στα θερμαντικά σώματα τοποθετούμε διακόπτες για δύο λόγους, για να διακόψουμε την παροχή του νερού στο σώμα, προκειμένου να το επισκευάσουμε, να το αντικαταστήσουμε ή να διακόψουμε τη θέρμανση του χώρου και ο άλλος λόγος είναι να ρυθμίσουμε την παροχή του νερού ώστε το σώμα να έχει την αναμενόμενη μέση θερμοκρασία.

Αυτός ο δεύτερος λόγος είναι συχνά πολύ σημαντικός, γιατί αν κάποια σώματα τροφοδοτούνται με μεγαλύτερη παροχή απ' όση χρειάζονται, είναι πολύ πιθανό σε κάποια άλλα σώματα η παροχή του νερού να μειωθεί δραστικά, οπότε τα σώματα αυτά να υπολειτουργούν και ο χώρος να μη θερμαίνεται ικανοποιητικά.

2.2.4.1 Τύποι διακοπτών

-Διακόπτες δισωληνίου συστήματος.

Κατασκευάζονται ίσιοι ή γωνιακοί. Οι περισσότεροι κλείνουν με έμβολο, αλλά υπάρχουν και σφαιρικοί, οι οποίοι παρουσιάζουν πολύ μικρή αντίσταση στη ροή του νερού. Επίσης κατασκευάζονται και με διπλή ρύθμιση. Στους διακόπτες με διπλή ρύθμιση μπορούμε να καθορίσουμε εμείς το μέγιστο άνοιγμα. Αυτό γίνεται, αφαιρώντας το καπάκι τους και στρέφοντας ένα παξιμάδι ή τοποθετώντας ένα πιράκι σε κάποια από τις υποδοχές που έχει εσωτερικά το πλαστικό καπάκι, έτσι ώστε ο ιδιοκτήτης να μην μπορεί να ανοίξει το διακόπτη περισσότερο από όσο τον έχουμε ρυθμίσει. Έτσι το σώμα δεν υπερτροφοδοτείται με νερό σε βάρος κάποιων άλλων.

-Διακόπτες μονοσωληνίου.

Είναι τετράοδοι διακόπτες με τους οποίους μπορούμε: Να στείλουμε στο σώμα όλο το νερό που διατρέχει το βρόχο. Να στείλουμε ένα ποσοστό του νερού στο σώμα. Να παρακάμψουμε εντελώς το σώμα χωρίς να επηρεασθεί η παροχή προς τα υπόλοιπα σώματα.

Οι διακόπτες μονοσωληνίου διακρίνονται σε: Εσωτερικού βρόχου, που τους χρησιμοποιούμε όταν έχουμε σώματα τύπου ΑΚΑΝ ή αλουμινίου. Εξωτερικού βρόχου, που χρησιμοποιούνται σε σώματα τύπου πάνελ ή ρούνταλ. Υπάρχουν σώματα πάνελ που έχουν ενσωματωμένο τον εξωτερικό βρόχο. Προκειμένου να συνδεθούν αυτά τα σώματα χρησιμοποιούμε ειδικούς διακόπτες μονοσωληνίου, τύπου Η, και έτσι αποφεύγουμε το εξωτερικό σωληνάκι.

-Θερμοστατικές κεφαλές.

Συχνά δημιουργείται η ανάγκη για ρύθμιση διαφορετικής θερμοκρασίας σε κάθε χώρο. Για παράδειγμα σε λουτρά ή υπνοδωμάτια πολλοί επιθυμούν μεγαλύτερη θερμοκρασία, ενώ σε αίθουσες γυμναστικής, η θερμοκρασία πρέπει να είναι μικρότερη. Ο πιο απλός τρόπος για την ικανοποίηση της παραπάνω ανάγκης είναι η χρήση διακοπτών με θερμοστατική κεφαλή. Η θερμοστατική κεφαλή αποτελείται από τύμπανα πεπλατυσμένα που εσωτερικά φέρουν υγρό, το οποίο θερμαινόμενο από τον αέρα του χώρου διαστέλλεται και αυξάνει το πάχος των τυμπάνων. Η διόγκωση των τυμπάνων γίνεται αιτία μετακίνησης ενός εμβόλου, το οποίο ελέγχει την ποσότητα του νερού που διέρχεται από το σώμα. Έτσι η θερμοκρασία του σώματος εξαρτάται από την παροχή του νερού και η ισχύς του μεταβάλλεται ανάλογα, επιτρέποντάς μας, με την κατάλληλη ρύθμιση της θερμοστατικής κεφαλής, να ρυθμίσουμε τη θερμοκρασία του χώρου. Η θερμοστατική κεφαλή πρέπει να τοποθετείται με τον άξονά της οριζόντιο.

2.2.5 Θερμοστάτης Χώρου

Ο θερμοστάτης χώρου είναι η συσκευή ή το σύστημα ελέγχου, που μετρά τη θερμοκρασία ενός κλειστού συστήματος (στην περίπτωσή μας ενός χώρου), ρυθμίζοντάς τη κοντά στην επιθυμητή προεπιλογή. Ο θερμοστάτης λειτουργεί θέτοντας το σύστημα θέρμανσης σε λειτουργία και κλείνοντάς το ή ρυθμίζοντας τη ροή του θερμού νερού όπως χρειάζεται, για να διατηρείται η επιθυμητή θερμοκρασία.

Οι θερμοστάτες μπορούν να κατασκευάζονται με πολλούς τρόπους και μπορούν να χρησιμοποιούν μεγάλη ποικιλία αισθητήρων για να μετρούν τη θερμοκρασία. Το output του αισθητήρα ελέγχει το σύστημα θέρμανσης.

Ο θερμοστάτης πρέπει να τοποθετείται μακριά από τα θερμαντικά σώματα του κτιρίου, αλλά να βρίσκεται σε σημείο τέτοιο ώστε να λαμβάνει μετρήσεις από τον χώρο που ελέγχει. Ένας ανοιχτός διάδρομος είναι η συνήθης θέση του θερμοστάτη για ένα κλειστό σύστημα μιας ζώνης, όπου σαλόνια και υπνοδωμάτια ρυθμίζονται από έναν μόνο θερμοστάτη. Αν ο θερμοστάτης βρίσκεται κοντά σε θερμαντικό σώμα, τότε το σύστημα έχει την τάση να "βραχυκυκλώνει" και συνεπώς να οδηγείται σε συχνές διακοπές και επανεκκινήσεις του συστήματος, διαδικασία που μπορεί να αποδειχθεί όχι μόνο ενοχλητική για τους ενοίκους αλλά να οδηγήσει σε καταπόνηση του εξοπλισμού και οπωσδήποτε σε ενεργειακή σπατάλη. Αντίθετα η ιδέα μας είναι ένα σύστημα πολλαπλών ζωνών το οποίο μπορεί να εξοικονομήσει ενέργεια, έχοντας θερμοστάτες να ρυθμίζουν τα θερμαντικά σώματα αυτόνομων περιοχών, επιτρέποντας τη θέρμανση συγκεκριμένων δωματίων και αγνοώντας δωμάτια που δε χρησιμοποιούνται οδηγώντας έτσι σε μεγιστοποίηση της απόδοσης του συστήματος θέρμανσης καθώς και σημαντική εξοικονόμηση ενέργειας.

Υπάρχουν πολλά είδη θερμοστάτη, το πιο συνηθισμένο παλιότερα ήταν ο απλός "αναλογικός" θερμοστάτης με ποτενσιόμετρο ρύθμισης. Τελευταία, μεγάλο μέρος της αγοράς έχουν λάβει οι ψηφιακοί και ηλεκτρονικοί/έξυπνοι θερμοστάτες.

2.2.5.1 Έξυπνοι Θερμοστάτες

Η νέα γενιά θερμοστατών συνήθως αναφέρονται ως "έξυπνοι" θερμοστάτες, πέρα από τις παραδοσιακές ρυθμίσεις που παρέχουν, μπορούν να προγραμματίζονται ή να επιτρέπουν τον χειρισμό τους από απόσταση (είτε μέσω ασύρματου δικτύου GSM ή μέσω internet)

Ως παράδειγμα έξυπνου θερμοστάτη μπορούμε να αναφέρουμε τον Nest. Ο **Nest**

χρησιμοποιεί σύστημα dial για τη ρύθμιση της θερμοκρασίας αντί για τα παραδοσιακά κουμπιά. Μια εβδομάδα λειτουργίας είναι αρκετή για να μάθει το καθημερινό πρόγραμμα θέρμανσης και να αρχίσει να ρυθμίζει μόνος του τη θερμοκρασία. Μεταξύ άλλων, ο έξυπνος θερμοστάτης γνωρίζει πότε ο χρηστης βρίσκεται εκτός σπιτιού, δείχνει πόση ώρα χρειάζεται για να ζεσταθεί το σπίτι, ενημερώνει για το πόση ενέργεια έχει καταναλωθεί και βοηθά στην εξοικονόμηση της. Η συσκευή κάνει χρήση WiFi, ενώ μπορεί να ρυθμίζεται μέσω υπολογιστή, tablet ή smartphone. Εκτός των άλλων, ειδική ένδειξη (ένα φύλλο) ενημερώνει τον χρήστη κάθε φορά που εξοικονομεί ενέργεια.

Εικόνα 2.4: Ο Θερμοστάτης Nest

Η αγορά βέβαια έχει διευρυνθεί, παρουσιάζοντας wifi θερμοστάτες σε τιμές γύρω στα $100 (ενδεικτικές οι εταιρείες Filtrete και Radio Thermostat Company of America). Ένα ακόμα παράδειγμα είναι ο Smart SI θερμοστάτης της ecobee που επίσης προσφέρει δυνατότητα ελέγχου του θερμοστάτη μέσω PC ή smartphone. Η εταιρεία honeywell προσφέρει wifi θερμοστάτη στην τιμή των $350. Ο θερμοστάτης αυτός μπορεί να ελέγχει και τις εξωτερικές περιβαλλοντικές συνθήκες και να ρυθμίζει ανάλογα τη θερμοκρασία στο εσωτερικό της εγκατάστασης. Άλλες εταιρείες, όπως η Icontrol network παρέχει ολοκληρωμένο σύστημα αυτονομίας, που περιλαμβάνει και προγραμματιζόμενο θερμοστάτη.

Στην Ελλάδα, ενδεικτικό παράδειγμα είναι η εταιρεία Crystal Audio, που παρέχει ασύρματο θερμοστάτη (μοντέλο: PH-TS20) στην τιμή των 65€.

2.2.6 Αυτονομία Θέρμανσης

Τα περισσότερα κτίρια στην Ελλάδα χρησιμοποιούν κεντρικό σύστημα θέρμανσης. Παρ'όλα αυτά, η κεντρική θέρμανση έχει αποδειχτεί πολλές φορές ακριβή και μη-αποδοτική. Για παράδειγμα, δεν είναι αποδοτικό να θερμαίνεται ένα δωμάτιο ώρες στις οποίες ο κάτοικος απουσιάζει ή όταν η θερμοκρασία του δωματίου είναι η επιθυμητή. Αντίθετα λοιπόν από την κεντρική θέρμανση, είναι πιο αποδοτική η μετατροπή του συστήματος σε αυτόνομο. Με αυτόν τον τρόπο, είναι δυνατό να θερμαίνεται αυτόνομα ένα μόνο διαμέρισμα ή ακόμα και συγκεκριμένα δωμάτια του διαμερίσματος αυτού.

2.2.6.1 Αυτονομία θέρμανσης με Ωρομετρητές

Ένα ποσοστό του καταναλωθέντος πετρελαίου (αν έτσι ορίζεται από τον κανονισμό της πολυκατοικίας) κατανέμεται ανάλογα με τα χιλιοστά κάθε ιδιοκτησίας. Το υπόλοιπο κατανέμεται ανάλογα με με τα χιλιοστά της ιδιοκτησίας σε συνάρτηση με τις ώρες χρήσεως της θέρμανσης από το αντίστοιχο διαμέρισμα σύμφωνα με τον τύπο:

$$K = \Pi * \frac{X\delta * \Omega\delta}{\sum X\delta * \Omega\delta}$$

Όπου:

Κ = η ζητούμενη κατανάλωση του διαμερίσματος σε Ευρώ.

Π = η συνολική κατανάλωση πετρελαίου της πολυκατοικίας σε Ευρώ.

Χδ = τα χιλιοστά του διαμερίσματος.

Ωδ = ή μέτρηση του διαμερίσματος το διάστημα πού μας ενδιαφέρει.

$\sum X\delta * \Omega\delta$ = Το άθροισμα των γινομένων των ωρών επί τα χιλιοστά όλων των διαμερισμάτων.

2.2.6.2 Αυτονομία θέρμανσης με υπολογισμό των θερμικών απωλειών

Στην περίπτωση αυτή οι δαπάνες πού αφορούν την κατανάλωση πετρελαίου κατανέμονται με βάση την μελέτη του Μηχανολόγου Μηχανικού πού έχει σχεδιάσει το σύστημα κεντρικής θέρμανσης της πολυκατοικίας. Ο Μηχανικός έχει υπολογίσει στην μελέτη

του τις σχετικές μεταβλητές οι οποίες χρησιμοποιούνται στον παρακάτω τύπο σε συνάρτηση με τον χρόνο που κάθε ιδιοκτησία χρησιμοποιεί την θέρμανση, ως ακολούθως:

$$K = \Pi * \left[fi * ei + \frac{\Omega\delta * ei}{\sum \Omega\delta * ei} * \left(1 - \sum_{i}^{\cdot} fi * ei \right) \right]$$

Όπου:

Κ = η ζητούμενη κατανάλωση του διαμερίσματος σε Ευρώ.

Π = η συνολική κατανάλωση πετρελαίου της πολυκατοικίας σε Ευρώ.

ei = ο συντελεστής επιβάρυνσης του διαμερίσματος.

fi = ο συντελεστής παραμένουσας επιβάρυνσης του διαμερίσματος.

Ωδ = ή μέτρηση του διαμερίσματος το διάστημα πού μας ενδιαφέρει.

$\sum ei * \Omega\delta$ = Το άθροισμα των γινωμένων των ωρών επί συντελεστές επιβάρυνσης όλων των διαμερισμάτων.

2.2.6.3 Αυτονομία θέρμανσης με θερμιδομετρητές

Ένα ποσοστό του καταναλωθέντος πετρελαίου (αν έτσι ορίζεται από τον κανονισμό της πολυκατοικίας) κατανέμεται ανάλογα με τα χιλιοστά κάθε ιδιοκτησίας. Το υπόλοιπο κατανέμεται ανάλογα με τις ενδείξεις των θερμιδομετρητών σύμφωνα με τον τύπο:

$$K = \Pi * \frac{\Theta\delta}{\sum \Theta\delta}$$

Όπου:

Κ = η ζητούμενη κατανάλωση του διαμερίσματος σε Ευρώ.

Π= η συνολική κατανάλωση πετρελαίου της πολυκατοικίας σε Ευρώ.

Θδ= Η κατανάλωση του διαμερίσματος σε θερμομονάδες.

$\sum \Theta\delta$ = Το άθροισμα των καταναλώσεων σε θερμομονάδες όλων των διαμερισμάτων.

2.2.7 Θερμιδομέτρηση

Δεν είναι τυχαίο ότι στην Ελλάδα αλλά και στις υπόλοιπες Ευρωπαικές χώρες έχει απογορευθεί τα τελευταία χρόνια η εγκατάσταση ωρομετρητών αυτονομίας για την κατανομή των δαπανών θέρμανσης στις νέες κατασκευές. Παρόλο που μια εγκατάσταση αυτονομίας που χρησιμοποιεί την τεχνολογία των ωρομετρητών για την κατανομή των δαπανών θέρμανσης στα διαμερίσματα θεωρείται σύγχρονη λύση, αυτό δεν είναι απαραίτητα αλήθεια. Χρησιμοποιώντας τους ωρομετρητές, κάνουμε απλά μια εκτίμηση του πραγματικού κόστους θέρμανσης και σε καμία περίπτωση έναν ακριβή υπολογισμό. Ετσι δημιουργούνται αμφισβητήσεις και διενέξεις ανάμεσα στους ενοίκους των διαμερισμάτων και δεν γίνεται δίκαιη κατανομή του κόστους θέρμανσης. Η λύση των ωρομετρητών θεωρείται πλέον παρωχημένη σε όλη την Ευρώπη και έχει σταματήσει πλέον η τοποθέτησή τους.

Το σύστημα της θερμιδομέτρησης είναι ο πλέον ιδανικός τρόπος για την αυτονόμηση διαμερισμάτων παλαιών πολυκατοικιών, αλλά και ο ορθός για την κατανομή δαπανών. Οι θερμιδομετρητές έχουν την ικανότητα να αποστέλλουν τα δεδομένα που συλλέγουν ασύρματα εκτός του κτιρίου όταν τους ζητηθεί. Γνωρίζοντας το κόστος καυσίμου που καταναλώσαμε και τις θερμίδες που αποδόθηκαν σε κάθε χώρο του κτιρίου, μπορούμε με ακρίβεια να προσδιορίσουμε την κατανομή της δαπάνης που αναλογεί σε κάθε ένοικο.

Μερικά από τα πλεονεκτήματα της θερμιδομέτρησης είναι:

- Δίκαιη κατανομή δαπανών
- Οικονομία καυσίμου
- Αποφυγή διαπληκτισμών για κακόβουλες τροποποιήσεις του συστήματος θέρμανσης (όπως για παράδειγμα αλλαγή σωμάτων)
- Εύκολη εγκατάσταση και ρύθμιση

Ας σημειωθεί ότι το συγκεκριμένο σύστημα προβλέπεται από το Υπουργείο Περιβάλλοντος Ενέργειας και Κλιματικής Αλλαγής με τον Κανονισμό Ενεργειακής Απόδοσης Κτιρίων (Κ.ΕΝ.Α.Κ.) - Αθήνα 30 Μαρτίου 2010 Αριθ. πρωτ.: Δ6/Β/οικ.5825 - Φ.Ε.Κ. Αρ. Φύλλου 407 9 Απριλίου 2010, καθώς και από το Τεχνικό Επιμελητήριο Ελλάδος (ΤΕΕ) με το τελικό κείμενο αναθεώρησης της ΤΟΤΕΕ 2427/83 (Κατανομή Δαπανών Θέρμανσης Κτιρίων).

ΚΕΦΑΛΑΙΟ ΤΡΙΤΟ

Βελτιώσεις στα Συστήματα Θέρμανσης

Τα συστήματα θέρμανσης έχουν βελτιωθεί σε μεγάλο βαθμό τα τελευταία χρόνια και έχουν γίνει σημαντικές προσπάθειες για βελτίωση της ευχρηστίας και για προσθήκη επιπλέον λειτουργιών με τη χρήση μικροελεγκτών και άλλων ηλεκτρονικών συστημάτων.

Το πρώτο μεγάλο βήμα ήταν η μετατροπή του συστήματος θέρμανσης από κεντρικό σε διανεμημένο, ένα σύστημα δηλαδή το οποίο δε λειτουργεί ταυτόχρονα για όλα τα διαμερίσματα μιας πολυκατοικίας (ανάλογα με την κρίση κάποιου "διαχειριστή"). Σε αυτό το σύστημα, ο κάθε χρήστης έχει την επιλογή να "ανοίξει" ή να "κλείσει" τη θέρμανση στο διαμέρισμά του, ανάλογα με τις ανάγκες του.

Μια άλλη μεγάλη αλλαγή που επήλθε μέσα από την ανάγκη για οικονομία ήταν η αυτονομία σε περισσότερα επίπεδα. Ένας λέβητας που λειτουργεί παράγοντας θερμότητα για ένα κτίριο, λειτουργεί σε ένα συγκεκριμένο βαθμό με σκοπό να καλύψει τις ανάγκες για θέρμανση ολόκληρου του κτιρίου, είτε έχει ανάγκη το σύστημα θέρμανσης ένας χρήστης, είτε το έχουν όλοι. Η αυτονομία αρχικά αποσκοπούσε στο να περιορίσει το σύστημα θέρμανσης ανά χρήστη, ώστε να θερμαίνεται, για παράδειγμα, ένα διαμέρισμα μιας πολυκατοικίας αυτόνομα, χωρίς να υπερλειτουργεί ο λέβητας, κάτι που οδηγούσε σε υπερβολικές δαπάνες. Αυτονομία στη θέρμανση σήμερα όμως, μπορεί να σημαίνει και την πλήρη ανεξαρτητοποίηση ενός διαμερίσματος-οικίας, μέσω ενός συστήματος το οποίο να λειτουργεί αποκλειστικά για έναν χρήστη, ακόμα και χρησιμοποιώντας νέες μεθόδους θέρμανσης, καθώς και ανανεώσιμες πηγές ενέργειας.

Εν τέλει, ένα ακόμα πρόβλημα που τίθεται στις μέρες μας είναι η επίτευξη ενός ακόμα βαθμού αυτονομίας. Ένα διαμέρισμα έχει θερμαντικά σώματα σχεδόν σε όλα τα δωμάτια, με αποτέλεσμα το σύστημα να θερμαίνει ολόκληρο το διαμέρισμα, άσχετα από το αν σε μια συγκεκριμένη στιγμή δε χρησιμοποιούνται όλα τα δωμάτιά του. Το ζητούμενο τώρα πια είναι ο χρήστης να μπορεί να επιλέγει ποιά θερμαντικά σώματα θα λειτουργούν και τι θερμοκρασία θα παρέχουν στους χώρους ενός σπιτιού, οδηγώντας έτσι όχι μόνο στην μέγιστη δυνατή οικονομία θέρμανσης αλλά και στην μέγιστη άνεση διαβίωσης των ενοίκων του. Αυτό μπορεί να επιτευχθεί με την χρήση ενός κατανεμημένου συστήματος αυτονομίας θέρμανσης ελεγχόμενο από έναν έξυπνο θερμοστάτη όπως η λύση που προτείνουμε στην παρούσα διπλωματική εργασία.

3.1 Η Λύση

Αυτό που θέλουμε να πετύχουμε λοιπόν, είναι ένα βελτιωμένο σύστημα θέρμανσης, αυτοματοποιημένο αλλά και πλήρως ελέγξιμο από τον χρήστη, μέσω ενός σύγχρονου, "έξυπνου" θερμοστάτη αλλά και μέσω νέων τεχνολογικών μέσων που ήδη χρησιμοποιούνται, οδεύοντας σταθερά προς το concept του "έξυπνου" σπιτιού, που ρυθμίζεται αυτόματα ή ελέγχεται είτε τοπικά είτε απομακρυσμένα από τον χρήστη του. Αυτό το σύστημα μπορεί να περιλαμβάνει αυτά που αναφέρθηκαν παραπάνω, ανεξάρτητη ρύθμιση των θερμαντικών σωμάτων, θερμιδομέτρηση, ρύθμιση από απόσταση της λειτουργίας του λέβητα κλπ ώστε να επιτευχθεί οικονομικότερη λειτουργία του συστήματος θέρμανσης, με καλύτερες αποδόσεις και ελάχιστες απώλειες.

3.2 Ανάλυση

3.2.1 Το Θερμαντικό Πρόβλημα

Για την εγκατάσταση του συστήματος θέρμανσης γίνεται μια μελέτη, η οποία βασίζεται στον τρόπο που είναι κατασκευασμένο το διαμέρισμα, στα τετραγωνικά του κάθε δωματίου, στη θέση του κάθε δωματίου, στη θέση των παραθύρων και ούτω καθ'εξής. Με βάση αυτή τη μελέτη, ο μηχανικός αποφασίζει ποιά, πώς και πού θα εγκατασταθούν τα θερμαντικά σώματα στα διάφορα δωμάτια του διαμερίσματος. Στη συνέχεια, ο θερμοστάτης τοποθετείται σε έναν κεντρικό διάδρομο του σπιτιού, στον οποίο συνήθως δε βρίσκεται θερμαντικό σώμα. Ο θερμοστάτης ρυθμίζεται με βάση τη θερμοκρασία του διαδρόμου στον οποίο βρίσκεται και "ανοίγει" ή "κλείνει" την παροχή θερμού μέσου στα σώματα.

Η διαδικασία αυτή χρησιμοποιείται χρόνια τώρα αλλά έχει κάποια σοβαρά σφάλματα που προέρχονται από εμπειρικές θεωρήσεις στον τρόπο λειτουργίας της. Παραδείγματος χάριν γίνεται η εξής παραδοχή ότι λόγω της μετάδοσης της θερμότητας σε ολόκληρο το σπίτι και αφού ο διάδρομος δεν έχει θερμαντικό σώμα θεωρείται ότι δεν θα έχει υψηλότερη θερμοκρασία από τα υπόλοιπα δωμάτια του διαμερίσματος και επειδή βρίσκεται περιτριγυρισμένος από θερμότερα δωμάτια σιγά σιγά οι θερμοκρασίες θα εξισωθούν σε ολόκληρο το διαμέρισμα στην επιλεγμένη θερμοκρασία του. Δηλαδή ότι η θερμοκρασία του διαδρόμου θα είναι η ίδια ή τουλάχιστον αντιπροσωπευτική της θερμοκρασίας στα υπόλοιπα

δωμάτια.

Βέβαια, κάποια δωμάτια ίσως να μη χρειάζονται θέρμανση, εαν δε χρησιμοποιούνται ή δεν πρόκειται να χρησιμοποιηθούν. Άλλα δωμάτια εμφανίζονται να είναι πιο ζεστά ή πιο κρύα από την επιθυμητή για τον χρήστη θερμοκρασία. Τέλος αρκετά από τα δωμάτια διαχωρίζονται από το θερμοστάτη με κλειστές πόρτες. Όλα αυτά είναι στοχαστικές μεταβλητές, οι οποίες δε μπορούν να προβλεφθούν "μια κι έξω" από τον μηχανικό κατά την διάρκεια της μελέτης αλλά είναι και αρκετά δύσκολο να αντιμετωπιστούν σε πραγματικό χρόνο (real time) από τον ίδιο τον χρήστη. Ένας μηχανικός μπορεί να κάνει τις απαραίτητες ρυθμίσεις του συστήματος σε διακριτό χρόνο είτε κατά την διάρκεια της εγκατάστασης είτε της συντήρησης αλλά για τον χρήστη είναι πολύ δύσκολο να εξισορροπήσει πόσο μάλλον να ρυθμίσει τις θερμοκρασίες των ξεχωριστών δωματίων με την χρήση των διακοπτών (βάνες) των σωμάτων και με ένα μόνο κεντρικό σημείο μέτρησης (θερμοστάτη).

3.2.2 Η Μελέτη

Χρειαζόμαστε ένα σύστημα λοιπόν, που θα επιτρέπει στον χρήστη να ρυθμίσει τις θερμοκρασίες σε κάθε δωμάτιο ξεχωριστά, χωρίς να χρειάζεται τεχνογνωσία και χωρίς διαδικασία trial-and-error με τις βάνες των θερμαντικών σωμάτων.

Το σύστημα αυτό μπορεί να περιλαμβάνει έναν ηλεκτρονικό θερμοστάτη με επιλογείς θερμοκρασιών ανεξάρτητα για κάθε σώμα. Ο θερμοστάτης χώρου (ΘΧ) είναι απαραίτητο να περιλαμβάνει μια οθόνη για την εμφάνιση των επιλογών/ρυθμίσεων και των δεδομένων που χρειάζονται για αυτές, για τον έλεγχο δηλαδή των θερμοκρασιών χώρων απ'ευθείας από το ΘΧ.

Ένα ζήτημα όμως που ανακύπτει είναι το πώς θα είναι δυνατή η επικοινωνία του συστήματος θέρμανσης με το οικιακό δίκτυο, πώς θα γίνεται η ρύθμιση των θερμοκρασιών "από απόσταση", αλλά και με ποιόν τρόπο θα γίνεται η ρύθμιση αυτή!

3.3 Η αρχική ιδέα της κατασκευής και οι μετατροπές που χρειάστηκαν

Στο ξεκίνημα, η ιδέα ήταν η κατασκευή ενός συστήματος που να ελέγχει πλήρως το σύστημα θέρμανσης, από τον λέβητα μέχρι και τα θερμαντικά σώματα ανεξάρτητα. Ο Λέβητας θα ελεγχόταν και θα ρυθμιζόταν από τον θερμοστάτη ο οποίος θα είχε την επιπλέον δυνατότητα να ρυθμίζει την ροή του πετρελαίου στον λέβητα. Παράλληλα, ο έλεγχος της ροής του θερμαντικού μέσου στα σώματα θα γινόταν με ηλεκτρονικούς actuators, οι οποίοι θα ρυθμίζονταν επί τόπου από τους μικροελεγκτές των σωμάτων.

Η κατασκευή αυτή θα απαιτούσε μεταξύ άλλων τη χρήση ενός πραγματικού συστήματος θέρμανσης για δοκιμές πραγματικού χρόνου (και "χώρου"). Επιπλέον, ως κατασκευή, περιλάμβανε πολύ μεγάλο όγκο εργασίας σε ένα πολύ μεγάλο φάσμα εφαρμογών hardware και software. Γι αυτό και τελικά επικεντρωθήκαμε αποκλειστικά στον έξυπνο θερμοστάτη, αφαιρώντας το στάδιο ελέγχου του λέβητα.

Επιπλέον, ενώ είχε αρχικά προταθεί η χρήση wifi, επιλέξαμε να το αντικαταστήσουμε με ένα απλούστερο proprietary πρωτόκολλο της Nordic στα 2.4Ghz, για τρεις λόγους. Πρώτον, δεδομένου του ότι τα σώματα βρίσκονται στην περιοχή ενός διαμερίσματος, δεν είχαμε ανάγκη για κάλυψη πολύ μεγάλης εμβέλειας. Δεύτερον, τα δεδομένα που θα ανταλλάσσαμε θα ήταν απλά και δεν υπήρχε ανάγκη για ανταλλαγή δεδομένων σε κάποια γλώσσα του διαδικτύου, παρά μόνο τιμές θερμοκρασιών. Τρίτον και κυριότερο το hardware θα κόστιζε σημαντικά λιγότερο και θα έκανε την τελική καταστευή πιο προσιτή στην αγορά χωρίς να μειώνεται καθόλου η λειτουργικότης και οι δυνατότητές της.

Τέλος, η ιδέα του να χρησιμοποιηθεί ο Θ.Χ. ως ένας δικτυακός server για τον έλεγχο του θερμοστάτη μέσω δικτύου εγκαταλείφθηκε διότι θεωρήθηκε πιο αποδοτικός ο έλεγχος του Θ.Χ. μέσω μιας εφαρμογής που θα τρέχει σε κάποιο Η/Υ που υπάρχει πλέον διαθέσιμος σε κάθε οικία και θα μπορεί να είναι άμεσα προσβάσιμος με την χρήση κάποιας εφαρμογής remote desktop.

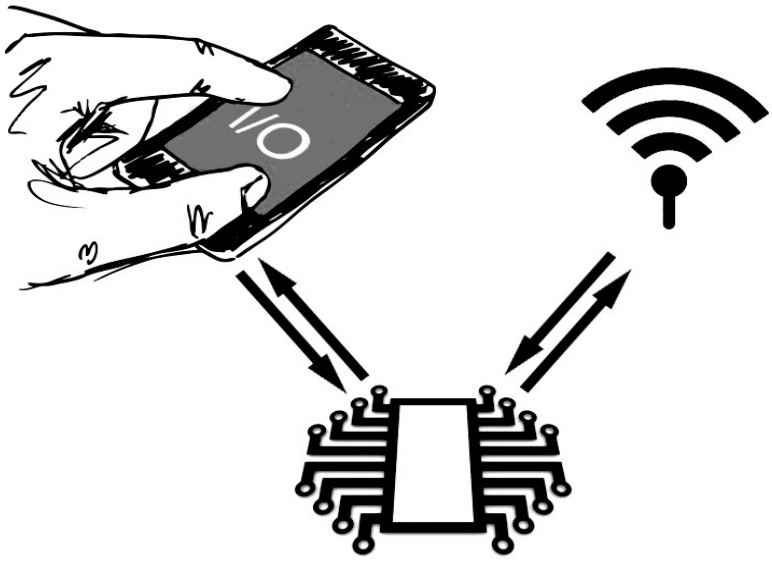
3.4 Σύνοψη

Συνοψίζοντας, με την κατασκευή αυτή επιχειρούμε μέσω ενός "έξυπνου" θερμοστάτη να ελέγχουμε τις θερμοκρασίες των θερμαντικών σωμάτων και κατ'επέκταση των χώρων μιας εγκατάστασης. Αυτό μπορεί να γίνεται με δύο τρόπους. Πρώτον μέσω του Θ.Χ.

χρησιμοποιώντας μια οθόνη TOUCH PANEL στην οποία θα παρουσιάζονται οι πληροφορίες για τον έλεγχο των θερμοκρασιών των θερμαντικών σωμάτων και θα ελέγχονται μέσω της λειτουγρίας αφής. Δεύτερον, μέσω ενός Η/Υ και μιας συσκευής (interface) που να παρέχει πρόσβαση μέσω RF στον Θ.Χ..

Η κατασκευή λοιπόν διαμορφώθηκε ως εξής: Ένας ηλεκτρονικός θερμοστάτης, ο οποίος αποτελείται από έναν μικροελεγκτή με οθόνη LCD και δυνατότητα touch, αντί για πληκτρολόγιο, επικοινωνεί μέσω RF με δυο διαφορετικά συστήματα. Το ένα σύστημα είναι ο μηχανισμός (actuator) ελέγχου ροής θερμαντικού μέσου στα καλοριφέρ (αντιστοιχεί ένας μηχανισμός σε κάθε σώμα) και το άλλο είναι ο μηχανισμός (interface) που παρεμβάλλεται ανάμεσα στο Θ.Χ. και έναν Η/Υ και επιτρέπει τον έλεγχο του μέσω του Η/Υ.

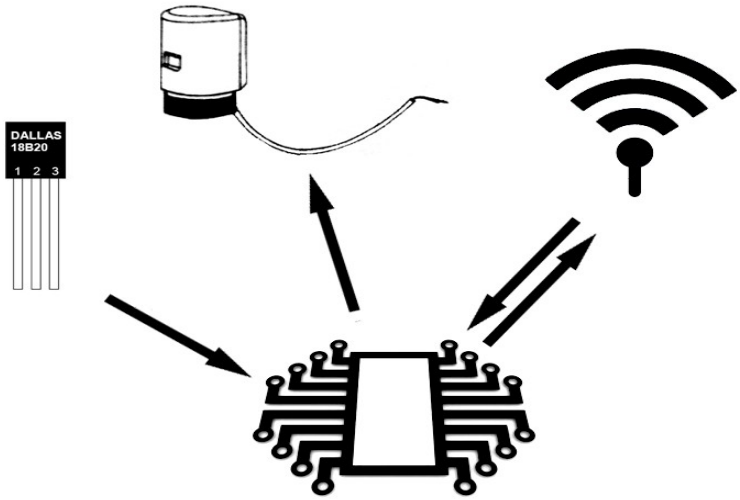
Η λειτουργία του θερμοστάτη περιλαμβάνει την προβολή των τρεχουσών θερμοκρασιών των δωματίων στην οθόνη και την επιλογή της επιθυμητής θερμοκρασίας για κάθε δωμάτιο ξεχωριστά. Γι αυτό το λόγο επιλέχθηκε μια οθόνη LCD με δυνατότητα αφής ώστε να συνδυάσουμε Input και Output δεδομένων σε μια "περιφερειακή" συσκευή του μικροελεγκτή που να δίνει την μέγιστη ευκολία στον χρήστη.



Εικόνα 4.1: Σχεδιασμός του θερμοστάτη

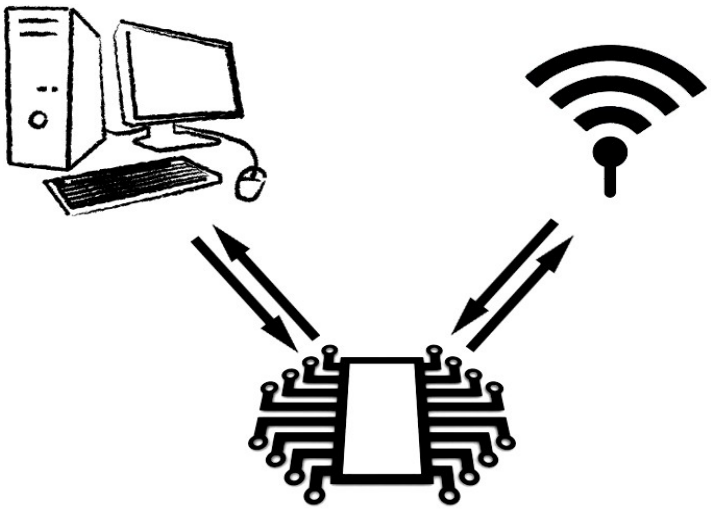
Ο μηχανισμός που τοποθετείται στα δωμάτια περιλαμβάνει έναν μικροελεγκτή, ένα ηλεκτρονικό θερμόμετρο, ώστε να μετρά σε συνεχή χρόνο τη θερμοκρασία του εκάστοτε δωματίου, καθώς και έναν actuator που μεταβάλλει τη ροή νερού στο θερμαντικό σώμα. Ο

μικροελεγκτής συγκρίνει τη θερμοκρασία που λαμβάνει μέσω του θερμομέτρου με τη θερμοκρασία που του αποστέλλει ο θερμοστάτης και αποκρίνεται ανάλογα, στέλνοντας εντολή στον actuator.



Εικόνα 4.2: Σχεδιασμός των μηχανισμών των σωμάτων

Ο μηχανισμός που συνδέεται στον Η/Υ αποτελείται από έναν μικροελεγκτή, ο οποίος συνδέεται σε κάποια USB θύρα του Η/Υ και ανταλάσσει σειριακά δεδομένα με αυτόν. Η λειτουργία του περιλαμβάνει την αποστολή των θερμοκρασιών του δικτύου μας καθώς και τη μεταφορά των θερμοκρασιών που θέτει ο χρήστης μέσω του Η/Υ στον θερμοστάτη.



Εικόνα 4.3: Σχεδιασμός του μηχανισμού στην πλευρά του Η/Υ

5.1 Εισαγωγή

Η κατασκευή απαιτεί τέσσερεις συσκευές, μια για το θερμοστάτη, μια για τον Η/Υ και δυο (όμοιες) για τα συστήματα ελέγχου των σωμάτων. Οι πλακέτες που κατασκευάστηκαν ήταν αναλυτικά οι παρακάτω.

Ο θερμοστάτης: Ο θερμοστάτης αποτελείται από έναν μικροελεγκτή ATMega1284p. Πάνω στον μικροελεγκτή συνδέονται δυο περιφερειακά, η LCD οθόνη και το RF Module. Η δυνατότητα touch της οθόνης και το RF module χρησιμοποιούν και τα δυο το Serial Peripheral Interface του μικροελεγκτή. Για να μπορέσουν να συνδεθούν αυτά τα δυο περιφερειακά πάνω στον μικροελεγκτή, κατασκευάστηκε ένα custom shield. Κατά τη διάρκεια κατασκευής του θερμοστάτη κατασκευάστηκε επίσης μια πλακέτα η οποία περιλαμβάνει μόνο ένα RF module και χρησιμοποιήθηκε για το αρχικό στήσιμο του RF δικτύου.

Οι διακόπτες των σωμάτων: Η κατασκευή του περιλαμβάνει έναν ATMega328p. Στην πλακέτα που τοποθετήθηκε ο μικροελεγκτής, υπάρχουν τα εξής: ένας voltage regulator, ο οποίος ρίχνει την τροφοδοσία των 9V στα 5V, ένα ηλεκτρονικό θερμόμετρο, μια ομάδα led και το RF module.

Η επικοινωνία με τον Η/Υ: Η κατασκευή της πλακέτας περιλαμβάνει έναν μικροελεγκτή ATMega328p. Το μόνο άλλο που υπάρχει πάνω στην πλακέτα είναι ένα RF module.

5.2 Τα υλικά που χρησιμοποιήθηκαν

5.2.1 Μικροελεγκτές σε Arduino boards

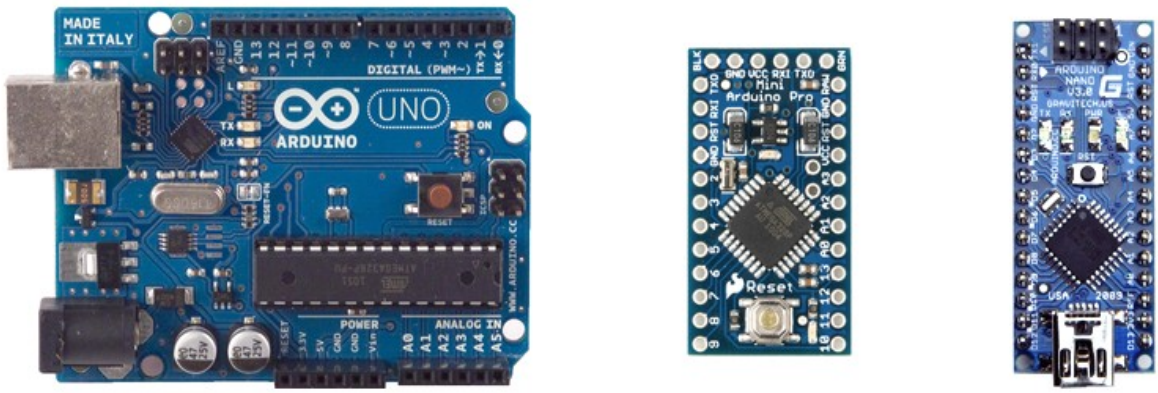
Το Arduino είναι μια πλατφόρμα πρωτότυπων ηλεκτρονικών κατασκευών ανοικτού κώδικα, που βασίζονται σε ευέλικτο και εύκολο στη χρήση hardware και software. Οι

πλακέτες που παρέχει χρησιμοποιούν μικροελεγκτές με τη δυνατότητα σύνδεσης περιφερειακών και το πρόγραμμα με το οποίο γράφονται οι κώδικες στη μνήμη των μικροελεγκτών είναι εύχρηστο και ελαφρύ, βασισμένο στη γλώσσα C++.

Οι πλακέτες που παρέχουν ποικίλουν σε μεγέθη και δυνατότητες. Στην εργασία χρησιμοποιήσαμε μια εκ των βασικότερων πλακετών που έχουν κατασκευάσει, με το πρότυπο όνομα UNO, η οποία λειτουργεί με έναν μC ATMega328p. Στην πορεία χρειάστηκαν επιπλέον πλακέτες, οι οποίες να είναι απλούστερες και να μη χρειάζεται να φέρουν εις πέρας μεγάλο φόρτο εργασίας. Για τον έλεγχο των actuators επιλέχθηκαν οι πλακέτες Arduino Pro Mini, οι οποίες είναι μικρότερες και περιλαμβάνουν τον ίδιο τύπο μικροελεγκτή με την πλακέτα UNO, δεν περιλαμβάνουν όμως USB interface. Τέλος, όταν εμφανίστηκε η ανάγκη για έναν ακόμα μικροελεγκτή, επιλέχθηκε η πλακέτα Arduino Nano, η οποία είναι σχεδόν ίδια με την Mini, αλλά έχει ενσωματωμένη USB θύρα, η οποία άλλωστε θα μας χρειαζόταν, καθώς η τελευταία αυτή πλακέτα πρέπει να συνδέεται στον Η/Υ μας.

Το πρόγραμμα του θερμοστάτη ήταν αρκετά μεγάλο και αγοράστηκε και συναρμολογήθηκε μια custom πλακέτα, η οποία είναι βασισμένη στις προδιαγραφές του UNO, αλλά χρησιμοποιεί μικροελεγκτή με μεγαλύτερη flash μνήμη. Τα χαρακτηριστικά της και η κατασκευή της θα αναλυθούν σε επόμενη ενότητα.

Στην παρακάτω εικόνα φαίνονται τρεις από τις προαναφερθείσες πλακέτες. Όλες μοιράζονται τα ίδια χαρακτηριστικά, έχουν τον ίδιο μC και παρέχουν 6 αναλογικές θύρες I/O και 13 ψηφιακές. Επιπλέον παρέχουν Serial Peripheral Interface για την ταυτόχρονη σύνδεση περιφερειακών.

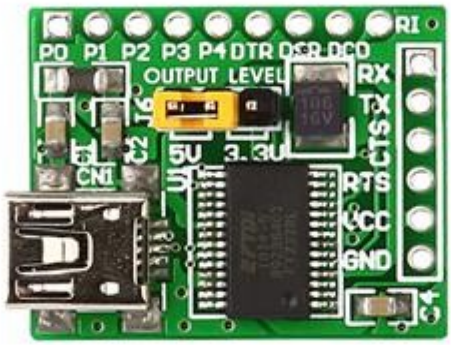


Εικόνα 5.1: Από αριστερά προς τα δεξιά: Arduino UNO, Mini και Nano

5.2.1.1 Ο προγραμματισμός των μικροελεγκτών

Αρκετές από τις έτοιμες πλακέτες με μικροελεγκτές που κυκλοφορούν στην αγορά έχουν ενσωματωμένο USB interface, το οποίο χρησιμοποιείται για την επικοινωνία των μικροελεγκτών με PC αλλά και για τον προγραμματισμό τους. Το interface αυτό βασίζεται στο FTDI chip. Οι FTDI μονάδες κατασκευάζονται από την Σκωτσέζικη εταιρεία Future Technology Devices International. Αυτό που κάνουν είναι να μετατρέπουν σειριακά σήματα σε σήματα USB.

Οι πλακέτες με μικροελεγκτές που κυκλοφορούν στην αγορά και έχουν ενσωματωμένη θύρα USB έχουν και ενσωματωμένο κάποιου είδους FTDI chip. Οι πλακέτες Arduino νέας γενιάς έχουν αντικαταστήσει το FTDI chip με έναν ATMega8U2, ο οποίος παρ'όλα αυτά επιτελεί την ίδια λειτουργία. Όπως και να'χει, στους σειριακούς ακροδέκτες τους έχει επικρατήσει το όνομα FTDI pins και έτσι θα τους αποκαλούμε. Οι πλακέτες UNO και Nano περιλαμβάνουν USB interface για τον προγραμματισμό τους. Αντίθετα, η πλακέτα Mini δεν έχει θύρα USB, αλλά έχει 6 FTDI pins, πάνω στα οποία συνδέεται ένας FTDI programmer. Το ίδιο ισχύει και για την πλακέτα με τον ATMega1284p. Για την αποφυγή περαιτέρω εξόδων, είναι δυνατό να χρησιμοποιηθεί η πλακέτα UNO ως FTDI προγραμματιστής, καθώς έχει ενσωματωμένο το FTDI chip.

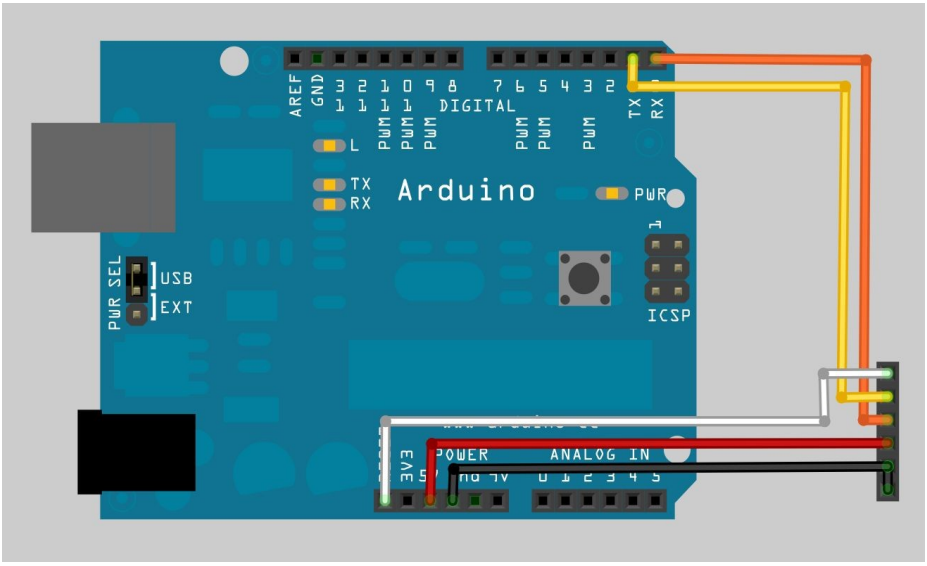


Εικόνα 5.2: FTDI programmer από την εταιρεία mikroElectronika

Με λίγη έρευνα στο διαδίκτυο βρέθηκαν αρκετά blogs που παρείχαν πληροφορίες για το πώς γίνεται ο προγραμματισμός ενός μικροελεγκτή χρησιμοποιώντας το chip που περιλαμβάνεται στην πλακέτα UNO. Στο site instructables.com βρέθηκε ένας αναλυτικός οδηγός που εξηγεί πώς μεταδίδονται τα δεδομένα κατά τη διάρκεια του προγραμματισμού ενός μικροελεγκτή και ποιες θύρες του μικροελεγκτή χρησιμοποιούνται.

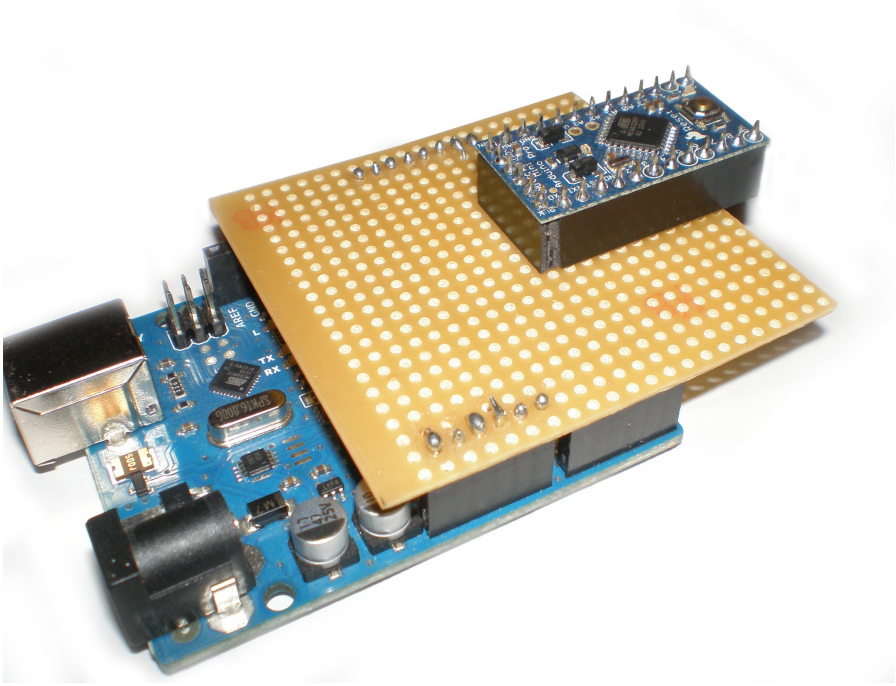
Οι FTDI προγραμματιστές χρησιμοποιούν 6 pins για τη μεταφορά των δεδομένων στη σειριακή πλευρά (όχι στη USB δηλαδή). Οι 6 αυτοί ακροδέκτες είναι με τη σειρά οι εξής: DTR (GRN), TX0, RX1, VCC, CTS, GND (BLK). Οι λέξεις BLK και GRN αντιστοιχούν στα χρώματα black και green, που έχουν τα ακραία καλώδια και αποτελούν οδηγό για το πώς συνδέεται στις θύρες ένα FTDI καλώδιο. Ο ακροδέκτης DTR χρησιμοποιείται για την επιανεκκίνηση του μC μετά τη φόρτωση του εκάστοτε προγράμματος και συνδέεται στη θύρα RST του μικροελεγκτή. Οι CTS και BLK είναι και οι δυο γειωμένοι. Οι ακροδέκτες TX και RX συνδέονται στις αντίστοιχες θύρες TX και RX του μC. Τέλος, ο ακροδέκτης VCC συνδέεται στη θύρα τροφοδοσίας 5V. Στην ουσία αυτό που κάναμε ήταν να συνδέσουμε τον μC που θέλουμε να προγραμματίσουμε στο κύκλωμα FTDI της πλακέτας UNO και να πάρουμε μέσω αυτής της μονάδας τα σήματα για να προγραμματίσουμε τον μC μας.

Παρατηρώντας στην εικόνα 5.1 τον Arduino Mini, θα δούμε ότι στην πάνω πλευρά του έχει 6 pins, των οποίων τα ονόματα δηλώνουν ότι είναι τα FTDI pins μέσω των οποίων προγραμματίζεται. Στην παρακάτω εικόνα 5.3, φαίνεται σχηματικά η σύνδεση των ακροδεκτών του Arduino UNO με τους FTDI ακροδέκτες σε οποιαδήποτε πλακέτα δεν έχει integrated έναν προγραμματιστή. Με αυτόν τον οδηγό κατασκευάστηκε μια πλακέτα για τον προγραμματισμό των Arduino Mini, και η οποία χρησιμοποιήθηκε και για τον προγραμματισμό της τελευταίας custom πλακέτας που κατασκευάστηκε για την εργασία.



Εικόνα 5.3: Σχεδιασμός ενός FTDI programmer με βάση το chip του UNO

Η πλακέτα που κατασκευάστηκε έχει τη μορφή shield για Arduino UNO, "κουμπώνει" δηλαδή πάνω στον UNO και στο κέντρο της βρίσκονται οι 6 ακροδέκτες πάνω στους οποίους συνδέεται ο Arduino Mini.

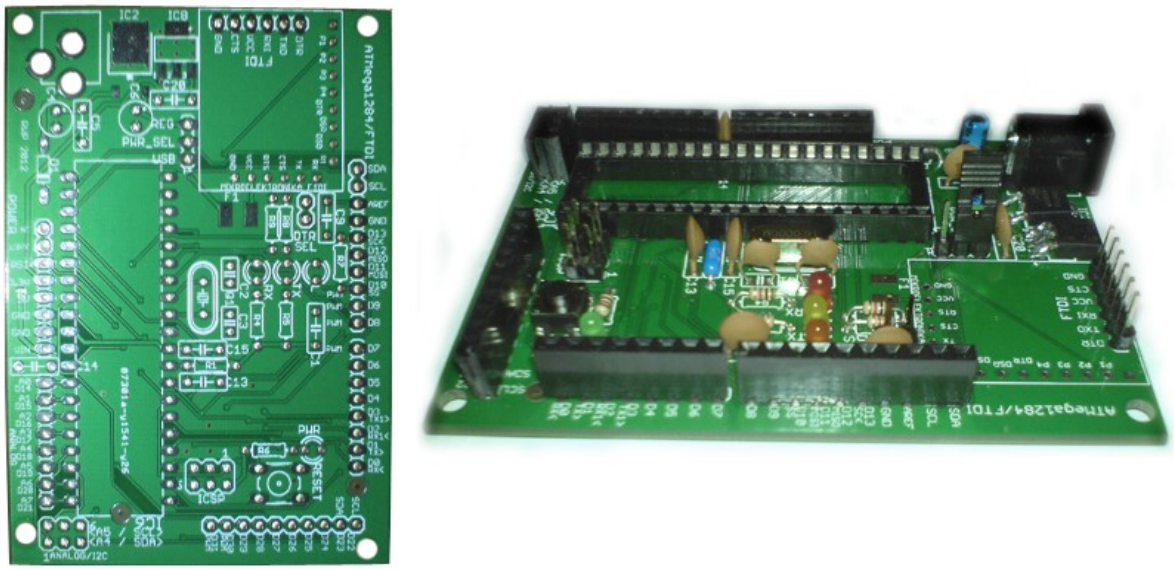


Εικόνα 5.4 Η πλακέτα του προγραμματιστή FTDI που κατασκευάστηκε

5.2.1.2 Η πλακέτα του ATMega1284P

Μετά την έκρηξη που έκαναν οι πλακέτες Arduino, και χάρη στο γεγονός ότι ήταν κατασκευές ανοιχτού κώδικα, εμφανίστηκαν πολλές άλλες κατασκευές βασισμένες στην ιδεολογία των πλακετών Arduino. Στην αναζήτηση για μια πλακέτα βασισμένη σε μικροελεγκτή με μεγαλύτερη flash βρήκα την πλακέτα "Bobuino", η οποία είναι κατασκευαστικά συμβατή με τα shields του Arduino UNO. Η πλακέτα αγοράστηκε σκέτη, μόνο το διάτρητο PCB, για $6 και τα υλικά κόστισαν 4 ευρώ, εξαιρουμένου του μικροελεγκτή. Η κατασκευή διήρκεσε περίπου 2 ώρες.

Η πλακέτα είναι σε μεγάλο ποσοστό ίδια με τον Arduino UNO. Οι λειτουργικες διαφορές της, με εξαίρεση τον διαφορετικό μικροελεγκτή, είναι ότι προσφέρει 10 περισσότερα digital I/O pins.



Εικόνα 5.5: Η πλακέτα Bobuino

Όπως φαίνεται κάτω δεξιά, το σημείο στο οποίο θα τοποθετούνταν ο προγραμματιστής MIKROE483 (εικόνα 5.2) έχει μείνει κενό, με εξαίρεση τα 6 αρσενικά pins που έχουν κολληθεί στους FTDI ακροδέκτες της πλακέτας. Αυτά τα 6 pins θα συνδεθούν με την πλακέτα που κατασκευάσαμε για να προγραμματίζουμε τα Arduino Mini, για να προγραμματίσουμε και τον μικροελεγκτή αυτής της πλακέτας, μέσω ενός καλωδίου FTDI!

5.2.1.3 To Serial Peripheral Interface (SPI)

Το SPI είναι ένα σύγχρονο serial data link κατασκευασμένο από τη Motorola, το οποίο λειτουργεί σε full duplex mode. Επιτρέπει την παράλληλη σύνδεση συσκευών σε αυτό και την παράλληλη λειτουργία τους. Μερικές φορές αποκαλείται four-wire serial bus. Λειτουργεί με μια μόνο master συσκευή, αλλά με πολλαπλούς slaves. Όπως φαίνεται, το interface αυτό χρησιμοποιεί τέσσερα καλώδια-ακροδέκτες, τέσσερα σήματα δηλαδή. Τα σήματα αυτά είναι τα εξής:

SCK: Serial Clock (αλλιώς CLK ή SCLK)
MOSI: Master Output Slave Input
MISO: Master Input Slave Output
SS: Slave Select (αλλιώς CS, CSN)

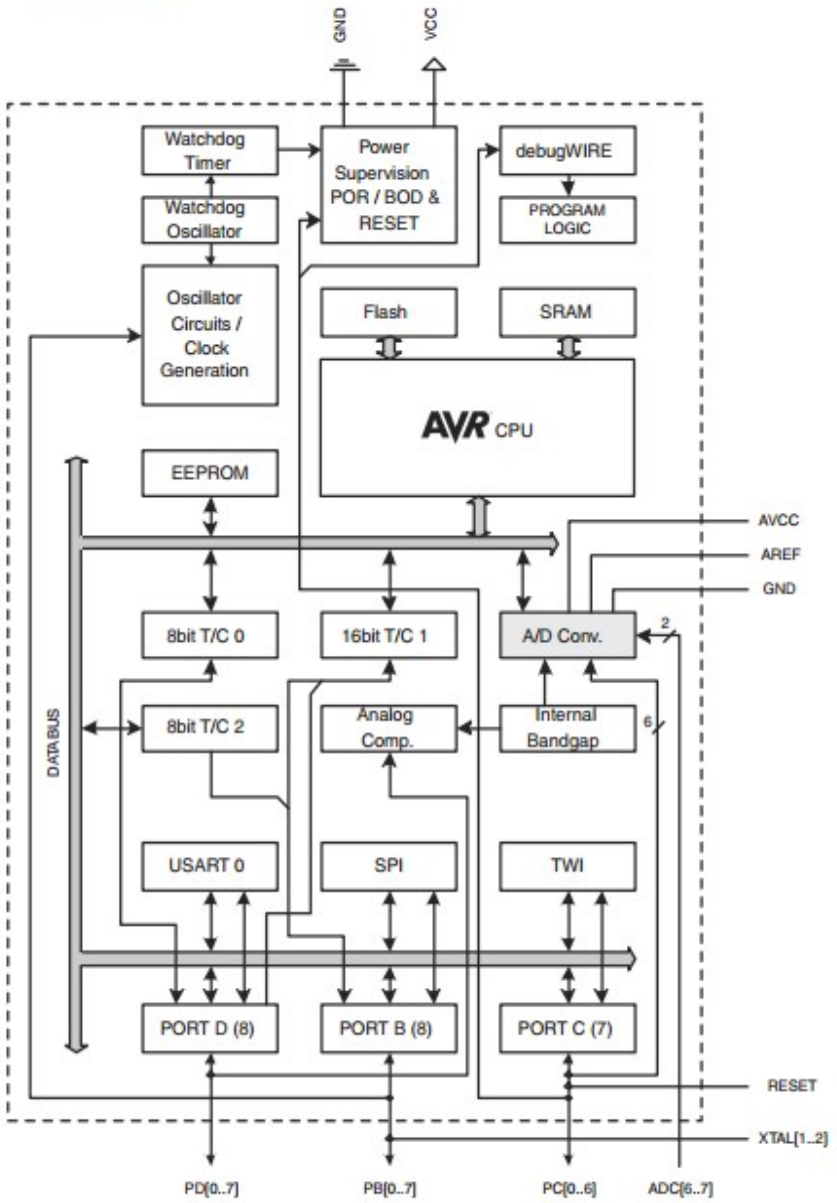
Τα σήματα MOSI και MISO χρησιμοποιούνται για την επικοινωνία μεταξύ του master και του/των slave/slaves. Το σήμα SS είναι που έχει ιδιαίτερη σημασία, καθώς αποτελεί τον ακροδέκτη που ενεργοποιεί και απενεργοποιεί τη συσκευή. Παρ'όλο που οι πλακέτες Arduino (και το documentation του SPI) παρέχουν μια θύρα SS, είναι δυνατόν οποιαδήποτε θύρα να χρησιμοποιηθεί για τον έλεγχο του κάθε Slave. Στις πλακέτες που χρησιμοποιήσαμε, οι προκαθορισμένες θύρες του SPI είναι οι: 11 (MISO), 12 (MOSI), 13 (SCK) και 10 (SS).

Όταν έχουμε συνδέσει έναν slave στην πλακέτα, το pin 10 χρησιμοποιείται (active low) ως Slave Select του συγκεκριμένου Slave. Σε περίπτωση που προσθέσουμε επιπλέον Slaves, πρέπει να ορίσουμε εμείς, και να συνδέσουμε στο αντίστοιχο pin, ποιο pin της πλακέτας θα χρησιμοποιείται για τον έλεγχο του δεύτερου Slave.

5.2.1.4 Η διασύνδεση hardware με software στους μικροελεγκτές Atmel

Οι μικροελεγκτές της Atmel περιλαμβάνουν ports για τις λειτουργίες I/O. Ο ATMega328p χρησιμοποιεί τρια ports (B,C,D), ενώ ο ATMega1284p χρησιμοποιεί τέσσερα ports (A,B,C,D). Κάθε port αποτελείται από μέχρι 8 bits, 8 ακροδέκτες δηλαδή. Με μια ματιά στα block diagrams των δυο μικροελεγκτών βλέπουμε ότι για παράδειγμα στον ATMega328p οι θύρες B και D χρησιμοποιούν 8 bits, ενώ η θύρα C χρησιμοποιεί 7 bits. Κάθε bit αντιστοιχεί και σε ένα pin του μικροελεγκτή.

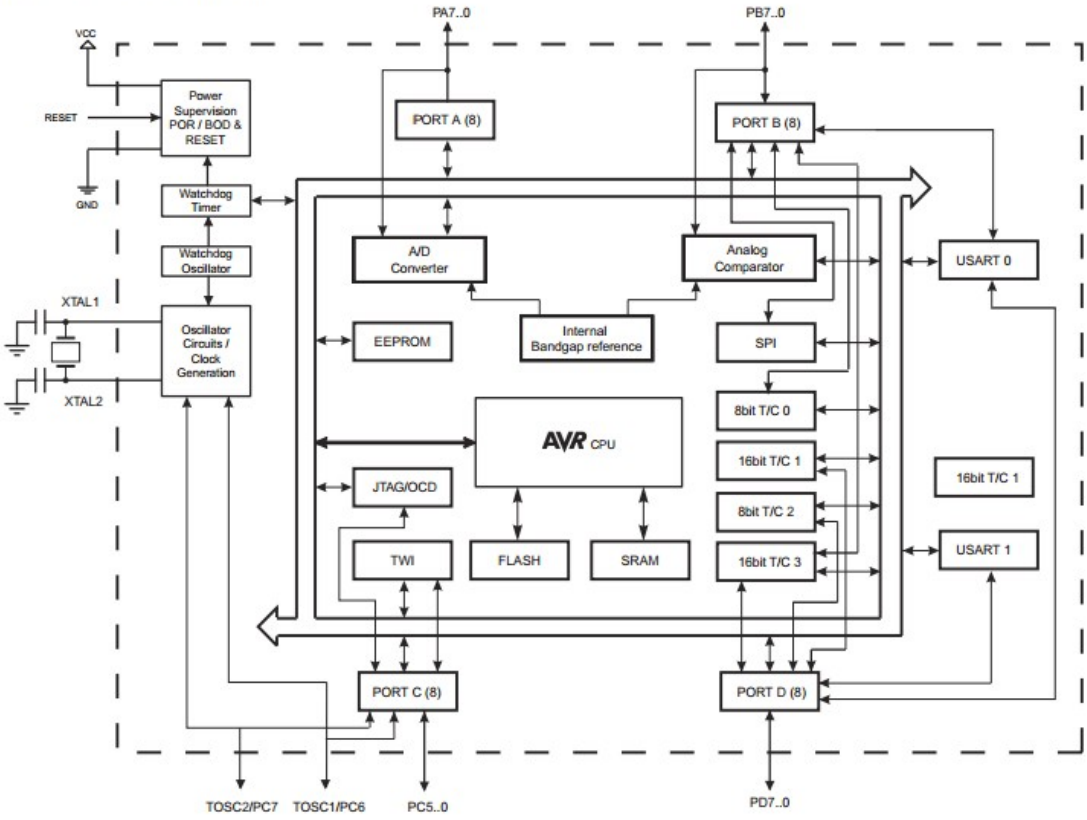
Figure 2-1. Block Diagram



Εικόνα 5.6: Το block diagram του μικροελεγκτή ATMega328p

Ο ATMega1284p χρησιμοποιεί, όπως είπαμε, τέσσερις θύρες, όλες με 8 I/O pins. Σημασία έχει λοιπόν πώς αντιστοιχίζονται αυτά τα pins των θυρών στα λειτουργικά pins του board, που χρησιμοποιούμε και καλούμε εμείς μέσα από τον κώδικα. Ακόμα μεγαλύτερη σημασία έχει το γεγονός ότι ο κώδικάς μας μπορεί να καλέσει αυτά τα σήματα και με τον αριθμό των pin τους, αλλά και με το γράμμα της θύρας τους. Αυτό αναφέρεται, και είναι πολύ σημαντικό, γιατί αποτέλεσε ένα από τα μεγαλύτερα προβλήματα που εμφανίστηκαν.

Figure 2-1. Block Diagram



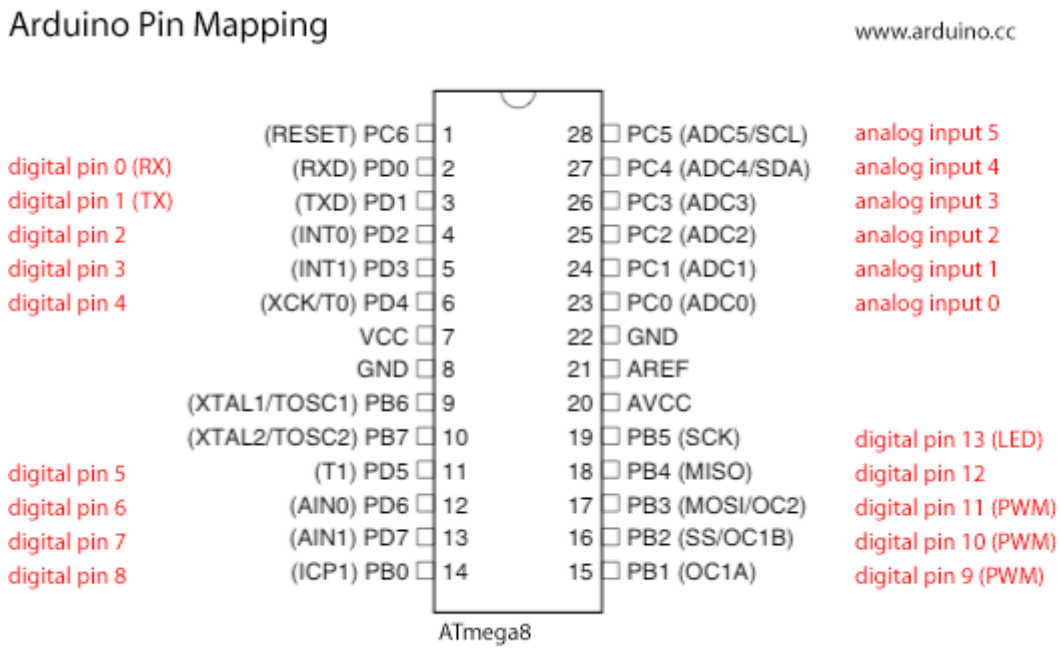
Εικόνα 5.7: Το block diagram του μικροελεγκτή ATMEga1284p

Ο συσχετισμός θυρών και pins στο Arduino γίνεται μέσω ενός αρχείου header που ονομάζεται pins_arduino και είναι μοναδικό για κάθε διαφορετική πλακέτα, καθώς αντιστοιχίζει σε κάθε digital I/O pin της πλακέτας έναν ακροδέκτη του μικροελεγκτή, ο οποίος και ανήκει σε μια θύρα.

Το πρόβλημα που εμφανίστηκε λοιπόν είναι ότι η νεα πλακέτα δεν ήταν πλήρως συμβατή με την πλακέτα UNO, καθώς τα digital pins των πλακετών είχαν οριστεί να είναι στην ίδια θέση, αλλά προέρχονταν από διαφορετικές θύρες των μC. Αυτό το πρόβλημα δεν είναι εμφανές όταν κάποιος τρέξει ένα απλό πρόγραμμα, το οποίο αναφέρεται στα pins με τα ονόματα που έχουν καθοριστεί στο αρχείο pins_arduino. Υπάρχουν όμως και libraries οι οποίες κάνουν χρήση των pins με τα ονόματα των θυρών τους. Για την ακρίβεια, κάνουν χρήση των θυρών, κυρίως εκεί που χρειάζεται 8-bit ή 16-bit επικοινωνία. Αν κάποιος θελήσει να μεταφέρει 8 bit παράλληλα, απλά θα αναφερθεί στην εκάστοτε θύρα και τα bits αυτά θα χρησιμοποιήσουν τα 8 pins της θύρας για να μεταδοθούν.

Η ασυμβατότητα των δυο πλακετών λοιπόν εμφανίστηκε κατά τη χρήση της βιβλιοθήκης UTFT για την οθόνη LCD. Η οθόνη χρησιμοποιεί 8bit data bus για τη μετάδοση

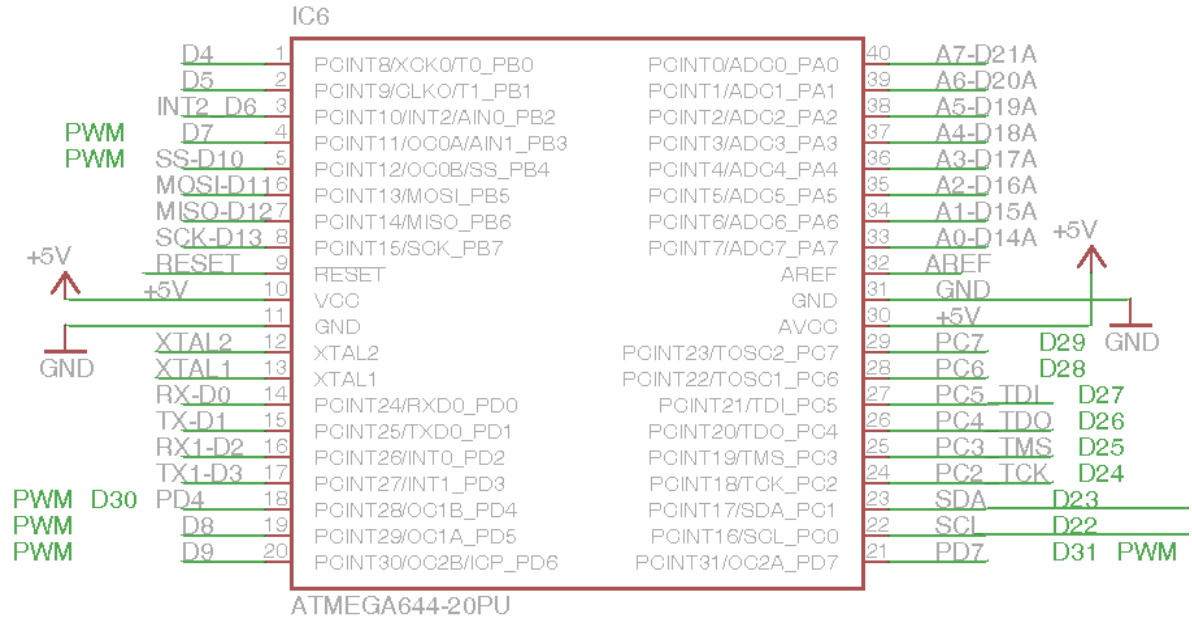
των δεδομένων. Αυτό το data bus καθορίζεται διαφορετικά για διαφορετικούς μικροελεγκτές, σεβόμενο πάντα την εσωτερική τους κατασκευή. Για παράδειγμα, στο UTFT ορίζεται ως data bus για τον ATMega328p η θύρα D, η οποία και έχει αντιστοιχηθεί στα digital pins D0-7 στο αρχείο pins_arduino για αυτόν τον μικροελεγκτή. Παρακάτω φαίνεται το pin mapping του ATMega328p στον Arduino UNO. Να σημειωθεί ότι οι ακροδέκτες αναφέρονται ως Pxn όπου x το γράμμα της θύρας και n ο αριθμός του bit της θύρας. Για παράδειγμα τα pins της θύρας D αναφέρονται ως PD0-PD7.



Εικόνα 5.8: Το Pin mapping του ATMega328p στον Arduino UNO

Για τον ATMega1280, παρόλο που από το UTFT χρησιμοποιείται ως data bus η θύρα A, υπάρχει η δυνατότητα να μεταφερθούν οι έξοδοι σε άλλες θύρες, ώστε να είναι δυνατό να συνδεθεί σε αυτόν τον μC το συγκεκριμένο περιφερειακό. Στην πλακέτα που χρησιμοποιήσαμε με τον ATMega1284p, παρόλο που τα digital pins της πλακέτας D0-3 αντιστοιχούν στα port pins PD0-3, τα digital pins D4-7 συνδέονται στα port pins PB0-3, αντί για τα PD4-7 όπως στον ATMega328p. Με αυτόν τον τρόπο δε μπορούσε να επικοινωνήσει ο μικροελεγκτής με την οθόνη, καθώς μέσα στη βιβλιοθήκη καλούνταν by default τα 8bits της θύρας D.

Για να γίνει πιο ξεκάθαρη η κατάσταση, παρατίθεται και το pin mapping του ATMega1284P παρακάτω:



Εικόνα 5.9: Το Pin mapping του ATMega1284p στον Bobuino

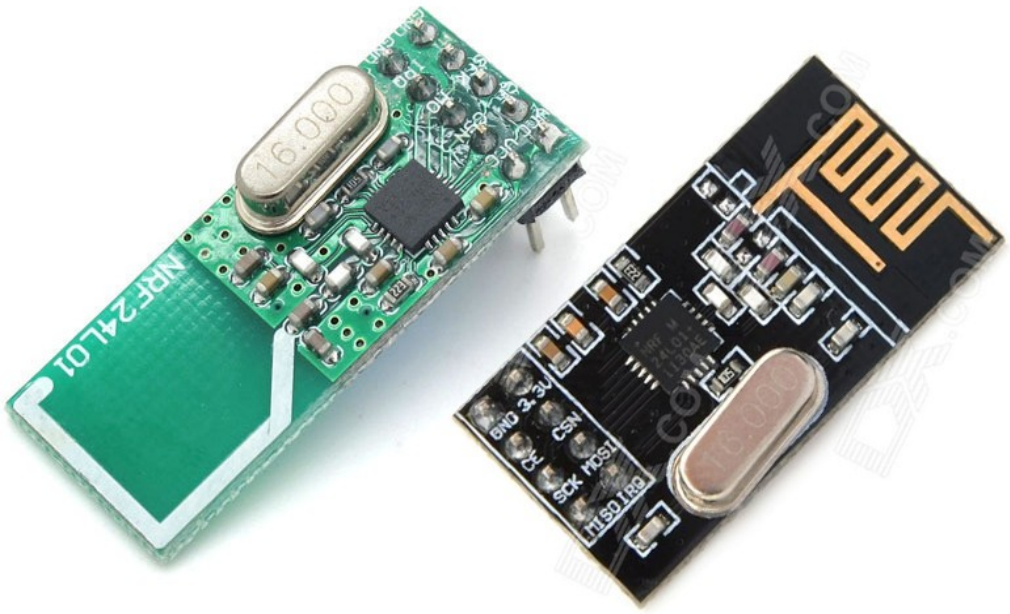
Όπως βλέπουμε η αντιστοίχιση των θυρών είναι διαφορετική από αυτή στον Arduino Uno. Μια προσπάθεια να διορθωθεί το πρόβλημα αυτό έγινε μέσα στη library UTFT, χρησιμοποιώντας πράξεις bitwise logic και μεταφέροντας την DATA έξοδο του προγράμματος σε διαφορετικές θύρες. Αυτό όμως επηρέαζε σε μεγάλο βαθμό τα pins των θυρών που χρησιμοποιούνταν από άλλες λειτουργίες. Για την ακρίβεια, παίρναμε τα δεδομένα, τα οποία αποθηκεύονταν σε registers με όνομα VH και VL και τα αποθηκεύαμε στα τέσσερα πρώτα bits της θύρας D και στα τέσσερα πρώτα bits της θύρας B. Τα τέσσερα επόμενα bits της θύρας B όμως, χρησιμοποιούνται από το Serial Peripheral Interface και οι πράξεις προκαλούσαν δυσλειτουργία σε αυτό, καθιστώντας την προγραμματιστική λύση αδύνατη.

Η λύση που ήταν δυνατή ήταν μόνο η λύση μέσω hardware. Για να γίνει αυτό χρειάστηκαν μικροεπεμβάσεις στην κατασκευή τόσο της οθόνης UTFT όσο και της πλακέτας που παρεμβαλλόταν ανάμεσα στην οθόνη και την πλακέτα του ATMega1284P. Αυτές οι επεμβάσεις θα αναλυθούν σε επόμενη ενότητα.

5.2.2 RF Modules NRF24L01 & NRF24L01+

Τα RF modules της εταιρείας Nordic semiconductors είναι RF συσκευές χαμηλής κατανάλωσης και μικρής εμβέλειας (περίπου 60 μέτρα). Συνδυάζουν RF πομποδέκτη, RF

synthesizer και λειτουργούν στα 2.4 GHz. Υποστηρίζουν διασύνδεση SPI υψηλής ταχύτητας μετάδοσης και αποτελούν ένα πολύ αποδοτικό μέσο ασύρματης επικοινωνίας μεταξύ των μικροελεγκτών που χρησιμοποιήσαμε. Τα πιο σημαντικά χαρακτηριστικά τους είναι η χαμηλή τους τιμή και η υψηλή τους απόδοση. Παρ'όλο που είναι αρκετά δύσκολα στον χειρισμό στην αρχή, η εξοικείωση με τις λειτουργίες τους είναι εύκολη. Για την κατασκευή αποκτήσαμε δυο διαφορετικά είδη modules, τα NRF24L01 και NRF24L01+. Εκτός από τη διαφορετική θέση των pins τάσης και γείωσης, τα modules αυτά λειτουργούν με τον ίδιο τρόπο.



Εικόνα 5.10: Τα modules NRF24L01 και L01+

5.2.2.1 Επικοινωνία με τον μικροελεγκτή

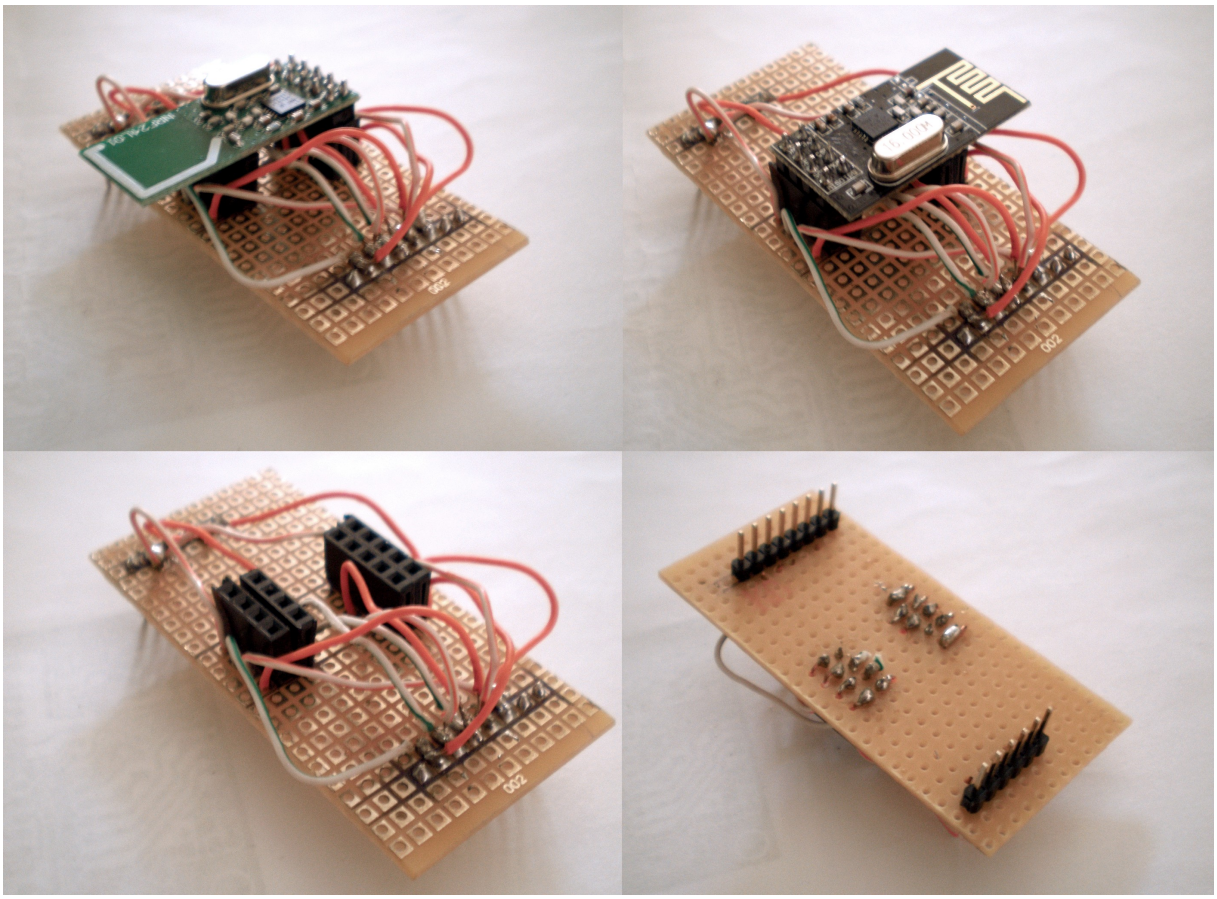
Τα modules αυτά έχουν 8 ακροδέκτες. Το NRF24L01 έχει 10, αλλά, όπως φαίνεται στην παραπάνω εικόνα, έχει διπλά ground και Vcc pins. Οι υπόλοιποι ακροδέκτες είναι οι CE, IRQ και οι τέσσερεις SPI ακροδεκτες (SCK, CSN, MISO, MOSI).

Ο ακροδέκτης CE χρησιμοποιείται για να ελέγχει τη λήψη και αποστολή δεδομένων όταν το module βρίσκεται σε TX ή RX mode (transmit-receive mode). Ο ακροδέκτης IRQ χρησιμοποιείται για interrupts. Οι τέσσερεις SPI ακροδέκτες είναι οι: SCK, MISO, MOSI και CSN.

Η σύνδεση που προτείνεται για τα modules, σύμφωνα με διάφορους οδηγούς και κώδικες, είναι η παρακάτω. Με εξαίρεση τα SPI pins, τα οποία προφανώς συνδέονται στα προκαθορισμένα pins (10-13), ο ακροδέκτης CE συνδέεται στο pin 9. Όπως θα δούμε

αργότερα βέβαια, η κατασκευή του κώδικα που υποστηρίζει τη λειτουργία αυτών των μονάδων δε συγκεκριμενοποιεί (και συνεπώς δεν περιορίζει από άποψη επιλογής) τα pins στα οποία θα συνδεθούν οι ακροδέκτες CE και CSN.

Για τη μελέτη της λειτουργίας των modules αυτών, κατασκευάσαμε μια πλακέτα,η οποία συνδέεται ως shield στον θερμοστάτη μας (UNO) και υποστηρίζει και τα δυο modules. Στην πλακέτα αυτή οι συνδέσεις έγιναν όπως αναφέρονται στην προηγούμενη παράγραφο. Παρακάτω φαίνεται η πλακέτα σκέτη και με συνδεδεμένα και τα δυο modules.



Εικόνα 5.11: Το αρχικό shield για τα NRF24L01 και NRF24L01+

5.2.3 Η Οθόνη LCD

Για την είσοδο και έξοδο δεδομένων προς και από τον μικροελεγκτή αρχικά είχε επιλεγεί μια dot matrix οθόνη με πληκτρολόγιο, αλλά τελικά επιλέχθηκε μια πιο εύχρηστη, πιο εξελιγμένη, αλλά εξίσου οικονομική λύση. Μια έγχρωμη οθόνη LCD 3.2” με resistive touch panel, και υποστήριξη TF card, από την εταιρεία UCTronics, σχεδιασμένη για Arduino

UNO/Mega2560. Η οθόνη αυτή χρησιμοποιεί τον LCD controller SSD1289 και τον touch controller ADS7843. Παρέχει ανάλυση 320x240 και χρησιμοποιεί έναν 8bit data bus, σε αντίθεση με πολλές που χρησιμοποιούν 16 bit data bus.

5.2.3.1 Επικοινωνία με τον μικροελεγκτή

Η οθόνη χρησιμοποιεί 8 digital pins για τη μεταφορά των δεδομένων (D0~7). Επιπλέον, χρησιμοποιεί το SPI για την ανάγνωση των touch δεδομένων και την ανάγνωση και αποθήκευση από και προς την TF κάρτα .

Εικόνα 5.12: Η οθόνη LCD

Αναλυτικά, η οθόνη χρησιμοποιεί τους εξής ακροδέκτες για την επικοινωνία με τον μικροελεγκτή:

D8: TF card Chip Select (TF_CS)
D9: Touch IRQ
D10: Touch Select (TCS)
D11: MOSI

D12: MISO

D13: Touch Clock (TCLK)

D0~7: data bus

A0: Chip Select (CS)

A1: Data/Command Select (RS)

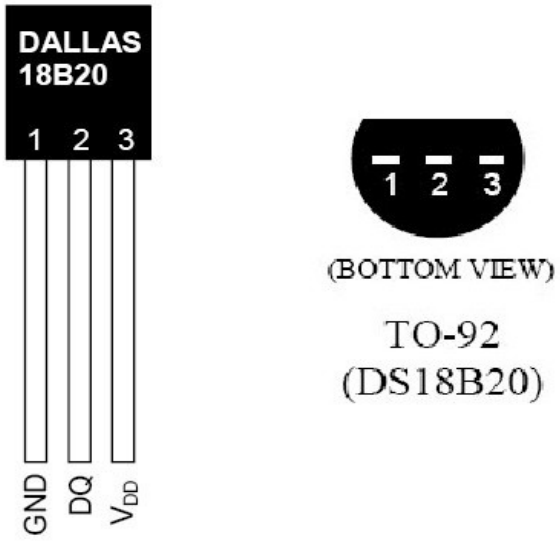
A2: Write (WR)

A3: Read (RD)

Ο ακροδέκτης TF_CS είναι όπως φαίνεται ο Chip Select ακροδέκτης του κυκλώματος της κάρτας μνήμης. Οι ακροδέκτες Touch (IRQ, TCS, TCLK) απευθύνονται στο chip που ελέγχει τη λειτουργία touch. Τα analog pins (A0~3) απευθύνονται, απ'ότι φαίνεται, στη λειτουργία της οθόνης.

5.2.4 Θερμόμετρα DS18B20

Τα θερμόμετρα DS18B20 κατασκευάζονται από την εταιρεία Maxim Integrated. Παρέχουν μετρήσεις θερμοκρασίας 9 ως 12 bit σε βαθμούς κελσίου και έχουν λειτουργία alarm με προγραμματιζόμενα μέγιστα/ελάχιστα trigger points. Έχουν τρεις ακροδέκτες, έναν για τάση, έναν για γείωση και έναν ακροδέκτη επικοινωνίας με τον μικροελεγκτή, που αποκαλείται συνήθως “One Wire”, λόγω της δυνατότητας μετάδοσης δεδομένων μέσω μόνο ενός καλωδίου.

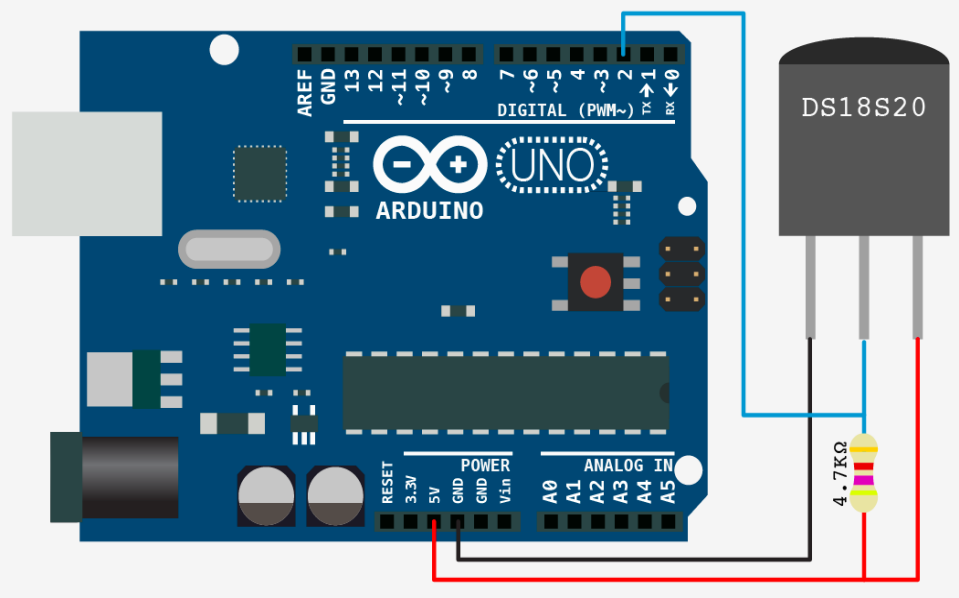


Εικόνα 5.13: Από το datasheet του DS18B20

Τα θερμόμετρα αυτά μπορούν να μετρήσουν θερμοκρασίες από -55°C μέχρι 125°C με ακρίβεια ±0.5°C τουλάχιστον στην κλίμακα από -10°C ως 85°C. Ένα πολύ σημαντικό χαρακτηριστικό τους είναι ότι μπορούν να τροφοδοτηθούν ακόμα και από τον ακροδέκτη επικοινωνίας, αποκλείοντας την ανάγκη για ακροδέκτη τροφοδοσίας. Κάθε θερμόμετρο έχει έναν ιδιαίτερο σειριακό κωδικό 64ων bit, που επιτρέπει την πολλαπλή λειτουργία θερμομέτρων μέσω ενός μόνο διαύλου.

5.2.4.1 Επικοινωνία με τον μικροελεγκτή

Η σύνδεση του θερμομέτρου στον μικροελεγκτή γίνεται συνδέοντας το One-Wire (τον ακροδέκτη δεδομένων DQ του θερμομέτρου) σε ένα από τα digital pins. Για τη λειτουργία του χρειάζεται η σύνδεση του ακροδέκτη One-Wire με τον ακροδέκτη τροφοδοσίας μέσω ενός pull-up resistor 4.7kΩ. Παρακάτω φαίνεται ένα σχηματικό από το site bildr.org.



Εικόνα 5.14: Η σύνδεση του DS18B20 στον Arduino UNO

5.2.5 Τα υπόλοιπα υλικά

Ο actuator που χρησιμοποιήσαμε είναι ένας απλός electrothermal actuator που χρησιμοποιεί On/off controls για να ενεργοποιεί βαλβίδες διαφόρων ειδών. Κυκλοφορεί σε μοντέλο που λειτουργεί με 24 ή 230V. Κάποια ενδιαφέροντα χαρακτηριστικά του είναι τα εξής:

Εικόνα 5.15: Ο Actuator που αποκτήσαμε για την εργασία

Voltage: AC 230V (AC24V available)
Power consumption: 2W
Force: 110N
Stroke: 3mm
Running time: Open-3mins, close-5mins
Valve interface: Cap nut M30*1.5mm
Ambient temperature: -5~60°C
Cable Length: 900mm
Housing material: Self-extinguishing PC (PA66 as internal components)
Protective housing: IP54
Mounting position: Vertical position to 90°declined

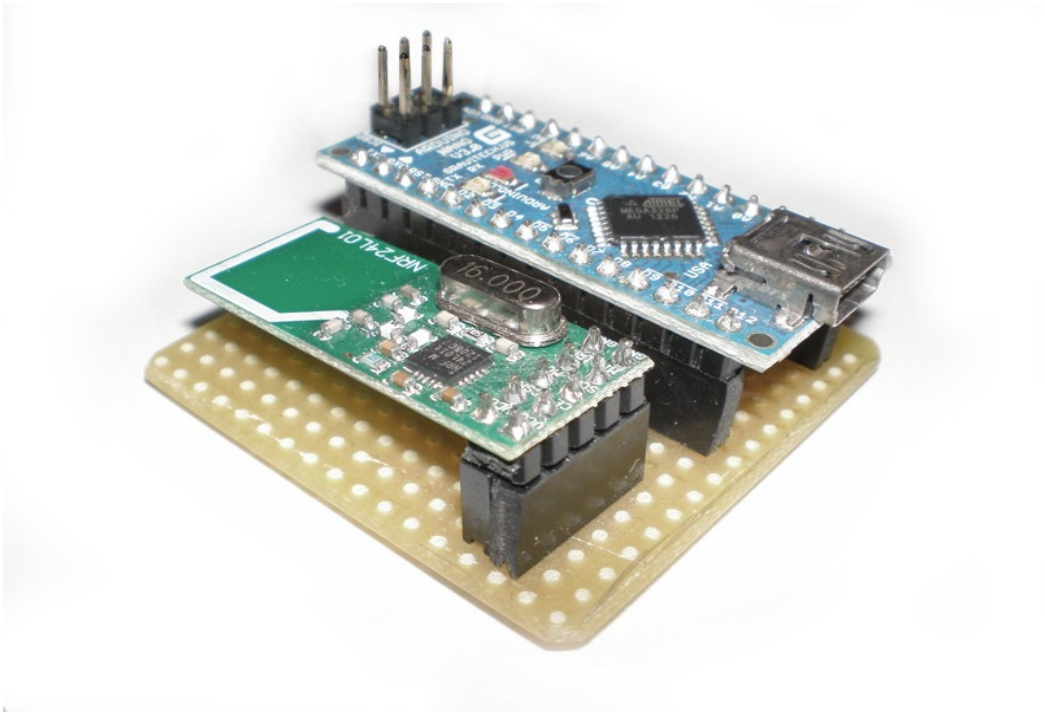
Κατά τις δοκιμές της κατασκεύης για μεγαλύτερη ευκολία η λειτουργία του actuator προσομοιώθηκε με τρία leds, χρώματος πράσινο, κίτρινο και κόκκινο. Το κίτρινο led είναι το power led, το οποίο ανάβει σηματοδοτώντας την τροφοδοσία του actuator με τάση. Το πράσινο led συμβολίζει τη λειτουργία του actuator η οποία αυξάνει τη ροή του νερού στο θερμαντικό σώμα. Το κόκκινο led συμβολίζει τη λειτουργία η οποία μειώνει τη ροή. Τα leds συνδέθηκαν μέσω αντιστάσεων 470Ω σε digital pins του μικροελεγκτή και ελέγχονται μέσω του προγράμματος λειτουργίας του. Η λειτουργία των leds θα αναλυθεί και αυτή σε επόμενο κεφάλαιο.

5.3 Η Κατασκευή των Επιμέρους Τμημάτων

Τα κυκλώματα που χρησιμοποιήθηκαν στα επιμέρους κομμάτια της εργασίας είναι τα προαναφερθέντα πέντε. Συνοπτικά, η οθόνη είναι shield και εφαρμόζει στην πλακέτα του μικροελεγκτή χρησιμοποιώντας την πλειοψηφία των διαθέσιμων pins. Οι μονάδες RF χρησιμοποιούν το SPI και εναν ακόμα ακροδέκτη. Το θερμόμετρο χρειάζεται έναν ακροδέκτη στον οποίο συνδέεται το OneWire. Τα leds που προσομοιώνουν τη λειτουργία του actuator χρειάζονται τρεις ακροδέκτες του μικροελεγκτή. Έχοντας αναλύσει τον τρόπο σύνδεσης του εκάστοτε κυκλώματος και έχοντας τεστάρει τα κυκλώματα ένα ένα για τη λειτουργία τους, το επόμενο βήμα ήταν η τοποθέτηση τους στις πλακέτες, όπου θα λειτουργήσουν παράλληλα.

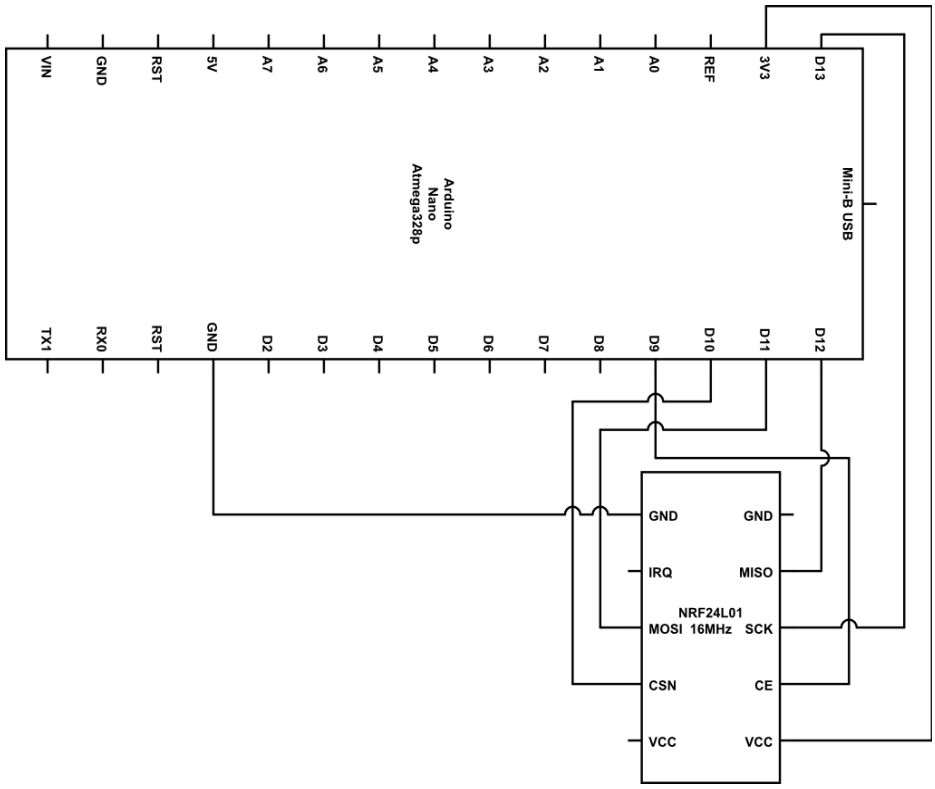
5.3.1 Η Επικοινωνία με τον Η/Υ

Για την επικοινωνία με τον Η/Υ, όπως είπαμε, χρησιμοποιήθηκε η πλακέτα Arduino Nano σε συνδυασμό με μια μονάδα NRF24L01, όπως φαίνεται σχηματικά στην Εικόνα 4.3.

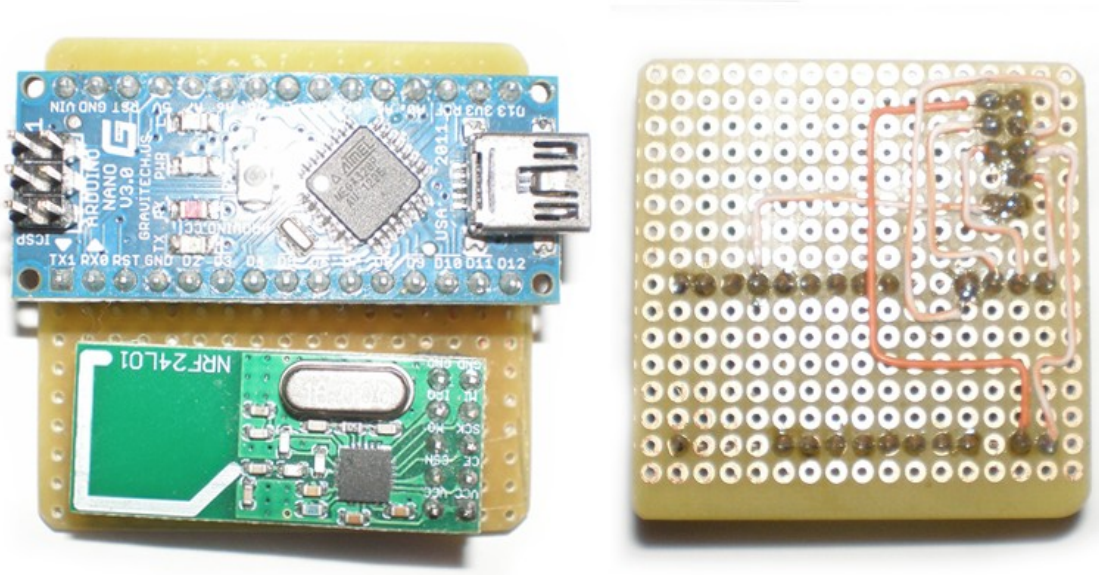


Εικόνα 5.16: Η πλακέτα για την επικοινωνία με τον Η/Υ

Η σύνδεση της μονάδας RF στον μικροελεγκτή έγινε σύμφωνα με τη σύνδεση που κάναμε όταν δοκιμάζαμε τη λειτουργία των RF modules, δηλαδή οι ακροδέκτες MISO, MOSI, SCK στα pins 11-13, ο ακροδέκτης CSN στο pin 10 και ο ακροδέκτης CE στο pin 9. Ακολουθεί το schematic και εικόνες της πλακέτας.



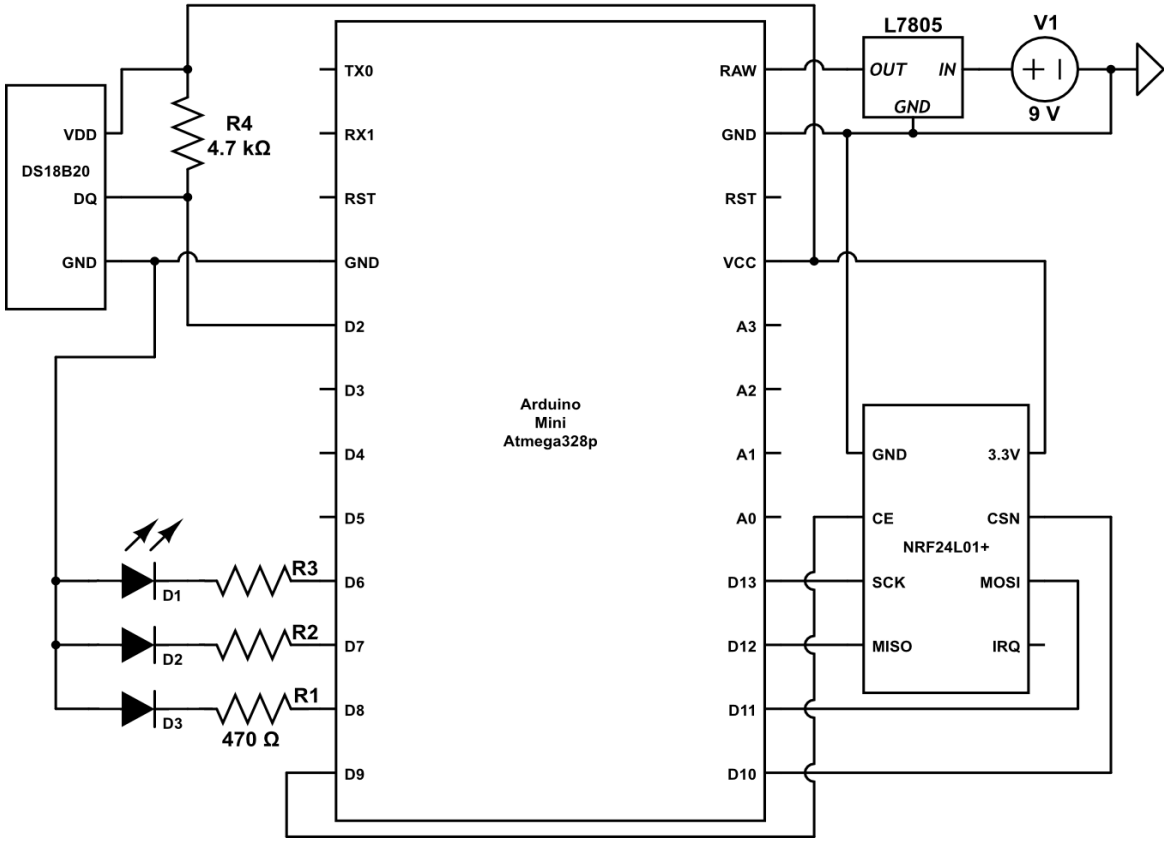
Εικόνα 5.17: Το Schematic της σύνδεσης των στοιχείων στην πλακέτα



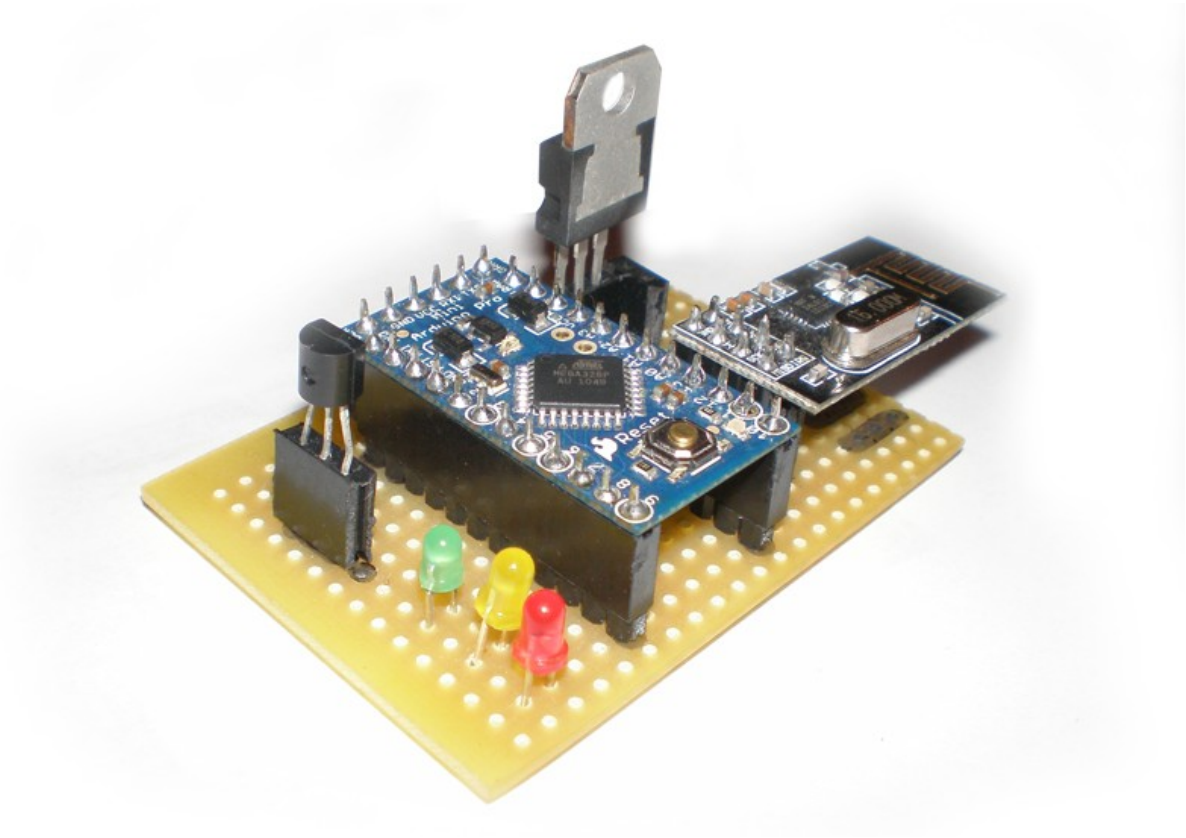
Εικόνα 5.18: Όψη της πλακέτας από πάνω και των κολλήσεων

Τα modules των σωμάτων συνδυάζουν τρία από τα κυκλώματα που αναλύθηκαν παραπάνω: μια μονάδα NRF24L01+, το θερμόμετρο DS18B20 και τα leds που προσομοιώνουν τη λειτουργία του actuator. Επειδή οι μικροελεγκτές δε θα συνδέονται μέσω κάποιας USB θύρας με κάποιον Η/Υ, φροντίσαμε για την τροφοδοσία τους. Η κάθε πλακέτα τροφοδοτείται από μια μπαταρία 9V. Οι πλακέτες Arduino Mini χρησιμοποιούν τροφοδοσία 5V, έχουν έναν ακροδέκτη RAW, ο οποίος παίρνει τάση μέχρι 12V και τη ρίχνει στα 5V. Παρ'όλα αυτά, για ασφάλεια, χρησιμοποιήσαμε έναν Voltage regulator LM7805 για να ρίξουμε την τάση πριν φτάσει στην πλακέτα.

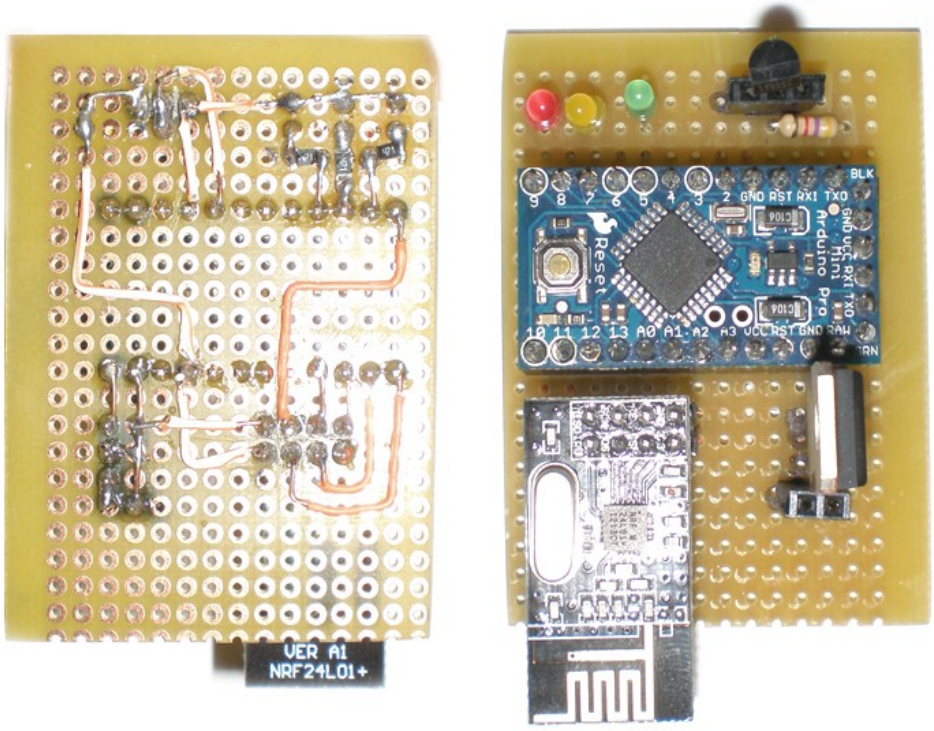
Τα RF modules συνδέθηκαν στα Arduino με την ίδια συνδεσμολογία που χρησιμοποιήσαμε στις υπόλοιπες πλακέτες ως τώρα. Το θερμόμετρο, από τη στιγμή που το pin 2 δε χρησιμοποιούνταν από κάτι άλλο, τοποθετήθηκε και αυτό με τη συνδεσμολογία δοκιμής που κάναμε. Τα τρια leds συνδέθηκαν λόγω χώρου στα pins 6,7 και 8, μέσω τριών SMD αντιστάσεων της τάξεως των 470Ω.



Εικόνα 5.19: Το schematic της σύνδεσης των στοιχείων στην πλακέτα



Εικόνα 5.20: Το module ελέγχου των θερμαντικών σωμάτων



Εικόνα 5.21: Το κύκλωμα και η πλακέτα σε άνω όψη

5.3.3 Ο Θερμοστάτης

Ο θερμοστάτης (Θ.Χ.) αποτέλεσε το πιο δύσκολο και πιο χρονοβόρο κομμάτι της εργασίας, από κατασκευαστική (αλλά και προγραμματιστική) άποψη. Τα RF modules αποδείχτηκαν πολύ απαιτητικά από άποψη κώδικα. Παρακάτω θα αναλυθούν όλα τα βήματα της κατασκευής και οι λύσεις που βρέθηκαν για τα προβλήματα που παρουσιάστηκαν κατά τη διάρκεια της υλοποίησης. Τέλος, θα παρουσιαστούν αναλυτικά όλες οι πλακέτες που χρησιμοποιήθηκαν και οι επεμβάσεις που χρειάστηκαν να γίνουν πάνω τους.

5.3.3.1 Η σύνδεση με την οθόνη

Η οθόνη που χρησιμοποιήθηκε παρουσιάστηκε παραπάνω. Αυτό που δεν αναφέρθηκε είναι ότι στην πραγματικότητα χρησιμοποιήθηκαν δύο οθόνες, η Uctronics LCD 3.2” και η LCD 3.2” Rev B. Η διαφορά των δυο αποδείχτηκε αρκετά σημαντική στη διαδικασία κατασκευής.

Η πρώτη οθόνη, η οποία αντικαταστάθηκε γιατί κάηκε ο controller της λειτουργίας αφής, απαιτούσε 2 παραπάνω ακροδέκτες, τους A4 και A5, οι οποίοι με την αντικατάστασή της απελευθερώθηκαν για να χρησιμοποιηθούν από άλλες περιφερειακές συσκευές που θα χρησιμοποιούνταν.

Η δεύτερη οθόνη λοιπόν χρησιμοποιήθηκε όπως αναφέρθηκε στην ενότητα 5.2.3. Οι ακροδέκτες A0-A3 χρησιμοποιούνται για την εμφάνιση των δεδομένων, οι ακροδέκτες D0-D7 για μεταφορά δεδομένων προς τη συσκευή, οι ακροδέκτες D11-D13 χρησιμοποιούνται από το serial peripheral interface. Η TF card χρησιμοποιεί τον ακροδέκτη D8 για slave select και ο μικροελεγκτής αφής χρησιμοποιεί τους ακροδέκτες D9 και D10 για IRQ και Slave Select αντίστοιχα. Από τη στιγμή που δε θα χρησιμοποιούσαμε την κάρτα, μπορούσαμε να χρησιμοποιήσουμε τον ακροδέκτη D8 του μC για ό,τι λειτουργία θέλουμε. Το ίδιο ισχύει και για τους ακροδέκτες A4 και A5 του μC. Αυτό στην ουσία έλυσε το πρόβλημα έλλειψης ακροδεκτών για τα υπόλοιπα περιφερειακά. Στους ακροδέκτες της οθόνης έγινε επέμβαση, κατά την οποία αφαιρέθηκαν οι ακροδέκτες A5 και D8, για να μην προκληθούν τυχόν παρεμβολές από τα σήματα αυτά.

5.3.3.2 Διαφορές UNO & Bobuino

Η πλακέτα Bobuino δεν ήταν 100% συμβατή με την UNO, όπως αναφέρθηκε στην ενότητα 5.2.1.4. Χρειάστηκε λοιπόν να επέμβουμε στην κατασκευαστική δομή των πλακετών για να αντιστοιχήσουμε τα σήματα εξόδου του μικροελεγκτή στα σήματα εισόδου της οθόνης. Παρακάτω εμφανίζονται στον πίνακα οι αντιστοιχήσεις των ακροδεκτών των μικροελεγκτών με της οθόνης και με τα generic ονόματα που τους έχουν δοθεί.

Arduino pins	UNO Ports	Bobuino Ports	UTFT pins
D0	PD0	PD0	RX
D1	PD1	PD1	TX
D2	PD2	PD2	DATA
D3	PD3	PD3	DATA
D4	PD4	**PB0**	DATA
D5	PD5	**PB1**	DATA
D6	PD6	**PB2**	DATA
D7	PD7	**PB3**	DATA
D8	PB0	**PD5**	TF_CS
D9	PB1	**PD6**	T_IRQ
D10	PB2	PB4	T_CS
D11	PB3	PB5	MOSI
D12	PB4	PB6	MISO
D13	PB5	PB7	SCK
A0	PC0	PA7	CS
A1	PC1	PA6	RS
A2	PC2	PA5	WR
A3	PC3	PA4	RD
A4	PC4	PA3	-
A5	PC5	PA2	-

Πίνακας 5.1: Το pin mapping των δυο πλακετών και της LCD shield

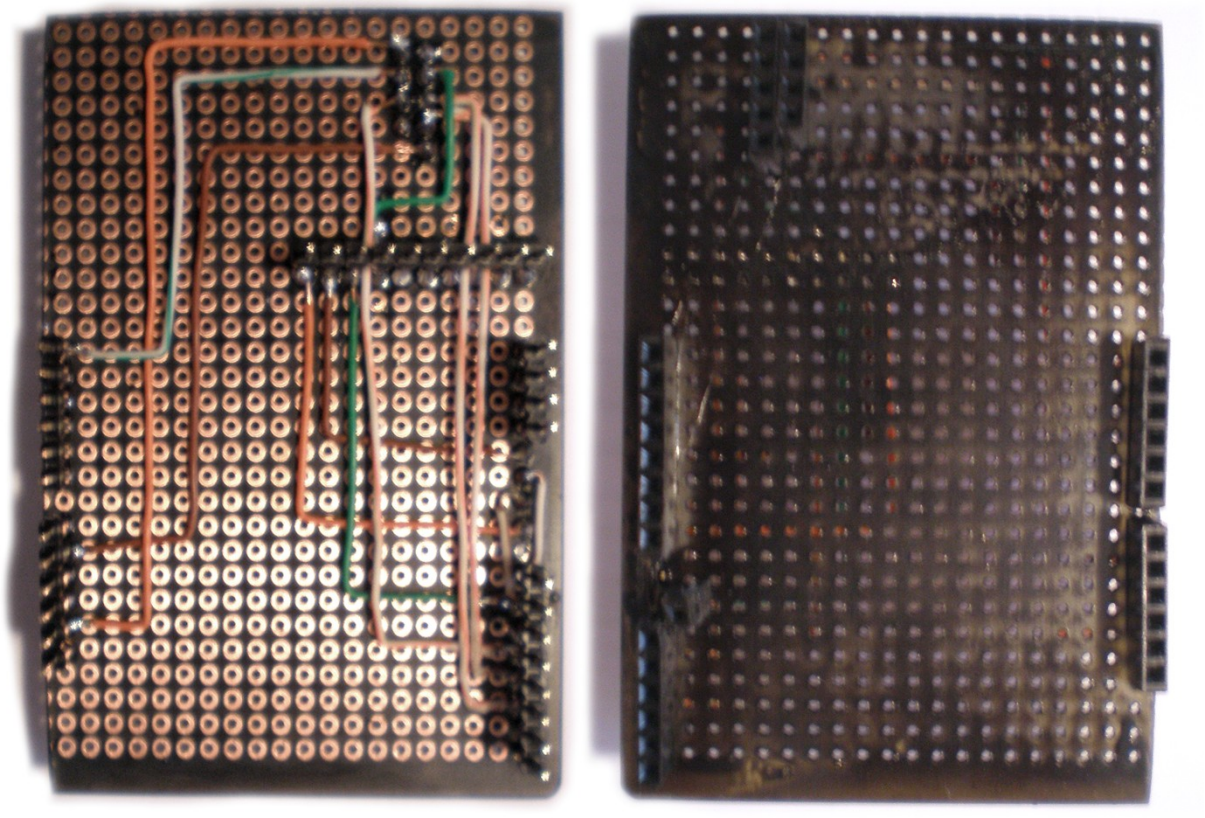
Το μόνο λοιπόν που εμφανίστηκε ως πρόβλημα είναι ότι στην ουσία τα 4 τελευταία bits της PortD χρησιμοποιούνταν από τον Bobuino με δυο τρόπους: στέλνοντας σήμα στα pins D8 και D9 (και D30, D31) αλλά και από το UTFT για τη μεταφορά των τεσσάρων από τα οκτώ data bits. Όπως προαναφέρθηκε, επειδή ακριβώς καλούνταν αυτά τα δυο pins και με το όνομα που τους δόθηκε στο pins_arduino αλλά και με το όνομα της θύρας τους, προκαλούνταν conflict. Γι αυτόν ακριβώς το λόγο η προγραμματιστική λύση δεν ήταν εφικτή. Μετά από δυο βδομάδες πειραματισμών σε επίπεδο προγραμματισμού αποφασίστηκε η επέμβαση στο hardware.

5.3.3.3 Η hardware κατασκευή

Η απόφαση αυτή περιλάμβανε τα παρακάτω: τα pins του bobuino που αντιστοιχούν στο Port PD4-PD7 θα μεταφέρονταν εκεί που έπρεπε στην LCD, δηλαδή στους ακροδέκτες DATA 4-7. Αυτό σήμαινε ότι οι έξοδοι των 8 και 9 pins, οι οποίοι αποτελούν μέρος της DATA bus αλλά και χρησιμοποιούνταν από τα Touch και TF card έπρεπε να μεταφερθούν, καθώς και ότι οι έξοδοι των pins D4-7 του bobuino δε θα χρησιμοποιούνταν πουθενά, ώστε να μην υπάρξει conflict μεταξύ ονόματος και λειτουργίας. Τα pins 8, 9, 30, 31 λοιπόν χρησιμοποιήθηκαν αντί των 4, 5, 6, 7 για το DATA bus. Αυτό όμως σήμαινε ότι θα έπρεπε να μεταφερθούν σε άλλα pins οι ακροδέκτες της LCD για TF_CS (pin D8) και T_IRQ (pin D9). Ευτυχώς, ο ακροδέκτης 8 (TF_CS) δε χρησιμοποιείται, καθώς δεν έχουμε κάποια κάρτα TF, άρα εύκολα μπορεί να αφεθεί "στον αέρα". Από την άλλη, η library για τη λειτουργία αφής μας επιτρέπει να επιλέξουμε σε ποιον ακροδέκτη αναθέτουμε τη λειτουργία IRQ του Touch, συνεπώς μεταφέραμε σε κάποιο άλλο pin τη σύνδεση, εμείς επιλέξαμε το pin D29.

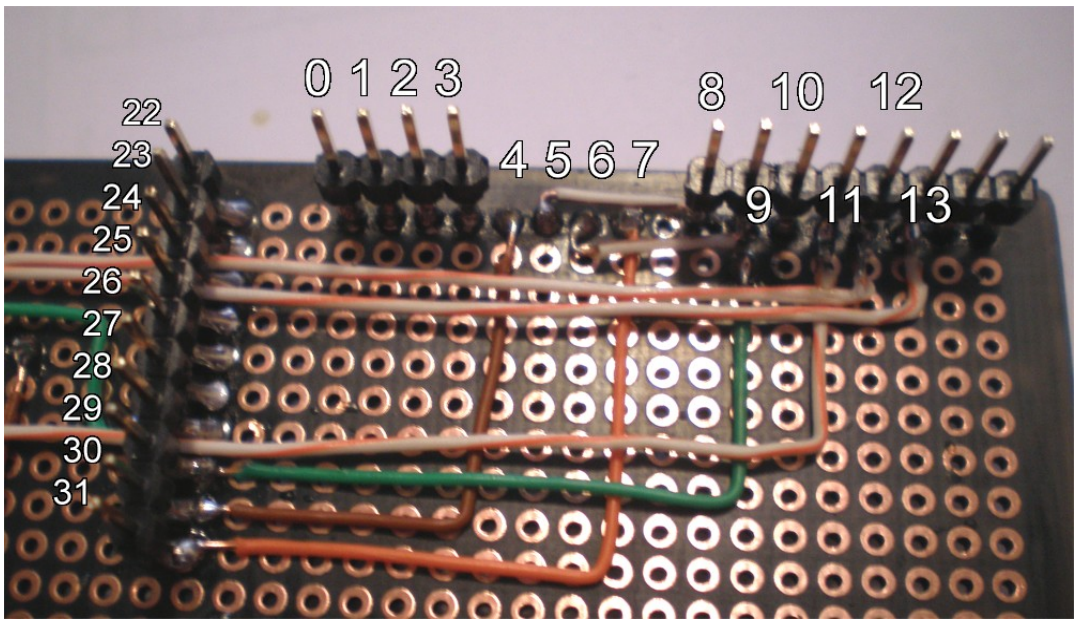
Όσον αφορά στη σύνδεση του RF module, αυτό, όπως προείπαμε έχει 5 σημαντικούς ακροδέκτες, τους τέσσερεις SPI (SS, MISO, MOSI, SCK) και τον CE. Προφανώς οι τρεις SPI (MISO, MOSI, SCK) συνδέονται στο SPI του μικροελεγκτή. Οι υπόλοιποι δυο ακροδέκτες μπορούν να συνδεθούν σε όποιο pin επιθυμούμε. Εμείς επιλέξαμε το A5 του Bobuino για να συνδέσουμε το Chip Select και το D28 για το Chip Enable.

Για να γίνουν εφικτές αυτές οι αλλαγές στις συνδέσεις των τριων αυτών συσκευών, κατασκευάστηκε μια πλακέτα, η οποία τοποθετήθηκε ανάμεσα στο Bobuino και την LCD οθόνη, έχοντας και 8 ακροδέκτες για τη σύνδεση του RF module. Η πλακέτα αυτή εμφανίζεται παρακάτω με λεπτομέρειες, για να γίνουν πιο ξεκάθαρες οι μετατροπές που κάναμε στη μεταφορά των σημάτων.

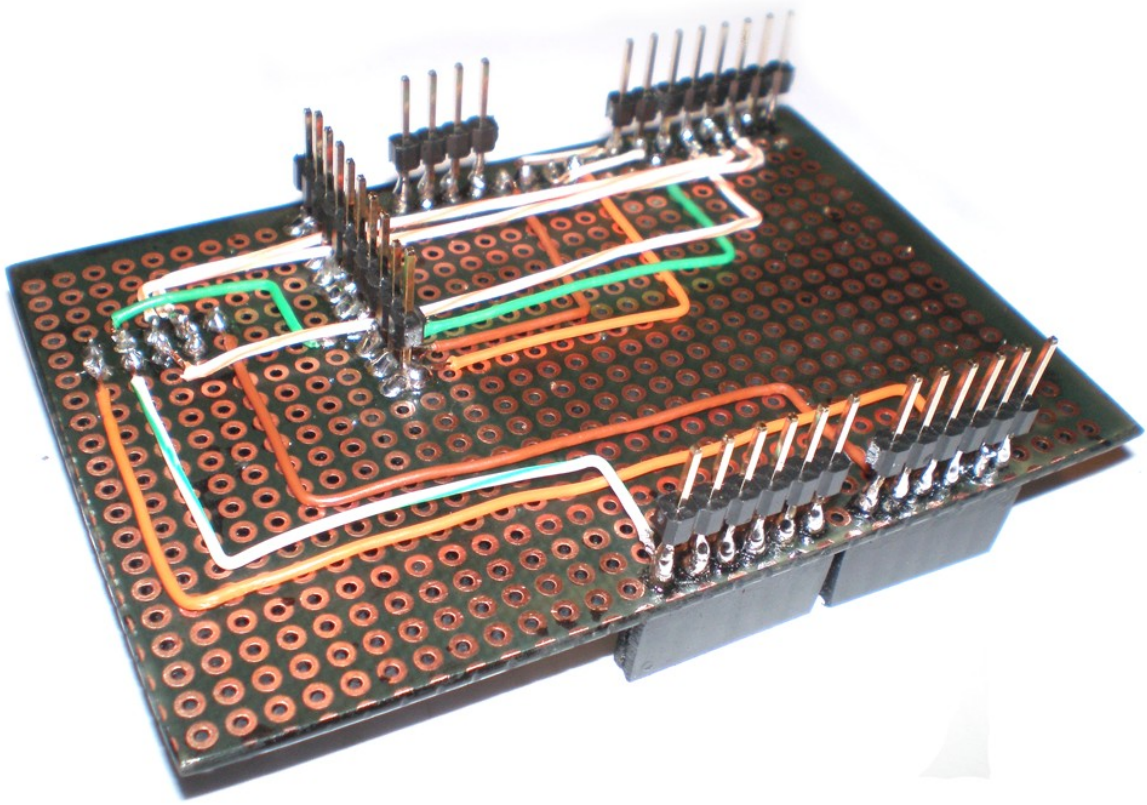


Εικόνα 5.22: Η πλακέτα του Θερμοστάτη σε κάτοψη

Όπως φαίνεται στην παραπάνω εικόνα, τα pins D4-D7 έχουν αφαιρεθεί και έχουν αντικατασταθεί με συνδέσεις με τα pins D8, D9, D30, D31. Για να γίνει πιο ξεκάθαρο παρατίθεται μια λεπτομέρεια του σημείου παρακάτω:

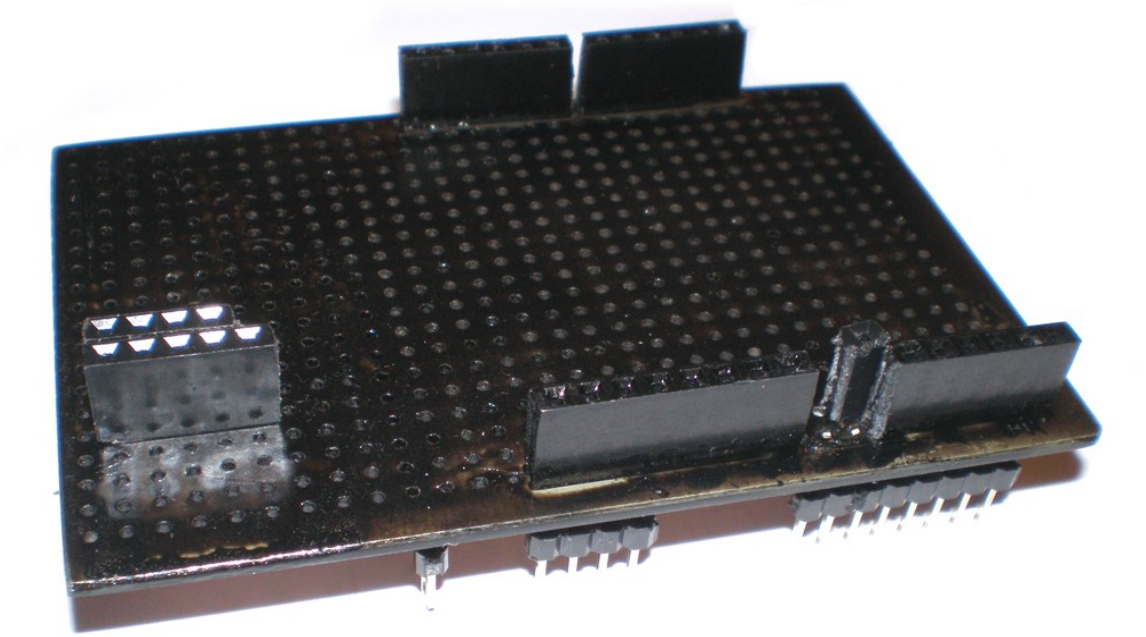


Εικόνα 5.23: Οι αντικαταστάσεις των D4-D7 με τα D30, D8, D9, D31 αντίστοιχα



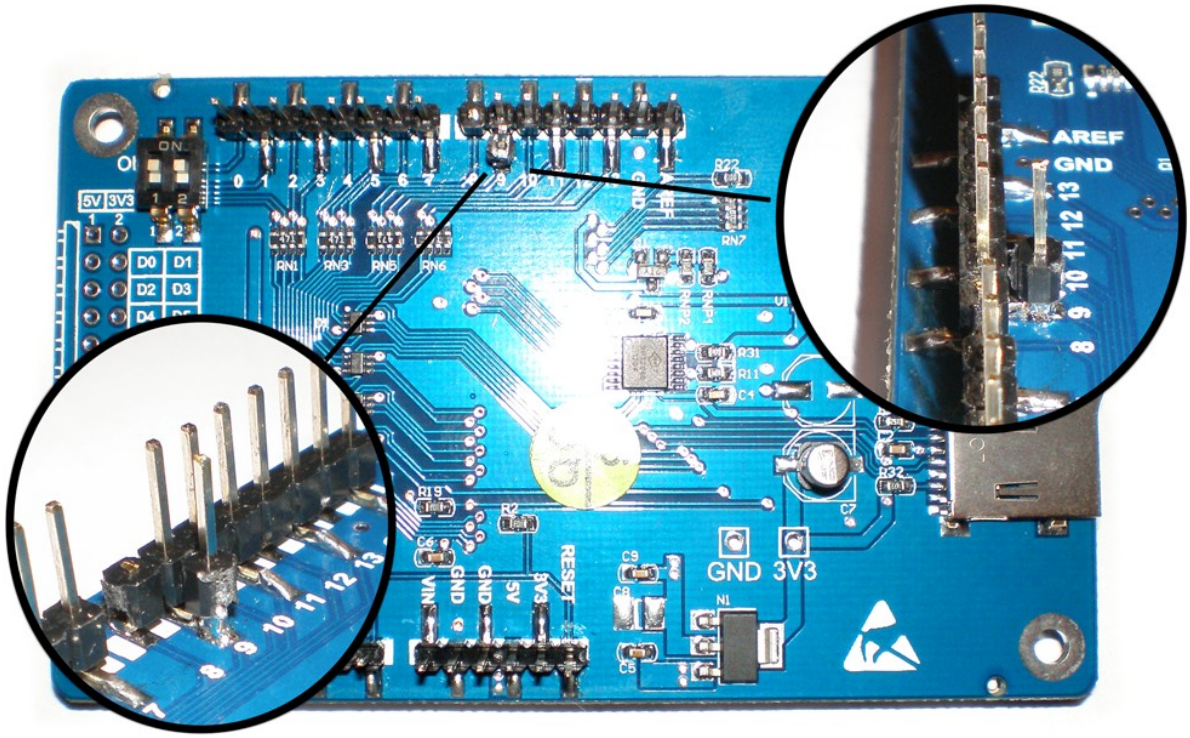
Εικόνα 5.24: Οι συνδέσεις των ακροδεκτών στην πλακέτα του Θερμοστάτη

Για να μπορέσουμε να χρησιμοποιήσουμε τον ακροδέκτη 9 της οθόνης χωρίς να υπάρχει σύνδεση με τον ακροδέκτη D9 του μικροελεγκτή χρειάστηκε να μετακινηθεί προς το εσωτερικό της πλακέτας ο ακροδέκτης, όπως φαίνεται στην άνω όψη της πλακέτας:



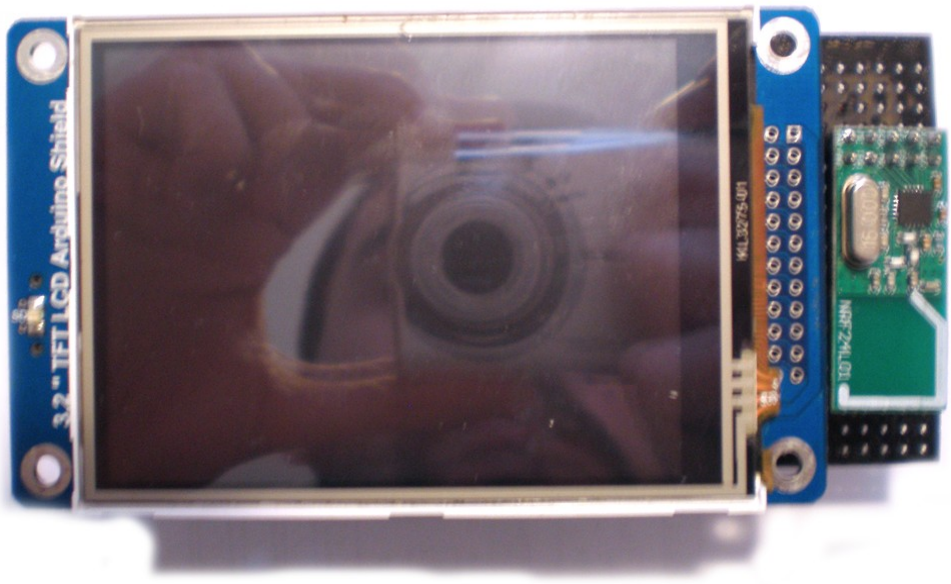
Εικόνα 5.25: Άνω όψη της πλακέτας του Θερμοστάτη

Αντίστοιχα, στην οθόνη έπρεπε να μεταφερθεί το pin πιο μέσα, έτσι ώστε να συνδέεται με το θηλυκό στην πλακέτα του Θερμοστάτη:

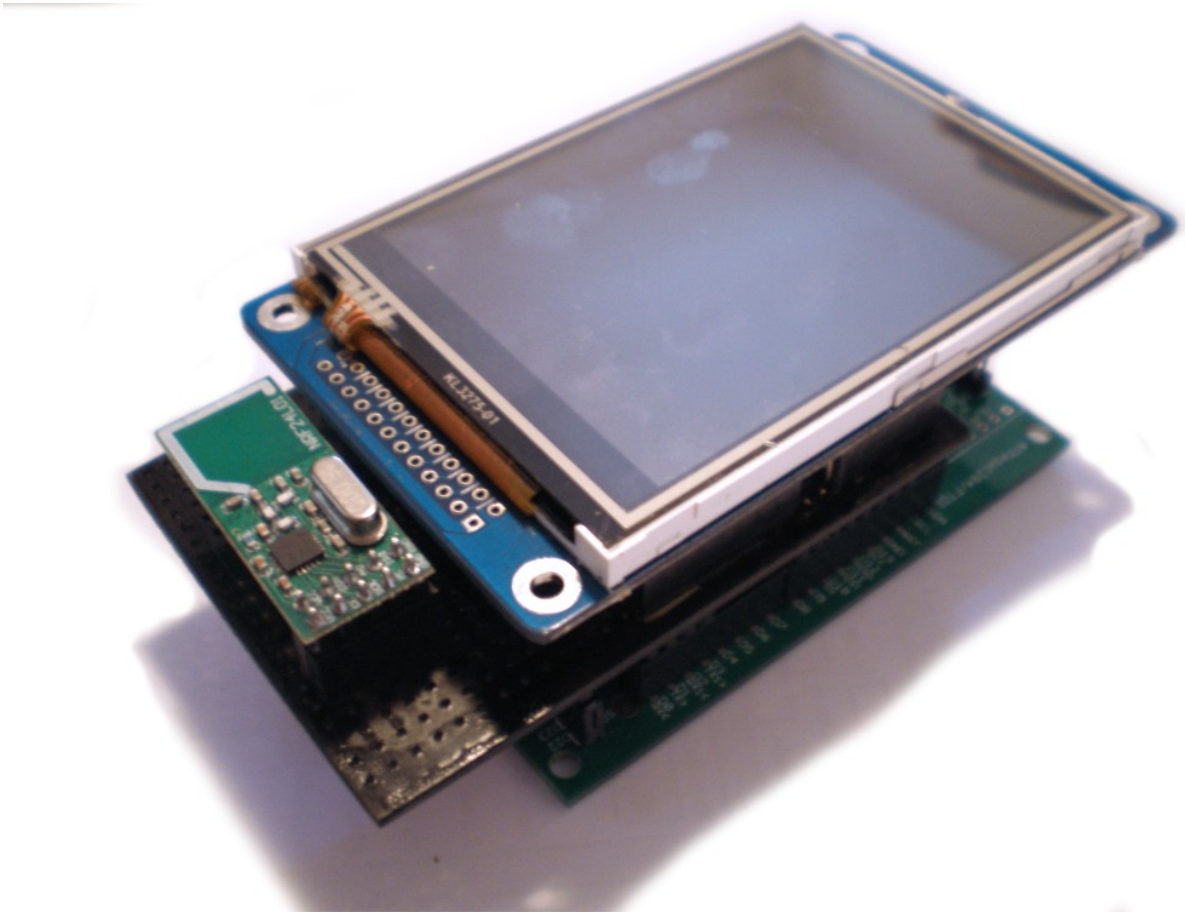


Εικόνα 5.26: Κάτω όψη της οθόνης LCD με λεπτομέρειες στο Pin9

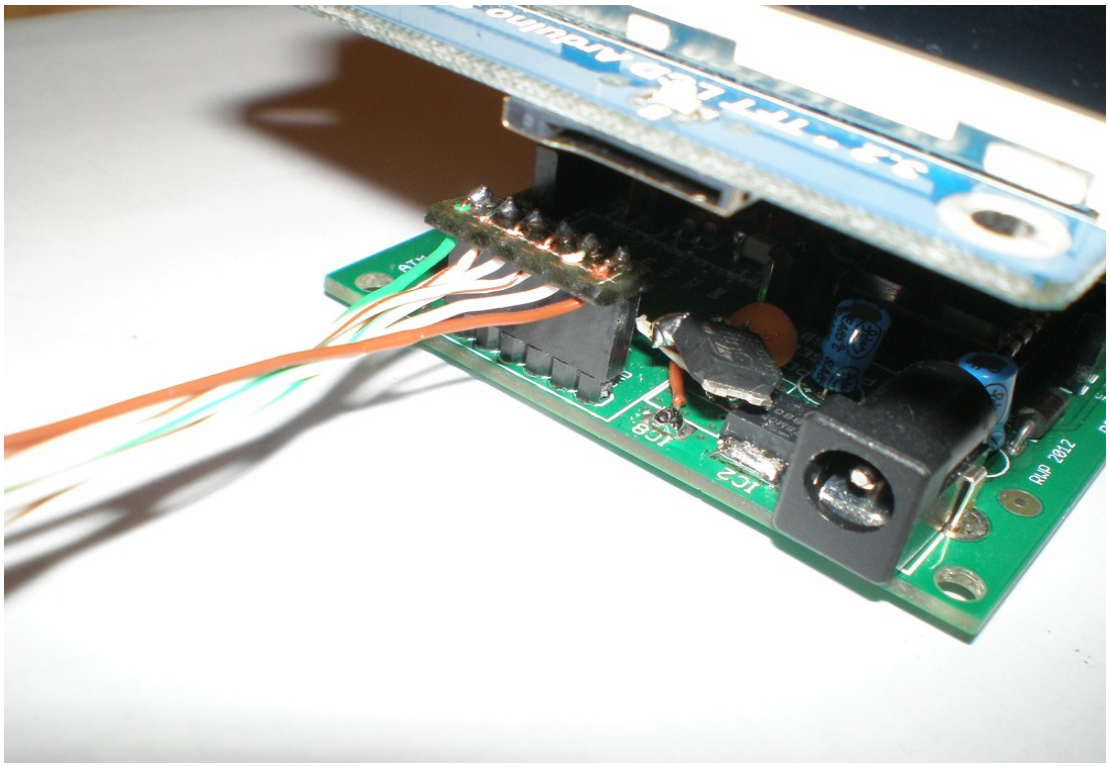
Με αυτές τις δυο μεταβολές, η hardware κατασκευή της πλακέτας ήταν έτοιμη. Παρατίθενται στη συνέχεια φωτογραφίες του θερμοστάτη.



Εικόνα 5.27: Άνω όψη της πλακέτας του Θερμοστάτη



Εικόνα 5.28: Ο Θερμοστάτης



Εικόνα 5.29: Λεπτομέρεια από το FTDI καλώδιο τροφοδοσίας του Θερμοστάτη

Ακολουθεί το σχηματικό διάγραμμα και ένας αναλυτικός πίνακας των συνδέσεων του μικροελεγκτή και των λειτουργιών του κάθε pin

Pin	**Λειτουργία**	**Σύνδεση με**	**Ακροδέκτης περιφερειακού**	**Σχολια**
D0	RX	-		
D1	TX	-		
D2	DATA	LCD	2	
D3	DATA	LCD	3	
D4	-	-		Δε χρησ/ται
D5	-	-		Δε χρησ/ται
D6	-	-		Δε χρησ/ται
D7	-	-		Δε χρησ/ται
D8	DATA	LCD	5	μεταφορα
D9	DATA	LCD	6	μεταφορα
D10	Touch Select	Touch	10	
D11	MOSI	SPI	11-M0	
D12	MISO	SPI	12-MI	
D13	SCK	SPI	13-SCK	
D14-A0	LCD Select	LCD	A0	
D15-A1	RS	LCD	A1	
D16-A2	WR	LCD	A2	
D17-A3	RD	LCD	A3	
D18-A4	-	-		Δε χρησ/ται
D19-A5	RF Select	NRF24	CSN	
D20~D27	-	-		Δε χρησ/νται
D28	Chip Enable	NRF24	CE	
D29	Touch IRQ	Touch	9	
D30	DATA	LCD	4	μεταφορα
D31	DATA	LCD	7	μεταφορα

Πίνακας 5.2: Το pin mapping της πλακέτας και η σύνδεση με τα περιφερειακά

ΚΕΦΑΛΑΙΟ ΕΚΤΟ

Ο Προγραμματισμός

Για τον προγραμματισμό των μικροελεγκτών υπάρχουν αρκετά αναπτυξιακά εργαλεία. Ένα αρκετά διαδεδομένο, το οποίο χρησιμοποιήσαμε, είναι το AVR Studio. Προσφέρει μεγάλη ποικιλία επιλογών και καλύπτει μεγάλο αριθμό μικροελεγκτών. Από τη στιγμή όμως που χρησιμοποιήσαμε μικροελεγκτές με πλακέτες Arduino, θεωρήθηκε καλύτερη λύση η χρήση του open source compiler που προσφέρεται από τους κατασκευαστές του Arduino.

Λόγω του γεγονότος ότι το software που προσφέρει το Arduino είναι ανοιχτού κώδικα έχουν προκύψει διάφορες μετατροπές του, όπως για παράδειγμα το Codebender project, που προσφέρει online compiling σε κώδικα Arduino και τη δυνατότητα να γραφτεί ο κώδικας στον μικροελεγκτή χωρίς καν να χρειάζεται να τρέξει ο χρήστης το πρόγραμμα από τον υπολογιστή του.

Στο κεφάλαιο αυτό θα αναλυθούν οι βιβλιοθήκες που χρησιμοποιήθηκαν καθώς και κάποια σημαντικά σημεία του κώδικα που γράφτηκε, ώστε να γίνει εύκολα κατανοητή η ανάγνωση του πλήρους κώδικα στο επόμενο κεφάλαιο. Οι βιβλιοθήκες και τα αντίστοιχα κομμάτια για διαφορετική λειτουργία θα αναλυθούν παράλληλα.

Αξίζει να αναφερθούν οι κατασκευαστές βιβλιοθηκών και οι συγγραφείς προγραμμάτων που χρησιμοποιήθηκαν στη σύνταξη των κωδίκων για αυτή την εργασία. Η βιβλιοθήκη OneWire, που επεξεργάζεται τα σήματα του θερμομέτρου DS18B20, γράφτηκε από τον Jim Studt. Η βιβλιοθήκη ArduCam Touch για τη λειτουργία Touch της οθόνης γράφτηκε από την εταιρεία uCtronics. Οι βιβλιοθήκες UTFT γράφτηκαν από τον Henning Karlsen και βελτιώθηκαν από την uCtronics. Οι βιβλιοθήκες για τα NRF συντάχθηκαν από τον James Coliz, γνωστό και ως Maniacbug.

Το πρόγραμμα στην πλευρά του Η/Υ γράφτηκε σε γλώσσα Java. Για τη σειριακή επικοινωνία χρειάστηκε η βιβλιοθήκη RxTx, η οποία γράφτηκε από τον Trent Jarvi. Το πρόγραμμα που γράφτηκε βασίστηκε στο πρόγραμμα Tiger Control Panel του Henry Poon.

6.1 Προγραμματισμός των Μικροελεγκτών

6.1.1 Μέτρηση θερμοκρασιών με θερμόμετρα DS18B20

Όπως είπαμε, για τα DS18B20, χρησιμοποιήσαμε τη βιβλιοθήκη OneWire. Η βιβλιοθήκη αυτή έχει βασιστεί σε sample code που παρέχεται από την Dallas Semiconductors. Το κάθε θερμόμετρο έχει μοναδικό σειριακό κωδικό 64bit, που καθιστά δυνατή τη σύνδεση πολλαπλών θερμομέτρων σε έναν μόνο δίαυλο. Σε κάθε επανάληψη του κώδικα, καλείται μια διεργασία η οποία απαριθμεί τα συνδεδεμένα στον "one wire" δίαυλο καλώδια. Η μέτρηση των θερμοκρασιών γίνεται με τη χρήση 9-12 bytes. Η αναγνώριση σφαλμάτων γίνεται μέσω ενός 8 bit CRC. Είναι δυνατόν να χρησιμοποιηθεί και 16bit CRC.

Ο κώδικας που χρησιμοποιείται για τη μέτρηση των θερμοκρασιών είναι μια function με το όνομα getTemp, η οποία επιστρέφει τη θερμοκρασία που στέλνει ένα θερμόμετρο σε βαθμούς Κελσίου.

Αρχικά καλείται η function "search", η οποία ψάχνει για ένα συνδεδεμένο θερμόμετρο και επιστρέφει τη διεύθυνση του θερμομέτρου που θα βρεί. Αν δε βρει θερμόμετρο ή το CRC είναι λανθασμένο ή είναι συνδεδεμένη άλλη συσκευή (το 0x28 αποτελεί το family code των DS18B20 1-wire, ενώ το 0x10 αποτελεί το family code των απλών DS18B20), επιστρέφει την τιμή "-1000". Αν βρει θερμόμετρο συνδεδεμένο, ανοίγει επικοινωνία μαζι του και του ζητά τη θερμοκρασία.

Συγκεκριμένα στέλνει την εντολή "Convert T" (0x44), η οποία κάνει το DS18B20 να ξεκινήσει τη μετατροπή θερμοκρασίας και να στείλει στον μικροελεγκτή το conversion status. Στη συνέχεια, με την εντολή "Read Scratchpad" (0xBE), το θερμόμετρο αποστέλλει μέχρι 9 bytes δεδομένων, συμπεριλαμβανομένου του CRC byte. Τα bytes αυτά αποθηκεύονται σε έναν πίνακα data[]. Τέλος, τα δεδομένα αυτά μετατρέπονται σε float αριθμό και τα επιστρέφονται.

Στον παρακάτω κώδικα, το όρισμα ds αποτελεί ένα instance του OneWire.

ΚΩΔΙΚΑΣ 6.1: Η διεργασία getTemp που επιστρέφει την double τιμή θερμοκρασίας

```
float getTemp(){
  //returns the temperature from one DS18S20 in DEG Celsius

  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
     //no more sensors on chain, reset search
     ds.reset_search();
     return -1000;
  }
  if ( OneWire::crc8( addr, 7) != addr[7]) {
     Serial.println("CRC is not valid!");
     return -1000;
  }
  if ( addr[0] != 0x10 && addr[0] != 0x28) {
     Serial.print("Device is not recognized");
     return -1000;
  }

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

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad

  for (int i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB); //using two's compliment
  float TemperatureSum = tempRead / 16;

  return TemperatureSum;

}
```

6.1.2 Η επικοινωνία μέσω RF modules

Η επικοινωνία μέσω RF modules αποτελεί μια αρκετά πολύπλοκη διαδικασία που πρέπει να εξετάσουμε σε βάθος. Κατά τη διάρκεια εκμάθησης των λειτουργιών των NRF24 modules ξεκινήσαμε δουλεύοντας με τη βιβλιοθήκη RF24, η οποία όμως είναι κατασκευασμένη για επικοινωνία μεταξύ δυο modules μόνο. Αυτό δε σημαίνει ότι δε μπορούν να συμμετέχουν περισσότερα, αλλά ότι στην ουσία πρέπει να ορίζονται pipes εισόδου και εξόδου για κάθε node και να αντιστοιχίζονται σε αυτά. Αντίθετα, η βιβλιοθήκη RF24Network λειτουργεί με τέτοιον τρόπο ώστε να μπορούν να επικοινωνούν περισσότερα nodes μεταξύ τους χωρίς να χρειάζεται να οριστούν pipes και να αντιστοιχηθούν σε nodes. Είναι στην ουσία ένα implementation της βιβλιοθήκης RF24, που αυτοματοποιεί την επικοινωνία ενός συνόλου από modules. Η βιβλιοθήκη RF24Network στηρίζεται σε και χρησιμοποιεί τη βιβλιοθήκη RF24, πράγμα που σημαίνει ότι χρειαζόμαστε και τις δυο για τη λειτουργία των modules.

6.1.2.1 Η επικοινωνία μέσω RF24

Όπως αναφέρθηκε προηγουμένως, η βιβλιοθήκη RF24 είναι έτσι κατασκευασμένη ώστε να επιτρέπει την επικοινωνία μεταξύ δυο μόνο nodes. Τη βιβλιοθήκη αυτή τη χρησιμοποιήσαμε για την κατανόηση της λειτουργίας των NRF24 modules και την εκμάθηση των κυριότερων εντολών που χρησιμοποιούνται στη μεταξύ τους επικοινωνία. Παρακάτω θα εξετάσουμε συνοπτικά κάποιες από τις ουσιώδεις αυτές εντολές με τα ορίσματά τους. Αρχικά, όπως είχε αναφερθεί και σε προηγούμενο κεφάλαιο, να θυμίσουμε ότι τα NRF24 modules χρησιμοποιούν για την επικοινωνία τους το SPI. Συγκεκριμένα, τα σήματα MISO, MOSI και SCK μεταδίδονται από τα pins 11-13 του μικροελεγκτή αντίστοιχα. Το σήμα CE και το σήμα CSN μπορούν να μεταδίδονται από όποια pins ορίσουμε εμείς.

Στην αρχή του προγράμματος εμείς πρέπει να ορίσουμε τα pins των CE και CSN για να τα περάσουμε ως ορίσματα στο RF24. Έστω ότι το instance του RF24 που θα χρησιμοποιήσουμε θα το ονομάσουμε radio. Στον παρακάτω πίνακα παρατίθενται οι βασικότερες εντολές του RF24 με το ρόλο τους.

Εντολή	Ρόλος
RF24 radio(rf_ce, rf_csn);	Δημιουργεί ένα instance του RF24, ορίζοντας τα pins στα οποία είναι συνδεδεμένα τα σήματα CE και CSN
EEPROM.write(address_at_eeprom_location, int node_addr);	Γράφει στην EEPROM του μC τη διεύθυνση που του αναθέτουμε
EEPROM.read(address_at_eeprom_location);	Διαβάζει από την EEPROM τη διεύθυνση που έχουμε αναθέσει στον μικροελεγκτή
radio.begin();	Ξεκινά το RF δίκτυο
radio.startListening();	Ακούει για δεδομένα που απευθύνονται στο node
radio.stopListening();	Διακόπτει τη λειτουργία σκαναρίσματος για δεδομένα
radio.read(&variable, sizeof(variable type));	Λαμβάνει ένα πακέτο και το αποθηκεύει
radio.write(&variable, sizeof(variable_type));	Αποστέλλει ένα πακέτο

Πίνακας 6.1: Μερικές εντολές της βιβλιοθήκης RF24

Με αυτές τις εντολές είναι εύκολο να επιτευχθεί η επικοινωνία μεταξύ δυο τουλάχιστον nodes. Για την εργασία χρειαστήκαμε μεγαλύτερο επίπεδο λειτουργικότητας στην ανταλλαγή δεδομένων και γι αυτό δε θα αναλύσουμε περαιτέρω τη βιβλιοθήκη αυτή, αλλά θα περάσουμε αμέσως στην ανάλυση της βιβλιοθήκης RF24Network.

6.1.2.2 Η κατασκευή του RF24 Network

Το RF24 Network στηρίζει τη λειτουργία του στη βιβλιοθήκη RF24, στην ουσία δημιουργεί ένα δίκτυο το οποίο δημιουργεί μια δομή δρομολόγησης δεδομένων χωρίς να χρειάζεται ο χρήστης να ορίζει pipes εισόδου και εξόδου για κάθε node ξεχωριστά και επιπλέον δημιουργεί ένα απλό αλλά αποδοτικό δίκτυο δρομολόγησης δεδομένων μέσω των

nodes, ώστε να μη χρειάζεται να επικοινωνούν άμεσα μεταξύ τους.

Η κατασκευή του δικτύου επικοινωνίας μεταξύ των nodes είναι σε σχήμα δέντρου, στο οποίο υπάρχει μια ρίζα. Η ρίζα αυτή έχει child nodes και αυτά με τη σειρά τους έχουν child nodes. Κάθε node μπορεί να επικοινωνεί άμεσα μόνο με το parent node του και τα δικά του φύλλα. Κάθε μήνυμα που απευθύνεται σε κάποιο άλλο node, δρομολογείται μέσω των nodes που παρεμβάλλονται μεταξύ του αποστολέα και του παραλήπτη.

Το δίκτυο αυτό ορίζεται με την εντολή "RF24Network network(radio);", όπου radio είναι ένα instance της βιβλιοθήκης RF24 που πρέπει να έχουμε ορίσει πριν επιχειρήσουμε να κατασκευάσουμε το δίκτυο. Στη συνέχεια, αφού έχουμε ξεκινήσει (begin) τη λειτουργία "radio", ξεκινάμε και τη λειτουργία του network με την εντολή "network.begin(channel, node_address);".

Στη συνέχεια χρησιμοποιούμε τις παρακάτω απλές εντολές για την επικοινωνία:

Εντολή	Ρόλος
network.update();	Σκανάρει το δίκτυο για δεδομένα
network.available();	'True' αν υπάρχουν δεδομένα για το node
RF24NetworkHeader header;	Ορίζει την κεφαλίδα του μηνύματος για να αποθηκεύσει τη διεύθυνση αποστολέα / παραλήπτη
network.read(header, &msg, sizeof(msg));	Διαβάζει το πακέτο που λαμβάνει και αποθηκεύει τη διεύθυνση αποστολέα, τα bytes του μηνύματος και το μέγεθος του μηνύματος
network.write(header, &msg, sizeof(msg));	Στέλνει ένα πακέτο το οποίο περιέχει τη διεύθυνση παραλήπτη, το μήνυμα και το μέγεθος του μηνύματος

Πίνακας 6.2: Μερικές εντολές της βιβλιοθήκης RF24Network

Αρκετή σημασία πρέπει να δοθεί στον ορισμό των διευθύνσεων των nodes. Αυτά ορίζονται στο αρχείο nodeconfig.cpp. Ο προγραμματιστής της βιβλιοθήκης φρόντισε να ορίσει ένα σταθερό τρόπο ονοματοδοσίας των nodes που να είναι πιστός στη δομή δέντρου που κατασκεύασε. Έτσι, στο αρχείο αυτό παρατηρούμε ότι η διευθυνσιοδότηση των nodes γίνεται επιλέγοντας διεύθυνση από έναν πίνακα, με την εντολή

"const uint16_t node_address_set[10] = { 00, 02, 05, 012, 015, 022, 025, 032, 035, 045 };"

όπου μπορούμε να παρατηρήσουμε ότι η διεύθυνση 00 αντιστοιχεί στη ρίζα του δένδρου. Οι διευθύνσεις 02 και 05 αντιστοιχούν στα παιδιά της ρίζας και στη συνέχεια τα παιδιά αυτών ορίζονται με έναν τριψήφιο αριθμό, ο οποίος ξεκινά με το 0, καταλήγει στον αριθμό του parent node και στο ενδιάμεσο παρεμβάλλει την τιμή του εκάστοτε παιδιού. Συνεπώς, τα παιδιά του node '02' θα ονομάζονται '012, 022, 032, 042' και ούτω καθ'εξής. Αντίστοιχα, τα παιδιά του node '05' θα ονομάζονται '015, 025, 035, 045' κοκ. Ο κώδικας αυτός στη συνέχεια περιλαμβάνει τις εντολές 'nodeconfig_read' και 'nodeconfig_listen'. Με την πρώτη εντολή διαβάζεται από την EEPROM του μC η διεύθυνση που του έχει δοθεί. Αν δεν έχει δοθεί διεύθυνση, ο χρήστης καλείται να εισάγει την επιθυμητή διεύθυνση, η οποία αποθηκεύεται στην EEPROM μέσω της εντολής 'nodeconfig_listen'.

6.1.3 Η οθόνη αφής

Η οθόνη αφής περιλαμβάνει δυο διαφορετικές λειτουργίες, την εμφάνιση και την ανάγνωση δεδομένων, οι οποίες ορίζονται από δυο διαφορετικές βιβλιοθήκες, τη UTFT και την ArduCAM_Touch αντίστοιχα. Ο τρόπος λειτουργίας της οθόνης δεν είναι δύσκολο να γίνει αντιληπτός. Ένα σημαντικό κομμάτι όμως είναι το calibration της οθόνης, το οποίο γίνεται μια φορά και απαιτείται για να διαβάζει η οθόνη σωστά τις συντεταγμένες αφής.

6.1.3.1 Η εμφάνιση των δεδομένων

Η βιβλιοθήκη της οθόνης, UTFT, χρησιμοποιεί επιπλέον τις βιβλιοθήκες SD και SPI, για τη λειτουργία της κάρτας μνήμης που μπορεί να διαθέτει για το διάβασμα διαφόρων γραφικών. Επειδή η βιβλιοθήκη αυτή καλύπτει ένα ευρύ φάσμα διαφορετικών LCD microcontrollers, έχει γραφτεί έτσι ώστε να μπορεί ο χρήστης εύκολα να ορίζει ποιον

controller έχει και ποια ATMega pins ανταποκρίνονται στη μεταφορά δεδομένων εξόδου.

Στην αρχικοποίηση της βιβλιοθήκης κατασκευάζουμε ένα instance του UTFT, πχ το myGLCD με την εντολή “myGLCD(model, RS, WR, CS, RD);”, όπου ορίζουμε το μοντέλο του controller που χρησιμοποιεί η οθόνη και ποια pins του ATMega θα χρησιμοποιηθούν για τις λειτουργίες Data/Command_Select, Write, Chip Select και Read αντίστοιχα. Στη δική μας περίπτωση χρησιμοποιούμε τον controller ITDB32S (SSD1289 στην ουσία) και τα pins είναι τα A1, A2, A0, A3 αντίστοιχα.

Ακολουθούν μερικές από τις σημαντικότερες εντολές της οθόνης

Εντολή	Ρόλος
myGLCD.initLCD();	Ξεκινά τη λειτουργία της οθόνης, παίρνει όρισμα 0 ή 1, δηλαδή PORTRAIT ή LANDSCAPE. Το όρισμα PORTRAIT σημαίνει κάθετη οθόνη, όπως πχ σε ένα smartphone, ενώ το LANDSCAPE οριζόντια, όπως οι οθόνες των PC
myGLCD.clrScr();	Σβήνει όλα τα περιεχόμενα της οθόνης
myGLCD.setFont();	Θέτει τη γραμματοσειρά που θα χρησιμοποιήσουμε. Οι βασικές γραμματοσειρές που χρησιμοποιούμε είναι οι 'SmallFont' και 'BigFont'
myGLCD.setColor(R, G, B);	Θέτει το χρώμα που θα χρησιμοποιήσουμε για οτιδήποτε εκτυπώσουμε
myGLCD.setBackColor(R, G, B);	Θέτει το χρώμα που θα εκτυπωθεί πίσω από τους χαρακτήρες. Μέχρι και τις αρχές του 2013 ήταν αναγκαίο, γιατί δεν υπήρχε “διαφανές” φόντο εκτύπωσης.
myGLCD.fillRect(x1, y1, x2, y2);	Σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο και το γεμίζει με χρώμα
myGLCD.drawRect(x1, y1, x2, y2);	Σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο

myGLCD.drawRoundRect(x1, y1, x2, y2);	Σχεδιάζει ένα ορθογώνιο παραλληλόγραμμο με καμπύλες γωνίες
myGLCD.drawLine(x1, y1, x2, y2);	Σχεδιάζει μια γραμμή
myGLCD.print(text, x, y, degrees);	Εκτυπώνει κείμενο. Διαθέτει και την επιλογή περιστροφής του κειμένου.
myGLCD.printNumI(integer, x, y);	Εκτυπώνει έναν ακέραιο αριθμό
myGLCD.printNumF(float, decimals, x, y);	Εκτυπώνει έναν αριθμό με δεκαδικά ψηφία, διαθέτοντας την επιλογή των πόσων δεκαδικών θα εκτυπωθούν

Πίνακας 6.3: Μερικές εντολές της βιβλιοθήκης UTFT

6.1.3.2 Η λειτουργία αφής

Η βιβλιοθήκη για τη λειτουργία αφής έχει αλλάξει πολλές φορες, συγκεκριμένα αυτή που συνοδεύει την εκάστοτε version της βιβλιοθήκης UTFT. Η τελευταία βιβλιοθήκη αφής που προσφέρεται από τον κατασκευαστή της οθόνης είναι η ArduCAM_Touch.

Η λειτουργία αφής χρησιμοποιεί το SPI και αποτελείται από αρκετά εύκολες εντολές για τη λειτουργία της. Μπορεί να καταστεί αρκετά πολύπλοκη η κατανόησή της, αλλά ένα πρόγραμμα μπορεί να υλοποιηθεί αρκετά εύκολα με λίγες απλές εντολές, ειδικά όταν πρόκειται για χρήση της λειτουργίας με κουμπιά κατά κύριο λόγο.

Η βιβλιοθήκη ορίζεται με την εντολή “ArduCAM_Touch myTouch(T_CS, T_IRQ);” όπου τα ορίσματα που επιλέγουμε είναι το Chip Select και το IRQ του Touch controller. Στην οθόνη μας το T_CS είναι by default το pin 10. Το IRQ είναι το pin 9 αλλά εμείς το αλλάξαμε στο pin 29. Το Touch αρχικοποιείται με την εντολή myTouch.InitTouch(); και στη συνέχεια έχουμε την επιλογή της ακρίβειας με την οποία θα διαβάζει τα δεδομένα ο controller.

Στον παρακάτω πίνακα παρατίθενται μερικές από τις κυριότερες εντολές της βιβλιοθήκης:

Εντολή	Ρόλος
myTouch.InitTouch();	Ξεκινά τη λειτουργία Touch. Δέχεται όρισμα 0 ή 1, τα οποία αντιστοιχούν σε PORTRAIT ή LANDSCAPE. Επιβάλλεται το όρισμα του Touch να είναι ίδιο με το όρισμα της οθόνης
myTouch.setPrecision();	Θέτει την ακρίβεια με την οποία θα διαβάζει τα δεδομένα ο controller. Η ακρίβεια μπορεί να τεθεί LOW, MEDIUM, HI, EXTREME, με την επιλογή MEDIUM ως default
myTouch.dataAvailable();	Επιστρέφει 'True' αν κάποιος αγγίζει την οθόνη
myTouch.read();	Διαβάζει τις συντεταγμένες στις οποίες έχει πατήσει ο χρήστης
myTouch.getX();	Επιστρέφει τη συντεταγμένη X στην οποία πάτησε ο χρήστης
myTouch.getY();	Επιστρέφει τη συντεταγμένη Y στην οποία πάτησε ο χρήστης

Πίνακας 6.4: Μερικές εντολές της βιβλιοθήκης ArduCAM_Touch

Για να οριστεί ένα κουμπί λοιπόν, ορίζουμε την περιοχή με τις εντολές UTFT για το σχήμα του κουμπιού και μετά με συγκρίσεις για να δούμε αν το σημείο επαφής βρίσκεται μέσα στην περιοχή του "κουμπιού" ορίζουμε τη λειτουργία του. Θα φανεί πιο ξεκάθαρα στο σχετικό κομμάτι κώδικα παρακάτω:

Κώδικας 6.2: Η κατασκευή ενός κουμπιού με βάση τη θέση που του έχει οριστεί σχηματικά

```
if (myTouch.dataAvailable())
  {
   myTouch.read();
   x=myTouch.getX();
   y=myTouch.getY();

   if ((y>=y1) && (y<=y2))  // Increase buttons
   {
    if ((x>=40) && (x<=80) && (settemp1<40))  // Client:1
    {
    settemp1 = settemp1 + 0.5;
    waitForIt(40, y1, 80, y2);
    }
   }
  }
```

Στον παραπάνω κώδικα λοιπόν βλέπουμε ότι αν οι συντεταγμένες αφής βρίσκονται μέσα στο τετράγωνο που ορίζεται από τα 40<x<80 και y1<y<y2, τότε μόνο καλείται η εντολή waitForIt(), η οποία επιτελεί την προσομοίωση του πατήματος του κουμπιού και θα αναλυθεί στο επόμενο κεφάλαιο.

6.1.4 Η προσομοίωση της λειτουργίας του Actuator

Όπως είπαμε σε προηγούμενο κεφάλαιο τον Actuator για μεγαλύτερη ευκολία τον προσομοιώσα στις δοκιμές με 3 led λειτουργίας. Αυτά μπορούμε να τα δούμε στην εικόνα 5.20. Το κίτρινο led, στη μέση συμβολίζει την ένδειξη λειτουργίας του actuator. Το κόκκινο led αντιστοιχεί σε "μείωση" και το πράσινο σε "αύξηση". Οι όροι "μείωση" και "αύξηση" αντιστοιχούν στη ροή θερμαντικού μέσου στα καλοριφέρ.

Όπως είδαμε στην ενότητα 5.2.5, ο χρόνος ανοίγματος του actuator είναι 3 λεπτά και ο χρόνος κλεισίματος είναι 5 λεπτά. Για να είναι πιο εύκολη η παρουσίαση της εργασίας, επιλέξαμε τα modules των σωμάτων να τρέχουν μια loop του προγράμματός τους ανά 5 δευτερόλεπτα. Η επανάληψη του προγράμματος θα μπορούσε να είναι 5 λεπτά ή οποιαδήποτε άλλη χρονική διάρκεια (σε πραγματικές συνθήκες). Τα 5 δευτερόλεπτα είναι αρκετά ώστε να φανεί σε συνθήκες δοκιμών η λειτουργία χωρίς να χρειάζεται μεγάλος χρόνος αναμονής. Μέσα σε αυτά τα 5 δευτερόλεπτα, επιλέξαμε η "λειτουργία" του actuator να διαρκεί 3 δευτερόλεπτα, τα οποία είναι μια συμβολική διάρκεια χρόνου.

Ο μικροελεγκτής του actuator λοιπόν, αφού συγκρίνει την πραγματική θερμοκρασία με την επιθυμητή, αποφασίζει αν χρειάζεται "άνοιγμα" ή "κλείσιμο" της ροής και στέλνει σήμα στον "actuator", ο οποίος στην προκειμένη τίθεται σε λειτουργία αύξησης ή μείωσης ροής για 3 δευτερόλεπτα. Κατά τη διάρκεια αυτών των τριών δευτερολέπτων το led αύξησης (πράσινο) ή μείωσης ροής (κόκκινο) παραμένει αναμμένο.

Αυτή η προσομοίωση έγινε με απλή χρήση των ψηφιακών εξόδων του μικροελεγκτή, στέλνοντας τάση στις εισόδους των leds, στις οποίες βέβαια παρεμβάλλονται αντιστάσεις 470Ω, οι οποίες για λόγους χώρου αποφασίστηκε να είναι SMD. Ο πλήρης κώδικας της διεργασίας θα παρατεθεί στο επόμενο κεφάλαιο.

6.1.5 Η σύνδεση RF δικτύου – Η/Υ

Η σύνδεση του RF δικτύου με τον Η/Υ γίνεται μέσω της πλακέτας με τον Arduino Nano. Ο ρόλος του μικροελεγκτή εδώ είναι να λαμβάνει τα δεδομένα από τον θερμοστάτη και να τα αποστέλει σειριακά στο Java Applet που τρέχει στη μεριά του Η/Υ. Είναι επίσης να λαμβάνει τα σειριακά δεδομένα που του στέλνει το Applet και να τα αποθηκεύει σε τιμές επιλογής θερμοκρασίας και να τα αποστέλλει στον θερμοστάτη, ο οποίος με τη σειρά του θα τα αποθηκεύσει και θα τα δρομολογήσει στα modules των σωμάτων

Η αποστολή των δεδομένων στον Η/Υ γίνεται απλά με μια εντολή Serial print, αφού στην ουσία η σειριακή έξοδος των δεδομένων από τον Nano εμφανίζεται στο Terminal που περιλαμβάνεται στο Applet. Να σημειωθεί ότι δεν είναι αναγκαίο να γίνει κάποια αποθήκευση αυτών των δεδομένων στον Η/Υ. Αντίθετα, η λήψη των δεδομένων από τον Η/Υ γίνεται με Serial read, όπου λαμβάνονται ξεχωριστά σε 6 διακριτά σειριακά μηνύματα από το PC τα ακέραια και τα δεκαδικά μέρη των δυο θερμοκρασιών που έχουμε θέσει από το Applet, καθώς και ενας delimiter και ένας χαρακτήρας λήξης αποστολής.

Κώδικας 6.3: Η σειριακή λήψη δεδομένων από το Applet

```
void readSerial(){
   ok = 0;
   if (Serial.available() >= 6){
   ak1 = Serial.read();
   dec1 = Serial.read();
   separator = Serial.read();
   ak2 = Serial.read();
   dec2 = Serial.read();
   space = Serial.read();
   settemp1 = ak1 + (dec1/100);
   settemp2 = ak2 + (dec2/100);
   readSerial();
   Serial.flush();
   ok = 1;
  }
}
```

Στον παραπάνω κώδικα βλέπουμε τη λήψη των δεδομένων από το Applet στον μικροελεγκτή μας. Ο μC περιμένει 6 bytes δεδομένων, με τη σειρά: ακέραιο μέρος θερμοκρασίας 1, δεκαδικό θερμοκρασίας 1, delimiter, ακέραιο μέρος θερμοκρασίας 2, δεκαδικό μέρος θερμοκρασίας 2 και χαρακτήρας λήξης. Όπως φαίνεται, ο μικροελεγκτής

διαβάζει τα δεδομένα μόνο αν έχουν ληφθεί όλα, τουλάχιστον δηλαδή 6 bytes. Αν έχουν ληφθεί παραπάνω bytes, για παράδειγμα αν ο χρήστης στον Η/Υ έχει αποστείλει σε σύντομο χρονικό διάστημα δυο ρυθμίσεις, ο μC θα αποβάλει τα πρώτα δεδομένα που θα λάβει και θα επαναλάβει τη διεργασία readSerial, για να αποθηκεύσει την πιο πρόσφατη ρύθμιση. Στο τέλος, αν υπάρχουν ακόμα δεδομένα στο buffer, τα οποία δεν είναι τουλάχιστον 6 bytes, αυτά απορρίπτονται με την εντολή serial.flush.

Η λήψη των δεδομένων από τον θερμοστάτη γίνεται σε ένα message που περιλαμβάνει τις τέσσερις θερμοκρασίες που έχει αποθηκευμένες ο θερμοστάτης. Το μήνυμα αυτό διαχωρίζεται σε τέσσερις double αριθμούς από τη διεργασία dataThermostat(), της οποίας ο κώδικας βρίσκεται στο Παράρτημα Α.

6.2 Ο έλεγχος από το PC

Για τον έλεγχο από το PC χρειαζόμασταν ένα πρόγραμμα το οποίο να επιτρέπει την επικοινωνία με κάποια συσκευή μέσω της σειριακής θύρας. Με τη Java μπορεί να χρησιμοποιηθεί η βιβλιοθήκη RXTXcomm, η οποία ήταν ακριβώς αυτό που χρειαζόμασταν και χρησιμοποιείται εκτενώς σε εφαρμογές που περιλαμβάνουν Arduino Boards και συσκευές ασύρματης επικοινωνίας, όπως για παράδειγμα οι Xbee.

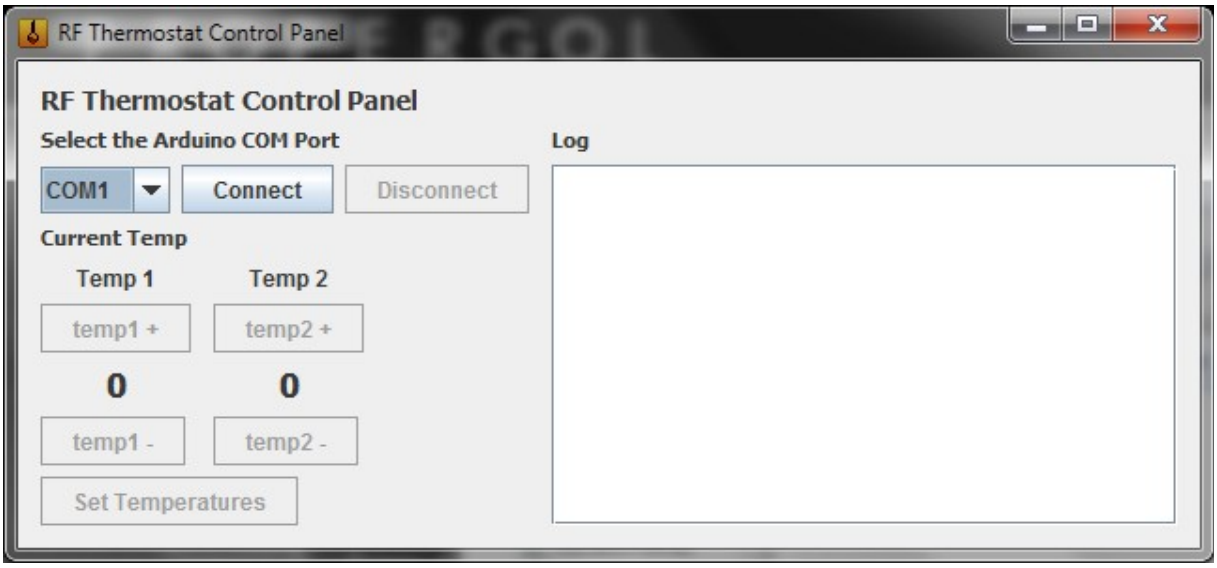
6.2.1 Το Java applet

Το Java Applet που κατασκευάσαμε βασίστηκε στο Applet TigerControlPanel, το οποίο χρησιμοποιούνταν για τον έλεγχο ενός τηλεκατευθυνόμενου Tiger Tank.

Το πρόγραμμα περιλαμβάνει ένα αρχείο με όνομα RfControlPanel, μέσα στο οποίο περιέχονται τρια αρχεία Java. Το κάθε αρχείο επιτελεί και διαφορετική λειτουργία. Το αρχείο GUI κατασκευάζει το γραφικό περιβάλλον. Το αρχείο Communicator διαχειρίζεται τις θύρες COM. Τέλος το αρχείο ActionList περιλαμβάνει τις διεργασίες που επιτελούνται από το πρόγραμμά μας, όπως για παράδειγμα την αυξομείωση των θερμοκρασιών ή τις δράσεις που προκαλούνται από το πάτημα των κουμπιών στο GUI. Στις επόμενες ενότητες θα αναλύσουμε κάποια βασικά στοιχεία του κώδικα και θα παρουσιάσουμε κάποιες εικόνες του Applet.

6.2.1.1 Το γραφικό περιβάλλον

Το γραφικό περιβάλλον περιλαμβάνει κουμπιά για τη σύνδεση και αποσύνδεση σε κάποια COM Port, κουμπιά για την αυξομείωση των θερμοκρασιών και την αποστολή των αλλαγών, καθώς και ένα Serial Terminal, που εμφανίζει τις τρέχουσες θερμοκρασίες όπως τις αποστέλλει σειριακά ο μικροελεγκτής που είναι συνδεδεμένος στον Η/Υ. Παρακάτω φαίνεται η εικόνα του προγράμματος κατά το άνοιγμά του, πριν συνδεθεί σε κάποια COM Port.



Εικόνα 6.1: Το Java Applet κατά το άνοιγμά του

Για την κατασκευή του γραφικού περιβάλλοντος χρησιμοποιήθηκε ένα αρχείο GUI.java, το οποίο είναι και το βασικό αρχείο του Applet. Στο αρχείο αυτό ορίζονται ως javax components όλα τα στοιχεία του περιβάλλοντος, τα κουμπιά δηλαδή, οι ετικέτες, το Terminal, καθώς και η διάταξή τους στο παράθυρο. Ο πλήρης κώδικας του GUI βρίσκεται στο Παράρτημα Α.

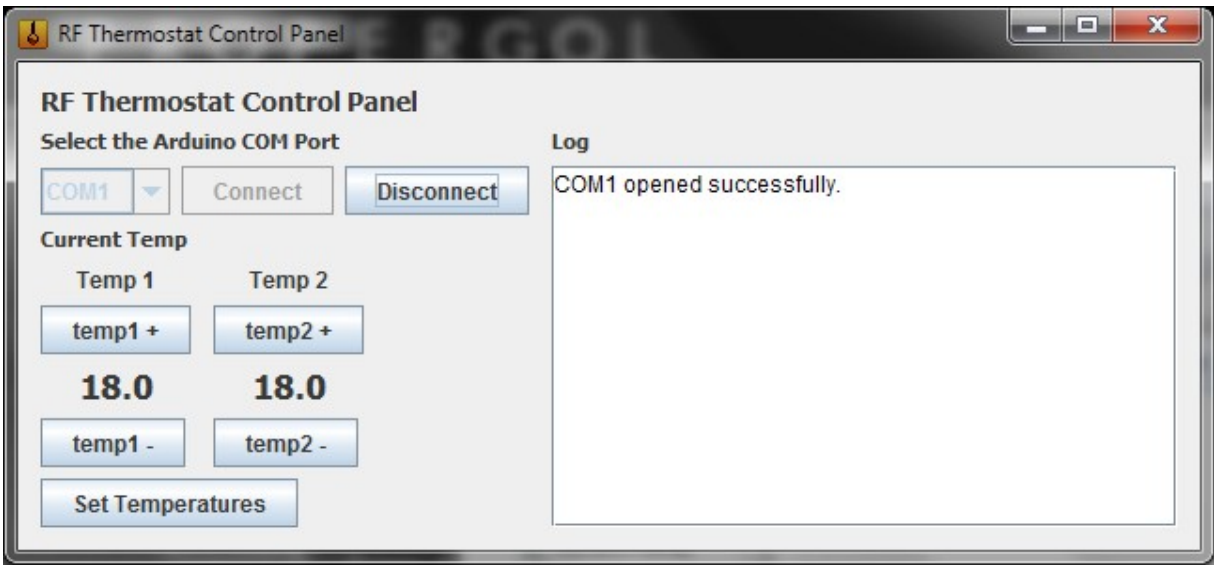
6.2.1.2 Η σειριακή επικοινωνία

Για τη σειριακή επικοινωνία χρησιμοποιήθηκε η βιβλιοθήκη RXTXComm. Στο πρόγραμμά μας κατασκευάσαμε ένα αρχείο Java το οποίο ονομάζεται Communicator και επιτελεί όλες τις λειτουργίες που έχουν σχέση με τη σειριακή επικοινωνία. Παρακάτω θα δούμε συνοπτικά μερικές από τις λειτουργίες που χρησιμοποιούνται.

Η function searchForPorts(); απαριθμεί όλες τις θύρες COM που έχουν συνδεδεμένη κάποια συσκευή και αποθηκεύει το όνομά τους σε έναν πίνακα. Η function connect(); ενεργοποιεί τη σύνδεση στην επιλεγμένη θύρα COM (που διαλέγουμε από το drop down menu). Στη συνέχεια το πρόγραμμα ανοίγει I/O streams για την επικοινωνία και περιλαμβάνει τη διεργασία writeData();, η οποία αποστέλλει στο output stream σειριακά τις τιμές των θερμοκρασιών που έχουμε θέσει μέσω του Η/Υ. Τέλος, υπαρχει και η διεργασία disconnect();, που ανταποκρίνεται στο πάτημα του κουμπιού disconnect και κλείνει τη σύνδεση με την COM Port. Ο πλήρης κώδικας του αρχείου Communicator.java βρίσκεται στο Παράρτημα Α.

6.2.1.3 Η λίστα ενεργειών

Η λίστα ενεργειών αποτελεί το τρίτο και τελευταίο αρχείο του προγράμματος, με όνομα ActionList.java. Σε αυτό το αρχείο βρίσκονται όλα τα actions που ανταποκρίνονται στο πάτημα ενός κουμπιού. Επίσης, περιλαμβάνεται και η διεργασία ToggleControls(), η οποία, μετά τη σύνδεση σε μια θύρα COM, ενεργοποιεί όλα τα κουμπιά (τα οποία, όπως φαίνεται στην εικόνα 6.1 είναι απενεργοποιημένα) και απενεργοποιεί την επιλογή "Connect". Παρακάτω εμφανίζεται μια εικόνα του Applet ακριβώς μετά τη σύνδεση με μια θύρα COM. Ο πλήρης κώδικας της Action List βρίσκεται στο Παράρτημα Α.



Εικόνα 6.2: Το Panel μετά τη σύνδεση σε κάποια θύρα COM.

Η αλλαγή των θερμοκρασιών γίνεται με τα κουμπιά αύξησης και μείωσης πάνω και κάτω αντίστοιχα από τις θερμοκρασίες και η αποστολή των επιλεγμένων θερμοκρασιών στον θερμοστάτη γίνεται με το πάτημα του κουμπιού "Set Temperatures".

6.2.1.4 Η μεταγλώττιση

Το πρόγραμμα αποτελείται από τρία αρχεία (τα οποία κατασκευάζουν κλάσεις) και μετά τη μεταγλώττιση θα δημιουργηθεί ένα αρχείο jar. Για να γίνει αυτό πρέπει οι κλάσεις να τοποθετηθούν σε έναν φάκελο, τον οποίο ονομάζουμε RfControlPanel. Στη συνέχεια κατασκευάζουμε το αρχείο manifest, στο οποίο ορίζουμε τη Main Class. Το περιεχόμενο του αρχείου manifest είναι η παρακάτω σειρά:

```
Main-Class: RfControlPanel.GUI
```

Για τη μεταγλώττιση των τριών αρχείων χρησιμοποιούμε την εντολή javac με κλήση όμως στο αρχείο RXTXComm.jar, ώστε να γνωρίζει η Java ότι θα συμπεριληφθεί το συγκεκριμένο jar στη classpath:

```
C:\path_to_folder\RfControlPanel> javac  -classpath  "c:\Program
Files\Java\jre6\lib\ext\RXTXcomm.jar"  GUI.java  Communicator.java  ActionList.java
```

Αφού κατασκευάσουμε τις κλάσεις, πρέπει να κατασκευάσουμε το αρχείο jar που θα χρησιμοποιείται για το τρέξιμο του Applet. Αυτό γίνεται με την παρακάτω εντολή:

```
C:\path_to_folder> jar -cvfm rf.jar  manifest.txt  RfControlPanel
```

όπου καλούνται με τη σειρά το όνομα του αρχείου jar που θα κατασκευαστεί, το αρχείο manifest, στο οποίο ορίζεται η main class και τέλος ο φάκελος στον οποίο βρίσκονται οι κλάσεις που κατασκευάσαμε.

Τέλος, για την έναρξη του προγράμματος καλούμε με εντολή Java το αρχείο rf.jar:

```
C:\path_to_folder> java  -jar  rf.jar
```

Να σημειωθεί ότι κατέστη δυνατό να προσθέσουμε ένα αρχείο icon στο πρόγραμμά μας, ώστε να εμφανίζεται η χαρακτηριστική εικόνα του Applet στο γραφικό περιβάλλον που το τρέχουμε. Αυτό έγινε με την εντολή

```
setIconImage(new ImageIcon("icon2.png").getImage());
```

Εικόνα 6.3: Το εικονίδιο του Applet

Στο παρακάτω κεφάλαιο θα παρουσιαστεί η λειτουργία των επιμέρους τμημάτων καθώς επικοινωνούν ανταλλάσσοντας δεδομένα και επιτελούν τις επιμέρους διεργασίες τους. Παράλληλα, θα παρατεθούν κάποια σημαντικά κομμάτια κώδικα, τα οποία θα επεξηγούνται. Κατά τη διάρκεια της λειτουργίας, να σημειωθεί ότι έχουμε ήδη θέσει διευθύνσεις στις συσκευές μας, οι οποίες έχουν αποθηκευτεί στις EEPROM των μικροελεγκτών. Ο θερμοστάτης μας έχει τη διεύθυνση 02, τα δυο modules και ο μικροελεγκτής του Η/Υ έχουν αντίστοιχα τις διευθύνσεις 012, 022 και 032.

7.1 Ο θερμοστάτης

Ο θερμοστάτης κατά την εκκίνησή του και αφού αρχικοποιήσει τις λειτουργίες του και τις μεταβλητές που θα χρησιμοποιήσει, ξεκινά τη λειτουργία της οθόνης, της αφής και της επικοινωνίας μέσω του RF δικτύου.

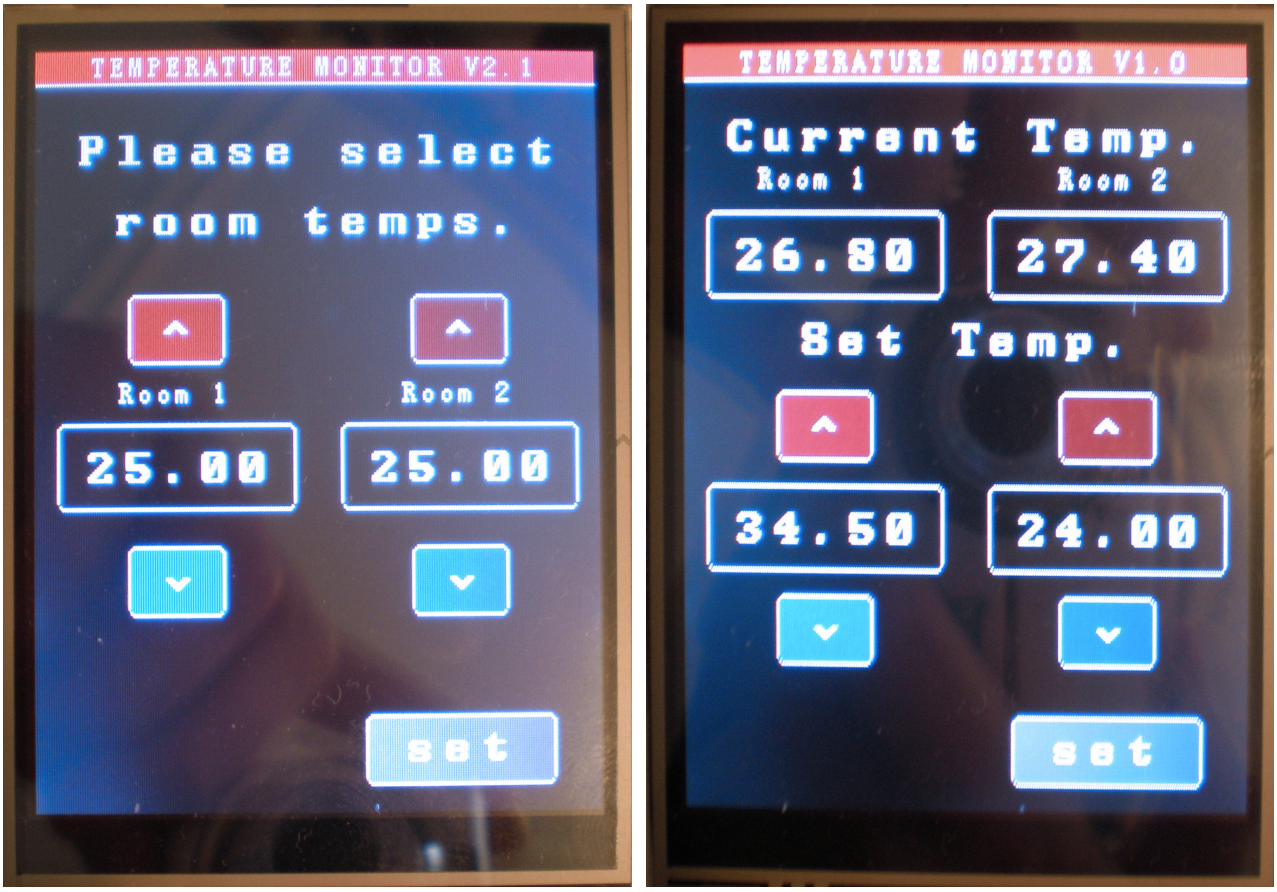
Μέσα στη διεργασία εκκίνησης (startup) φορτώνει την οθόνη εκκίνησης, στην οποία ο χρήστης επιλέγει τις επιθυμητές θερμοκρασίες για τα δυο δωμάτια, πριν ακόμα αρχίσει ο θερμοστάτης να επικοινωνεί με τα σώματα. Οι θερμοκρασίες κατά το άνοιγμα του θερμοστάτη βρίσκονται πάντα στη τιμή των 25°C.

Αφού ο χρήστης επιλέξει τις θερμοκρασίες που θέλει, πατά το πλήκτρο set, μέσω του οποίου μεταβαίνει η λειτουργία στη διεργασία επανάληψης (loop), στην οποία αρχικά ο θερμοστάτης λαμβάνει τις τιμές των θερμοκρασιών από τα σώματα και στη συνέχεια αποστέλλει στα σώματα (και στον Η/Υ) τις τιμές που έχει επιλέξει ο χρήστης.

Τέλος, ο χρήστης έχει τη δυνατότητα να επιλέξει τις θερμοκρασίες που θέλει ή να αφήσει τον θερμοστάτη να συνεχίσει την ανταλλαγή δεδομένων με τις υπόλοιπες συσκευές.

Στις παρακάτω δυο εικόνες εμφανίζεται το μενού εκκίνησης (αριστερά), όπου βλέπουμε ότι παρέχεται η δυνατότητα να μεταβληθούν οι επιθυμητές θερμοκρασίες, ξεκινώντας από την "φυσιολογική" θερμοκρασία των 25°C και να τεθούν πριν τη συνολική εκκίνηση της λειτουργίας της συσκευής. Στο μενού επανάληψης βλέπουμε ότι πλέον εμφανίζονται οι μετρήσεις των θερμοκρασιών που έχουν ληφθεί από τα modules των

σωμάτων, καθώς και ότι οι επιθυμητές θερμοκρασίες έχουν μεταβληθεί. Ο λόγος που η θερμοκρασία του πρώτου σώματος είναι τόσο υψηλή είναι για να φανεί αργότερα η λειτουργία "ανοίγματος" του "actuator", αφού η πραγματική θερμοκρασία είναι μικρότερη της επιθυμητής, σε αντίθεση με το δεύτερο σώμα, όπου η πραγματική θερμοκρασία είναι υψηλότερη από την επιθυμητή, που σημαίνει ότι ο "actuator" του δεύτερου module θα λειτουργήσει στη "θέση κλεισίματος".



Εικόνα 7.1: Οι οθόνες ρύθμισης και εμφάνισης του Θερμοστάτη

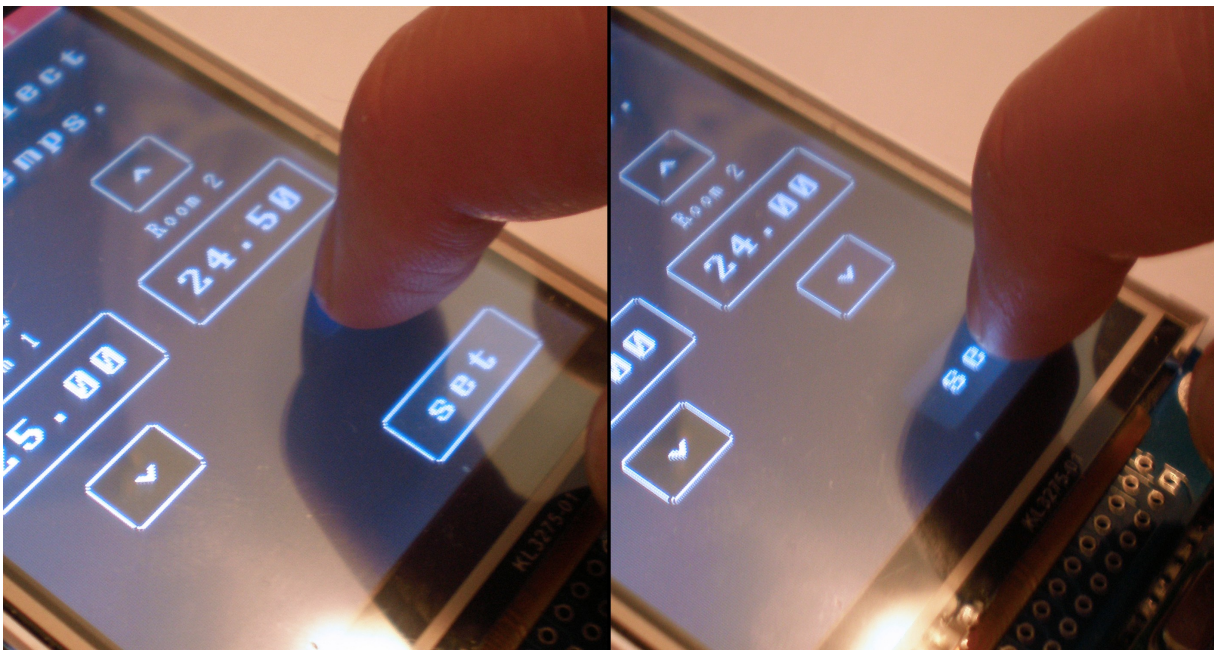
Αξίζει να σημειωθεί ότι οι θερμοκρασίες έχουν τεθεί να μεταβάλλονται με βήμα 0.5°C, χάριν λειτουργικότητας. Ένα μεγαλύτερο βήμα (της τάξης του 1°C) είναι αρκετά μεγάλο, ενώ ένα μικρότερο βήμα θα ήταν τόσο ακριβές που δε θα είχε ουσία. Αυτό επιτυγχάνεται θέτοντας στη λειτουργία "πατήματος" του κουμπιού την πρόσθεση και αφαίρεση αντίστοιχα μισού βαθμού κελσίου από την τρέχουσα επιλεγμένη θερμοκρασία. Επιπλέον, οι θερμοκρασίες έχουν τεθεί με όρια, από 18 εως 40°C. Τα όρια φυσικά θα έπρεπε να ξεκινούν από τους 0°C, δεδομένου ότι σε πολλές χώρες οι θερμοκρασίες πέφτουν μακράν χαμηλότερα των 0°C και χρειάζεται τα καλοριφέρ να λειτουργούν σε αυτή τη θερμοκρασία

για να αποτρέψουν το πάγωμα των σωληνώσεων. Επίσης, οι 40°C είναι πάρα πολλοί, αλλά δεδομένης της θερμοκρασίας του περιβάλλοντος την περίοδο της παρουσίασης, αν το όριο βρισκόταν στους 30°C, ίσως να ήταν δύσκολο να φανεί η λειτουργία "ανοίγματος" της ροής από τον "actuator", καθώς η θερμοκρασία του περιβάλλοντος, που θα λαμβάνουν τα modules των σωμάτων θα είναι της τάξης των 30°C.

Κώδικας 7.1 Αύξηση και μείωση της θερμοκρασίας για ένα module

```
if ((y>=y1) && (y<=y2))  // Increase buttons
    {
     if ((x>=40) && (x<=80) && (settemp1<40))  // Client:1
     {
     settemp1 = settemp1 + 0.5;
     waitForIt(40, y1, 80, y2);
     }
    }
    if ((y>=y3) && (y<=y4)) // Decrease buttons
    {
     if ((x>=40) && (x<=80) && (settemp1>18)) // Client: 1
     {
     settemp1 = settemp1 - 0.5;
     waitForIt(40, y3, 80, y4);
     }
    }
```

Στον παραπάνω κώδικα παρατηρούμε ότι, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, περιορίζουμε το πεδίο στο οποίο θα ενεργοποιείται η δυνατότητα αφής θέτοντας άνω και κάτω όρια για τις δυο συντεταγμένες. Αν αυτά τα όρια επικυρώνονται, τότε μεταβαίνουμε στη λειτουργεία "πατημένου κουμπιού", όπου η θερμοκρασία αυξάνεται ή μειώνεται αντίστοιχα και καλείται η διεργασία waitForIt, με τις συντεταγμένες του κουμπιού, η οποία θα αντικαταστήσει το λευκό περίγραμμα του κουμπιού με ένα γκρίζο, προσομοιώνοντας το εφέ σκοτεινιάσματος του πλαισίου, με τον ίδιο τρόπο δηλαδή που χρησιμοποιείται και από προγραμματιστές στα γραφικά περιβάλλοντα διαφόρων προγραμμάτων στους Η/Υ και άλλες συσκευές. Η waitForIt επίσης ενεργοποιεί μια loop η οποία διαρκεί όσο διατηρείται πατημένο το κουμπί, έτσι ώστε με ένα πάτημα να γίνεται μια αλλαγή μόνο και να μη μεταβάλλονται ραγδαία οι θερμοκρασίες θέσης.



Εικόνα 7.2: Το αποτέλεσμα της διεργασίας waitForIt

Στην παραπάνω εικόνα φαίνεται το πάτημα του κουμπιού μείωσης θερμοκρασίας του δωματίου 2, καθώς και του κουμπιού θέσης θερμοκρασιών στο μενού έναρξης. Παρατέθηκαν δυο φωτογραφίες ώστε να φαίνοντα τα κουμπιά και στην θέση ενεργοποίησης καθώς και στη θέση idle. Παρακάτω παρατίθεται ο κώδικας που υλοποιεί τη συγκεκριμένη λειτουργία.

Κώδικας 7.2 Προσομοίωση πατήματος κουμπιού

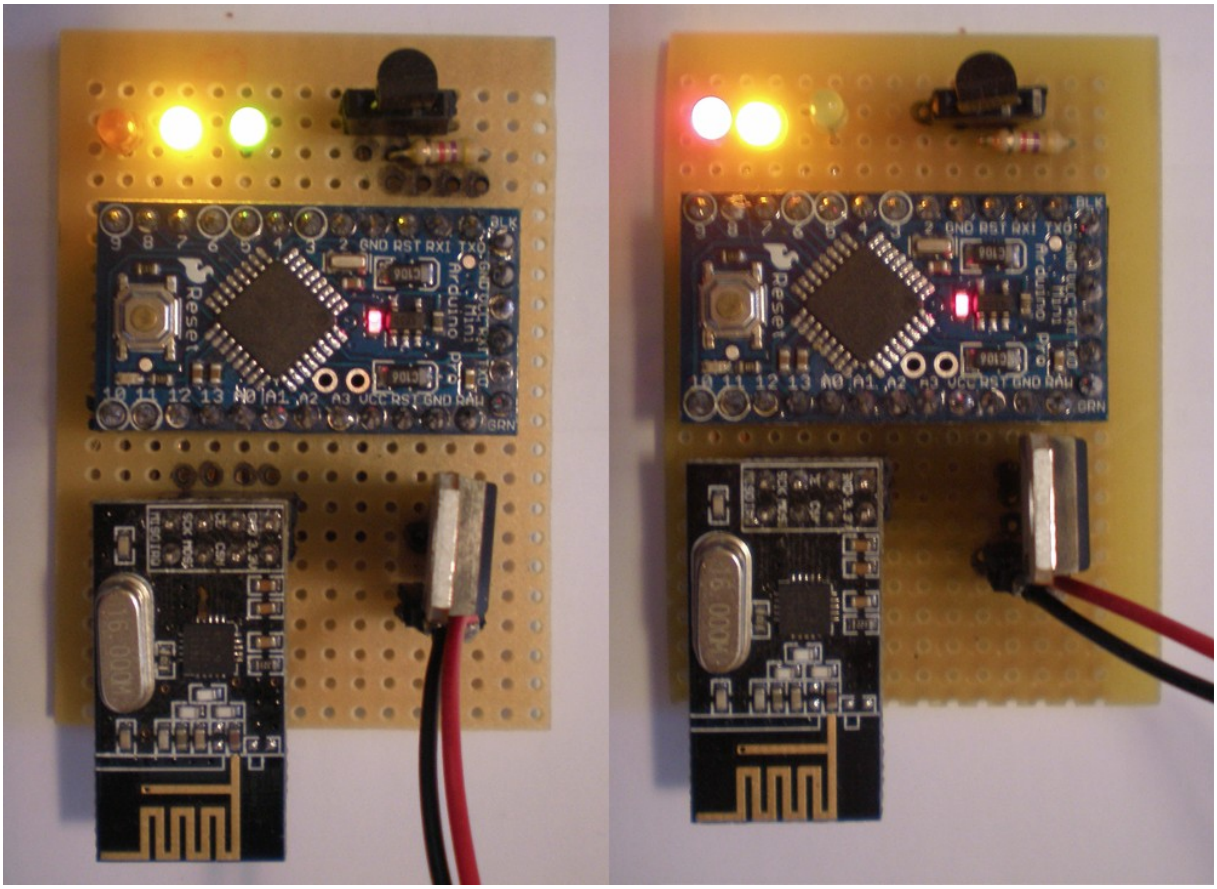
```
void waitForIt(int x1, int y1, int x2, int y2)
{
  myGLCD.setColor(155, 155, 155);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
  while (myTouch.dataAvailable())
    myTouch.read();
  myGLCD.setColor(255, 255, 255);
  myGLCD.drawRoundRect (x1, y1, x2, y2);
}
```

7.2 Τα modules των θερμαντικών σωμάτων

Στα modules των σωμάτων οι λειτουργίες που επιτελούνται είναι δυο, η μέτρηση των θερμοκρασιών, με τη βιβλιοθήκη OneWire, και η ρύθμιση της ροής με τα leds προσομοίωσης του actuator.

Κατά την εκκίνηση του module, αρχικοποιούνται τα instances των RF24 και RF24Network και τίθενται οι ακροδεκτες D6, D7 και D8 ως OUTPUT, αφού θα είναι οι έξοδοι του σήματος που θα τροφοδοτεί με τάση τα τρια leds. Το κίτρινο led, με όνομα ledOn ανάβει, συμφολίζοντας την ενεργοποίηση του actuator, ενώ τα άλλα δυο παραμένουν σβηστά.

Στη συνέχεια το module ψάχνει για πακέτα που να απευθύνονται σε αυτό, λαμβάνει τη θερμοκρασία από το DS18B20 (με τη διεργασία getTemp() που αναλύθηκε στην ενότητα 6.1.1) και την αποστέλλει στον θερμοστάτη. Τέλος, αφού έχει στη διάθεσή του την πραγματική τιμή της θερμοκρασίας και τη θερμοκρασία που έχει θέσει ο χρήστης, τις συγκρίνει και ενεργοποιεί μια από τις λειτουργίες ανοίγματος/κλεισίματος του actuator.



Εικόνα 7.3 Τα δυο modules σε λειτουργία αύξησης και μείωσης ροής αντίστοιχα

Η διεργασία ελέγχου των leds είναι πολύ απλή, περιλαμβάνοντας συγκρίσεις double αριθμών με ένα δεκαδικό ψηφίο. Η λειτουργία του “actuator” διαρκεί τρια δευτερόλεπτα, μετά το πέρας των οποίων ο “actuator” παραμένει idle μέχρι την επόμενη επανάληψη.

Κώδικας 7.3 Η κωδικοποίηση της λειτουργίας των leds

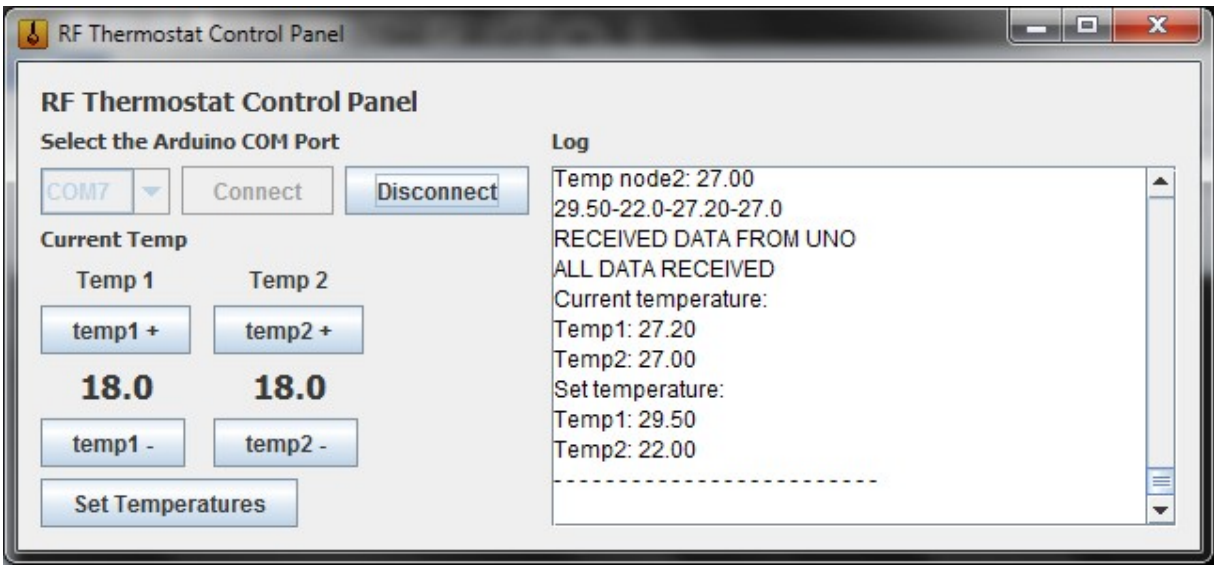
```
void ledActions()
{
 if ( (tempData != 0) && (tempSet != 0) )
 {
  if ( round(tempData) > round(tempSet) )
  {
   digitalWrite(ledDec, HIGH);
   delay(3000);
   digitalWrite(ledDec, LOW);
  }
  else if ( round(tempData) < round(tempSet) )
  {
   digitalWrite(ledInc, HIGH);
   delay(3000);
   digitalWrite(ledInc, LOW);
  }
  else {}
 }
}
```

7.3 Ο έλεγχος μέσω PC

Στην πλευρά του Η/Υ έχουμε δυο διαφορετικά κομμάτια που χρειάστηκαν προγραμματισμό, το Java Applet και τον μικροελεγκτή που μετατρέπει το σειριακό σήμα σε πακέτα RF.

Ο μικροελεγκτής λαμβάνει τα πακέτα που αποστέλλονται από τον θερμοστάτη και αποθηκεύει τις τιμές των θερμοκρασιών των σωμάτων και των θερμοκρασιών που έχουν επιλεγεί από τον χρήστη. Στη συνέχεια, εμφανίζει σειριακά τις θερμοκρασίες στο Log του Applet. Η λειτουργία αυτή επαναλαμβάνεται κάθε 5 δευτερόλεπτα, πάλι για λόγους ευκολίας στην παρουσίαση.

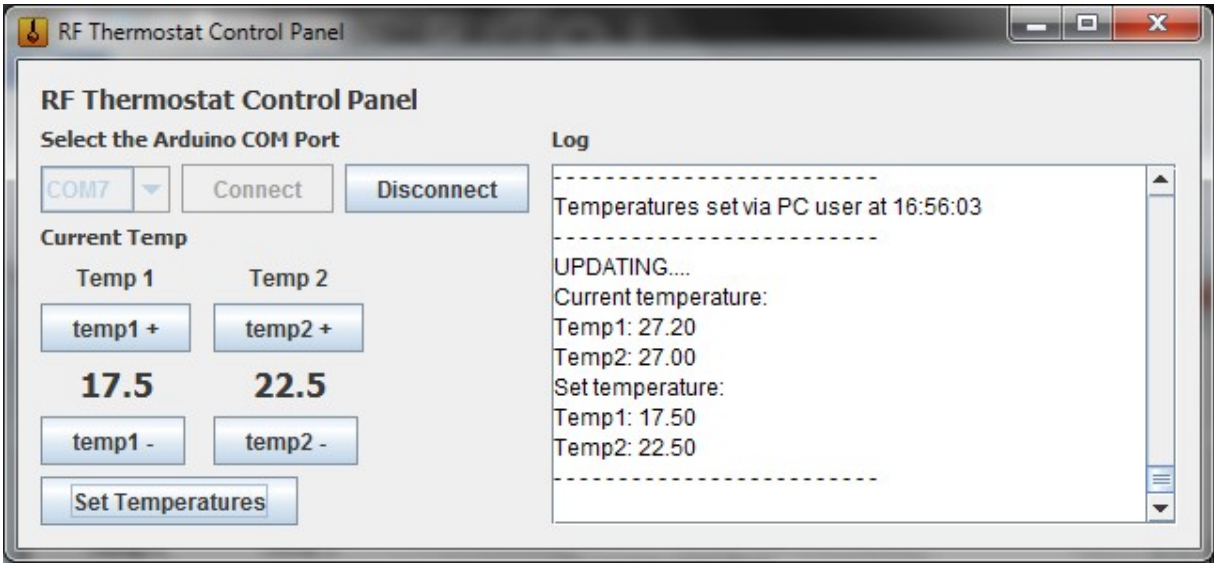
Το Applet έχει, όπως είπαμε κουμπιά ίδια με αυτά που έχει ο θερμοστάτης, για την αυξομείωση των θερμοκρασιών των σωμάτων στα δυο δωμάτια. Οι θερμοκρασίες δε θα αποσταλούν όμως παρά μόνο αν ο χρήστης πατήσει το κουμπί “Set Temperatures”.



Εικόνα 7.4 Η λειτουργία λήψης δεδομένων από τον θερμοστάτη

Στην παραπάνω εικόνα φαίνεται η λειτουργία κατά την οποία λαμβάνουμε τις θερμοκρασίες από τον θερμοστάτη. Στον κώδικα του μικροελεγκτή έχουν τοποθετηθεί μηνύματα για να κάνουν πιο ξεκάθαρη τη λειτουργία του συστήματος, όπως το μήνυμα ότι λάβαμε δεδομένα από τον θερμοστάτη.

Η αποστολή των δεδομένων συνοδεύεται από ένα μήνυμα που ενημερώνει τον χρήστη ότι οι θερμοκρασίες απεστάλησαν επιτυχώς, εκτυπώνοντας και την ακριβή ώρα συστήματος κατά την οποία τέθηκαν.



Εικόνα 7.5 Η λειτουργία αποστολής δεδομένων από τον Η/Υ

Όπως φαίνεται στην παραπάνω εικόνα, στο Log εμφανίζεται το μήνυμα θέσης θερμοκρασιών από τον Η/Υ, η ώρα καθώς και η επικύρωση από τον θερμοστάτη, με αποστολή των επιλεγμένων θερμοκρασιών από τον Η/Υ (17.5 και 22.5°C αντίστοιχα), όπως φαίνεται στο Panel ελέγχου των θερμοκρασιών και στο Log.

Συμπεράσματα

Η κατασκευή που παρουσιάσαμε αποτελεί μια σημαντική πρόταση και ιδέα για την ανάπτυξη ενός έξυπνου συστήματος εξοικονόμησης ενέργειας και βελτίωσης της άνεσης στον τομέα της θέρμανσης και της αυτονομίας. Έχουν ήδη υλοποιηθεί αρκετά διαφορετικά "έξυπνα" συστήματα θέρμανσης και θερμοστάτες, όμως η αγορά δεν έχει διευρυνθεί αρκετά και συστήματα όπως το συγκεκριμένο δεν έχουν κάνει την εμφάνισή τους στην καθημερινότητά μας. Η κατασκεύη αυτή αποδεικνύει ότι τέτοια συστήματα όχι μόνο είναι εφικτά άλλά μπορούν άμεσα και με χαμηλό κόστος να υλοποιηθούν, αντικαθιστώντας τα συστήματα παλαιότερης γενιάς, και να συνδυάσουν την αύξηση της άνεσης, την μείωση του κόστους θέρμανσης και την βελτίωση του περιβαλλοντικού αποτυπώματος.

[1] ManiacBug: Getting Started with NRF24L01+ on Arduino. Available at: http://maniacbug.wordpress.com/2011/11/02/getting-started-rf24/

[2] Arduino Playground nRF24L01 Documentation. Available at: http://playground.arduino.cc/InterfacingWithHardware/Nrf24L01

[3] Nordic Semiconductor nRF24L01 Ultra low power 2.4GHz RF Transceiver. Available at: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01

[4] Arduino Info: nRF24L01 Mirf Examples/Mirf Library: nRF24L01 Test. Available at: http://arduino-info.wikispaces.com/nRF24L01-Mirf-Examples

[5] Bildr: One Wire Digital Temperature – DS18B20 + Arduino . Available at: http://bildr.org/2011/07/ds18b20-arduino/

[6] Maxim Integrated DS18B20 One wire thermometer Datasheet. Available at: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[7] Instructables: Arduino Examples #2 Use an Arduino as a FTDI Programmer. Available at: www.instructables.com/id/Arduino-Examples-2-Use-an-Arduino-as-a-FTDI-Progr/

[8] Arduino.cc Forum: How to program a Pro Mini with another Arduino. Available at: http://forum.arduino.cc/index.php/topic,7571.0.html

[9] UCTronics 3.2” TFT LCD Shield + Touch Panel for Arduino User Guide. Available at: http://www.uctronics.com/download/Arduino_shileds/3inch2_Arduino_LCD_Shield_RevB_UG.pdf

[10] Henning Karlsen Electronics: UTFT Library. Available at: http://www.henningkarlsen.com/electronics/library.php?id=51

[11] Henning Karlsen Electronics: Modifying UTFT to accept Arduino shields on Arduino Mega. Available at: www.henningkarlsen.com/electronics/h_utft_arduino_shield_on_mega.php

[12] Arduino Forum: 3.2” TFT Shield – White, Flickering, Display Problem on UNO. Available at: http://forum.arduino.cc/index.php?topic=134581.0

[13] Lincomatic's Blog: UCTronics 3.2” TFT LCD Arduino Shield with Touchscreen. Available at: http://blog.lincomatic.com/?p=1074

[14] Arduino Forum: 3.2” Sainsmart TFT Touch Issues. Available at: http://forum.arduino.cc/index.php?topic=121017.0

[15] ManiacBug: Arduino on ATMega1284P. Available at: http://maniacbug.wordpress.com/2011/11/27/arduino-on-atmega1284p-4/

[16] OT-Hobbies: Arduino & ATMega1284P Problem Issues. Available at: http://www.ot-hobbies.com/resource/ard-1284.htm

[17] Arduino Forum: ATMega1284P End to End using 1.0 IDE. Available at: http://forum.arduino.cc/index.php?topic=80483.0

[18] Arduino Forum: Using the 1284p/664p (IDE, Breadboard and Bootloaders). Available at: http://forum.arduino.cc/index.php/topic,64612.0.html

[19] Gammon: How to make an Arduino-compatible minimal board. Available at: http://www.gammon.com.au/forum/?id=11637

[20] Arduino: SPI Library. Available at: http://arduino.cc/en/Reference/SPI

[21] Stack Overflow: Managing two SPI Devices at once. Available at: http://stackoverflow.com/questions/15304919/arduino-managing-two-spi-devices-at-once

[22] Henry Poon: Serial Communication in Java with Example Program. Available at: http://henrypoon.wordpress.com/2011/01/01/serial-communication-in-java-with-example-program/

[23] Java2s: Read from a Serial Port, notifying when data arrives. Available at: http://www.java2s.com/Code/Java/Development-Class/ReadfromaSerialportnotifyingwhendataarrives.htm

[24] Wikibooks: Serial Java. Available at: http://en.wikibooks.org/wiki/Serial_Programming/Serial_Java

[25] RXTX: Using RXTX. Available at: http://rxtx.qbang.org/wiki/index.php/Using_RXTX

[26] Arduino Forum: Installing Java to use RXTX. Available at: http://forum.arduino.cc/index.php?topic=128937.0

[27] Embedded Freaks: How to open a serial port using RXTX. Available at: http://embeddedfreak.wordpress.com/2008/08/08/how-to-open-serial-port-using-rxtx/

[28] Skylit: Creating an Executable jar File. Available at: http://www.skylit.com/javamethods/faqs/createjar.html

[29] FortyStones: Creating a Simple GUI for Absolute Beginners. Available at: http://www.fortystones.com/creating-simple-gui-beginners-java-tutorial/
FortyStones: Event Handlers in Java. Available at: http://www.fortystones.com/event-

[30] handlers-java/

[31] Solin S.A: Θέρμανση και συστήματα μεταφοράς. Available at: http://www.solin.gr/index.php?option=com_content&view=article&id=13&Itemid=14%E2%8C%A9=en

[32] Ecowiz: Κατάργηση Ωρομετρητών Αυτονομίας. Available at: http://www.ecowiz.gr/page12.html

[33] Multitherm: Θερμιδομέτρηση. Available at: http://www.multitherm.gr/index.php?option=com_content&view=article&id=20

[34] Techgear: Nest, ο έξυπνος θερμοστάτης. Available at: http://www.techgear.gr/nest-the-learning-thermostat-32380/

[35] All About Energy: Περιβάλλον και διαχείριση ενέργειας. Παραγωγή ενέργειας, βασικός εξοπλισμός: Λέβητες. Available at: http://www.allaboutenergy.gr/Levites.html

Στο Παράρτημα αυτό παρατίθενται οι κώδικες που χρησιμοποιήθηκαν για τις τρεις διακριτές συσκευές που απαρτίζουν την κατασκευή, μαζί με τα header και cpp files που περιλαμβάνουν. Τα αρχεία αυτά παρατίθενται στο τέλος του κάθε κώδικα. Στους κώδικες περιλαμβάνονται εκτενή σχόλια για την πλήρη κατανόηση του κάθε κομματιού κώδικα που χρησιμοποιείται.

Α.1 Ο θερμοστάτης

Κώδικας Α.1 Το βασικό αρχείο κώδικα του θερμοστάτη

```
////////////////////////////////////////////////////////////////////////////////
// This program contains procedures for the functions of the //
// LCD screen and the communication of the NRF modules  //
////////////////////////////////////////////////////////////////////////////////

#include <avr/pgmspace.h>
#include <RF24Network.h>
#include <RF24.h>
#include <UTFT.h>
#include <ArduCAM_Touch.h>
#include <SD.h>
#include <SPI.h>
#include "nodeconfig.h"
#include "ftoa.h"

//////////////////////////////////////
// RF24 Radio + Network  //
//////////////////////////////////////
#ifndef PINS_DEFINED
#define __PLATFORM__ "Getting Started board"

// Pins for radio
const int rf_ce = 28;
const int rf_csn = A5;

#endif

// Starting up the RF24 radio and network
RF24 radio(rf_ce,rf_csn);
RF24Network network(radio);
```

```
// Defining our node address
uint16_t this_node;

////////////////////////////////////////
// UTFT + Touch               //
////////////////////////////////////////

UTFT       myGLCD(ITDB32S,A1,A2,A0,A3);
ArduCAM_Touch  myTouch(10,29);

// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];


////////////////////////////////////////////////////////////
// Variables for communication with nodes //
////////////////////////////////////////////////////////////

char message[24];
double tempFromNode1 = 00.00;
double tempFromNode2 = 00.00;
double settemp1 = 25.00;
double settemp2 = 25.00;

//Touch coordinates
int x, y;


////////////////////////////////////////////////
// Welcome screen                    //
////////////////////////////////////////////////
void welcome()
{
        //Setting the layout of the welcome screen
        myGLCD.setFont(SmallFont);
        myGLCD.setColor(0, 0, 0);
        myGLCD.fillRect(0, 0, 239, 359);
        myGLCD.setColor(255, 0, 0);
        myGLCD.fillRect(0, 0, 239, 13);
        myGLCD.setColor(255, 255, 255);
        myGLCD.setBackColor(255, 0, 0);
        myGLCD.drawLine(0, 14, 239, 14);
        myGLCD.print("TEMPERATURE MONITOR V2.1", CENTER, 1);
        myGLCD.setBackColor(0, 0, 0);
        myGLCD.print("Room 1         Room 2", CENTER, 140);
        myGLCD.setFont(BigFont);
        myGLCD.print("Please select", CENTER, 35);
        myGLCD.print("room temps.", CENTER, 65);
```

```
        // Increase Temperature Buttons
        myGLCD.setColor(155, 0, 0);
        myGLCD.fillRoundRect (40, 105, 80, 133);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (40, 105, 80, 133);
        myGLCD.setColor(155, 0, 0);
        myGLCD.fillRoundRect (160, 105, 200, 133);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (160, 105, 200, 133);
        myGLCD.setBackColor(155, 0, 0);
        myGLCD.print("^", 52, 115);
        myGLCD.print("^", 172, 115);

        // Print Selected Temperature
        myGLCD.drawRoundRect (10, 159, 110, 194);
        myGLCD.drawRoundRect (130, 159, 230, 194);
        myGLCD.setBackColor(0, 0, 0);
        myGLCD.printNumF(settemp1, 2, 20, 169);
        myGLCD.printNumF(settemp2, 2, 140, 169);

        //Decrease Temperature Buttons
        myGLCD.setBackColor(0, 0, 155);
        myGLCD.setColor(0, 0, 155);
        myGLCD.fillRoundRect (40, 210, 80, 238);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (40, 210, 80, 238);
        myGLCD.setColor(0, 0, 155);
        myGLCD.fillRoundRect (160, 210, 200, 238);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (160, 210, 200, 238);
        myGLCD.print("^", 68, 229, 180);
        myGLCD.print("^", 188, 229, 180);

        // Set Button
        myGLCD.setBackColor(155, 155, 155);
        myGLCD.setColor(155, 155, 155);
        myGLCD.fillRoundRect (140, 280, 220, 308);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (140, 280, 220, 308);
        myGLCD.print("set", 155, 285);

        myGLCD.setBackColor(0, 0, 0);
}
```

```
//////////////////////////////////////
// Function screen              //
//////////////////////////////////////
void drawButtons()
{
        // Setting the layout of the normal function screen
        myGLCD.setFont(SmallFont);
        myGLCD.setColor(0, 0, 0);
        myGLCD.fillRect(0, 0, 239, 359);
        myGLCD.setColor(255, 0, 0);
        myGLCD.fillRect(0, 0, 239, 13);
        myGLCD.setColor(255, 255, 255);
        myGLCD.setBackColor(255, 0, 0);
        myGLCD.drawLine(0, 14, 239, 14);
        myGLCD.print("TEMPERATURE MONITOR V2.1", CENTER, 1);
        myGLCD.setBackColor(0, 0, 0);
        myGLCD.print("Room 1          Room 2", CENTER, 50);
        myGLCD.setFont(BigFont);
        myGLCD.print("Current Temp.", CENTER, 30);
        myGLCD.print("Set Temp.", CENTER, 115);

        // Print the temperature received from nodes
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (10, 70, 110, 105);
        myGLCD.drawRoundRect (130, 70, 230, 105);

        // Increase Temperature Buttons
        myGLCD.setColor(155, 0, 0);
        myGLCD.fillRoundRect (40, 145, 80, 173);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (40, 145, 80, 173);
        myGLCD.setColor(155, 0, 0);
        myGLCD.fillRoundRect (160, 145, 200, 173);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (160, 145, 200, 173);
        myGLCD.setBackColor(155, 0, 0);
        myGLCD.print("^", 52, 155);
        myGLCD.print("^", 172, 155);

        // Print Selected Temperature
        myGLCD.drawRoundRect (10, 184, 110, 219);
        myGLCD.drawRoundRect (130, 184, 230, 219);
        myGLCD.setBackColor(0, 0, 0);
        myGLCD.printNumF(settemp1, 2, 20, 194);
        myGLCD.printNumF(settemp2, 2, 140, 194);

        //Decrease Temperature Buttons
        myGLCD.setBackColor(0, 0, 155);
        myGLCD.setColor(0, 0, 155);
        myGLCD.fillRoundRect (40, 230, 80, 258);
```

```
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (40, 230, 80, 258);
        myGLCD.setColor(0, 0, 155);
        myGLCD.fillRoundRect (160, 230, 200, 258);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (160, 230, 200, 258);
        myGLCD.print("^", 68, 249, 180);
        myGLCD.print("^", 188, 249, 180);

        // Set Button
        myGLCD.setBackColor(155, 155, 155);
        myGLCD.setColor(155, 155, 155);
        myGLCD.fillRoundRect (140, 280, 220, 308);
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (140, 280, 220, 308);
        myGLCD.print("set", 155, 285);

        myGLCD.setBackColor(0, 0, 0);
}

////////////////////////////////
// Edit temps             //
////////////////////////////////
void touchMenu(int y1, int y2, int y3, int y4, int y5, int y6, int ytext)
{
        // Enter constant loop until break
        while (true)
        {
                if (myTouch.dataAvailable())
                {
                        myTouch.read();
                        x=myTouch.getX();
                        y=myTouch.getY();

                        if ((y>=y1) && (y<=y2))  // Increase buttons
                        {
                                if ((x>=40) && (x<=80) && (settemp1<40))  // Room:1
                                {
                                        // Increase temperature by 0.5 C and call push button func
                                        settemp1 = settemp1 + 0.5;
                                        waitForIt(40, y1, 80, y2);
                                }
                                if ((x>=160) && (x<=200) && (settemp2<40)) // Room: 2
                                {
                                        settemp2 = settemp2 + 0.5;
                                        waitForIt(160, y1, 200, y2);
                                }
                        }
```

```
                        if ((y>=y3) && (y<=y4)) // Decrease buttons
                        {
                                if ((x>=40) && (x<=80) && (settemp1>18)) // Room: 1
                                {
                                        settemp1 = settemp1 - 0.5;
                                        waitForIt(40, y3, 80, y4);
                                }
                                if ((x>=160) && (x<=200) && (settemp2>18)) // Room: 2
                                {
                                        settemp2 = settemp2 - 0.5;
                                        waitForIt(160, y3, 200, y4);
                                }
                        }
                        if ((y>=y5) && (y<=y6) && (x>=140) && (x<=220))
                        {
                                // When set is pressed, leave while loop
                                waitForIt(140, y5, 220, y6);
                                break;
                        }
                }
                // Print temperatures in screen
                myGLCD.printNumF(settemp1, 2, 20, ytext);
                myGLCD.printNumF(settemp2, 2, 140, ytext);
        }
}


//////////////////////////////////////////////////////////////////////
// Draw a grey frame while a button is touched //
//////////////////////////////////////////////////////////////////////
void waitForIt(int x1, int y1, int x2, int y2)
{
        myGLCD.setColor(155, 155, 155);
        myGLCD.drawRoundRect (x1, y1, x2, y2);
        while (myTouch.dataAvailable())
                myTouch.read();
        myGLCD.setColor(255, 255, 255);
        myGLCD.drawRoundRect (x1, y1, x2, y2);
}


//////////////////////////////////////////////////////////////////////////
// Take message from nano & split into two temps //
//////////////////////////////////////////////////////////////////////////
void dataThermostat(char *mes)
{
        // Split received message into two messages and convert to double with one decimal
        char *temp1;
        char *temp2;
```

```
        // Save the contents of a message in a buffer string
        String msgbuf(mes);
        temp1 = strtok(mes,"-");
        temp2 = strtok(NULL,"-");
        settemp1 = onedecimal ( atof (temp1));
        settemp2 = onedecimal ( atof (temp2));
        // Return the contents of the message from the buffer to the message
        msgbuf.toCharArray(mes,12);
}


//////////////////////////////////////////////////
// RF24Network procedures              //
//////////////////////////////////////////////////
void radioTransceive()
{
        // Pump the network regularly
        network.update();
        // Is there anything ready for us?
        while ( network.available() )
        {
                // If so, grab it
                RF24NetworkHeader header;
                char message[24];
                network.read(header,&message,sizeof(message));
                // Find who sent us the package, save it and print it
                if(header.from_node == 032)
                {
                        dataThermostat(message);
                        myGLCD.printNumF(settemp1, 2, 20, 194);
                        myGLCD.printNumF(settemp2, 2, 140, 194);
                }
                else if (header.from_node == 012)
                {
                        tempFromNode1 = onedecimal( atof ( message ));
                        myGLCD.printNumF(tempFromNode1, 2, 20, 80);
                }
                else if (header.from_node == 022)
                {
                        tempFromNode2 = onedecimal( atof ( message ));
                        myGLCD.printNumF(tempFromNode2, 2, 140, 80);
                }
        }
        // Stop listening
        // Send data to nodes
        // Send all temperatures to PC in a message
        RF24NetworkHeader header(032);
        floatsToMsg(message, settemp1, settemp2, tempFromNode1, tempFromNode2, 2);
        network.write(header,&message,sizeof(message));
```

```
	// Send temperature selection to radiator modules
	RF24NetworkHeader header02(012);
	ftoa(message, settemp1, 2);
	network.write(header02,&message,sizeof(message));

	RF24NetworkHeader header05(022);
	ftoa(message, settemp2, 2);
	network.write(header05,&message,sizeof(message));
}


//////////////////////////////////////////////////
// SETUP + LOOP                                 //
//////////////////////////////////////////////////
void setup(void)
{
	// Bring up LCD and Touch
	myGLCD.InitLCD(PORTRAIT);
	myGLCD.clrScr();

	myTouch.InitTouch(PORTRAIT);
	myTouch.setPrecision(PREC_LOW);

	// Load startup menu, set temperatures and load standard menu
	welcome();
	touchMenu(105, 133, 210, 238, 280, 308, 169);
	myGLCD.clrScr();
	drawButtons();


	// Pull node addr out of eeprom & bring up network
	this_node = nodeconfig_read();
	SPI.begin();
	radio.begin();
	network.begin(/*channel*/ 92, /*node address*/ this_node);
}



void loop(void)
{
	radioTransceive();
	int timer;
	// Use a timer of almost 5 seconds to wait for the user to select temperatures
	// or else run loop again and receive new temperatures from nodes
	for (timer=50;timer!=0;timer--)
	{
		if (myTouch.dataAvailable())
		{
			myTouch.read();
```

```
                    x=myTouch.getX();
                    y=myTouch.getY();

                    if ((y>=145) && (y<=173))  // Increase buttons
                    {
                        if ((x>=40) && (x<=80) && (settemp1<40))  // Room:1
                        {
                            settemp1 = settemp1 + 0.5;
                            waitForIt(40, 145, 80, 173);
                        }
                        if ((x>=160) && (x<=200) && (settemp2<40)) // Room: 2
                        {
                            settemp2 = settemp2 + 0.5;
                            waitForIt(160, 145, 200, 173);
                        }
                    }
                    if ((y>=230) && (y<=258)) // Decrease buttons
                    {
                        if ((x>=40) && (x<=80) && (settemp1>18)) // Room: 1
                        {
                            settemp1 = settemp1 - 0.5;
                            waitForIt(40, 230, 80, 258);
                        }
                        if ((x>=160) && (x<=200) && (settemp2>18)) // Room: 2
                        {
                            settemp2 = settemp2 - 0.5;
                            waitForIt(160, 230, 200, 258);
                        }
                    }
                    if ((y>=280) && (y<=308) && (x>=140) && (x<=220))
                    {
                        waitForIt(140, 280, 220, 308);
                        break;
                    }
                }
        //Print temperatures in screen
        myGLCD.printNumF(settemp1, 2, 20, 194);
        myGLCD.printNumF(settemp2, 2, 140, 194);
        myGLCD.printNumI(timer, 45, 285);
        }

// Listen for a new node address if the user sends serial data to the uC
//nodeconfig_listen();
}
```

```
// Convert float number to string
char *ftoa(char *a, double f, int precision)
{
	long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};
	char *ret = a;
	long heiltal = (long)f;
	itoa(heiltal, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal = abs((long)((f - heiltal) * p[precision]));
	itoa(desimal, a, 10);
	return ret;
}

// Cut the second decimal off the double number
double onedecimal(double f)
{
	double minus = 10*f;
	int cut = minus;
	double buf = cut;
	double ret = buf/10;
	return ret;
}

// Add all temperature numbers to a string to send to the PC
char *floatsToMsg(char *a, double f, double d, double t1, double t2, int precision)
{
	long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

	char *ret = a;
	long heiltal = (long)f;
	itoa(heiltal, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal = abs((long)((f - heiltal) * p[precision]));
	itoa(desimal, a, 10);
	while (*a != '\0') a++;
	*a++ = '-';
	long heiltal2 = (long)d;
	itoa(heiltal2, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal2 = abs((long)((d - heiltal2) * p[precision]));
	itoa(desimal2, a, 10);
	while (*a != '\0') a++;
	*a++ = '-';
	long heiltal3 = (long)t1;
```

```
        itoa(heiltal3, a, 10);
        while (*a != '\0') a++;
        *a++ = '.';
        long desimal3 = abs((long)((t1 - heiltal3) * p[precision]));
        itoa(desimal3, a, 10);
        while (*a != '\0') a++;
        *a++ = '-';
        long heiltal4 = (long)t2;
        itoa(heiltal4, a, 10);
        while (*a != '\0') a++;
        *a++ = '.';
        long desimal4 = abs((long)((t2 - heiltal4) * p[precision]));
        itoa(desimal4, a, 10);
        return ret;
}
```

A.2 Τα modules των σωμάτων

Κώδικας Α.3 Το βασικό αρχείο κώδικα των modules

```
////////////////////////////////////////////////////////////////////////////////////
// This program contains procedures for temperature reading   //
// and transmission of the temperatures via RF24Network       //
////////////////////////////////////////////////////////////////////////////////////
#include <avr/pgmspace.h>
#include <RF24Network.h>
#include <RF24.h>
#include <SPI.h>
#include <OneWire.h>
#include <Tictocs.h>
#include <TictocTimer.h>
#include "nodeconfig.h"
#include "printf.h"
#include "getTemp.h"
#include "ftoa.h"

/////////////////////////////////////////
// RF24 Radio + Network  //
/////////////////////////////////////////
#ifndef PINS_DEFINED
#define __PLATFORM__ "Getting Started board"

const int rf_ce = 9;
const int rf_csn = 10;

#endif

RF24 radio(rf_ce,rf_csn);
RF24Network network(radio);

// Our node address
uint16_t this_node;

// Actuator – Led Definitions
int ledOn = 7;
int ledInc = 6;
int ledDec = 8;

//Temperature variables
double tempData;
double tempSet;
```

```
//////////////////////////
// LED functions  //
//////////////////////////
void ledActions()
{
	if ( (tempData != 0) && (tempSet != 0) )
	{
		if ( round(tempData) > round(tempSet) )
		{
			digitalWrite(ledDec, HIGH);
			delay(3000);
			digitalWrite(ledDec, LOW);
		}
		else if ( round(tempData) < round(tempSet) )
		{
			digitalWrite(ledInc, HIGH);
			delay(3000);
			digitalWrite(ledInc, LOW);
		}
		else {}
	}
}


void setup(void)
{
	Serial.begin(57600);
	printf_begin();
	printf_P(PSTR("\n\rRF24Network/examples/sensornet/\n\r"));
	printf_P(PSTR("PLATFORM: " __PLATFORM__ "\n\r"),program_version);
	printf_P(PSTR("VERSION: %s\n\r"),program_version);

	// Pull node addr out of eeprom & bring up network
	this_node = nodeconfig_read();
	SPI.begin();
	radio.begin();
	network.begin(/*channel*/ 92, /*node address*/ this_node);

	// Initialize led pins and light up function led
	pinMode(ledOn, OUTPUT);
	pinMode(ledInc, OUTPUT);
	pinMode(ledDec, OUTPUT);
	digitalWrite(ledOn, HIGH);
}
```

```
void loop(void)
{
	// Pump the network regularly
	network.update();
	// Is there anything ready for us?
	while ( network.available() )
	{
		// If so, grab it and save it as double
		RF24NetworkHeader header;
		char message[6];
		network.read(header,&message,sizeof(message));
		tempSet = atof ( message );

		// Printing messages for serial verification if module is connected to PC
		//printf_P(PSTR("%lu: APP Received #%u %s from
		0%o\n\r"),millis(),header.id,message,header.from_node);
		//    printf_P(PSTR("Temperature 0%o: "), header.from_node);
		//    Serial.println(message);
		//    Serial.println(tempSet);
	}

	// Get temperature reading
	tempData = getTemp();
	char message[6];
	ftoa(message, tempData, 2);

	// Printing messages for serial verification if module is connected to PC
	//printf_P(PSTR("---------------------------------\n\r"));
	//printf_P(PSTR("%lu: APP Sending %s to 0%o...\n\r"),millis(),message,0);

	// Send it to the pc
	RF24NetworkHeader header(/*to node*/ 02);
	network.write(header,&message,sizeof(message));

	// Actuator – Led function
	ledActions();
	delay(2000);

	// Listen for a new node address
	//nodeconfig_listen();
}
```

```
// Convert float number to string
char *ftoa(char *a, double f, int precision)
{
	long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

	char *ret = a;
	long heiltal = (long)f;
	itoa(heiltal, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal = abs((long)((f - heiltal) * p[precision]));
	itoa(desimal, a, 10);
	return ret;
}
```

```
// DS18B20 library and pin definitions
int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2
//Temperature chip i/o
OneWire ds(DS18S20_Pin);  // on digital pin 2

float getTemp()
{
	//returns the temperature from one DS18S20 in DEG Celsius
	byte data[12];
	byte addr[8];

	if ( !ds.search(addr)) {
		//no more sensors on chain, reset search
		ds.reset_search();
		return -1000;
	}

	if ( OneWire::crc8( addr, 7) != addr[7])
	{
		Serial.println("CRC is not valid!");
		return -1000;
	}

	if ( addr[0] != 0x10 && addr[0] != 0x28)
	{
		Serial.print("Device is not recognized");
		return -1000;
	}

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

	byte present = ds.reset();
	ds.select(addr);
	ds.write(0xBE); // Read Scratchpad

	for (int i = 0; i < 9; i++)  // we need 9 bytes
	{
		data[i] = ds.read();
	}

	ds.reset_search();
	byte MSB = data[1];
	byte LSB = data[0];
```

```
        float tempRead = ((MSB << 8) | LSB); //using two's compliment
        float TemperatureSum = tempRead / 16;

        return TemperatureSum;
}
```

```
/*
Copyright (C) 2011 J. Coliz <maniacbug@ymail.com>
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
version 2 as published by the Free Software Foundation.
 */

/**
 * @file printf.h
 *
 * Setup necessary to direct stdout to the Arduino Serial library, which
 * enables 'printf'
 */

#ifndef __PRINTF_H__
#define __PRINTF_H__

#ifdef ARDUINO

int serial_putc( char c, FILE * )
{
	Serial.write( c );
	return c;
}

void printf_begin(void)
{
	fdevopen( &serial_putc, 0 );
}

#else
#error This example is only for use on Arduino.
#endif // ARDUINO

#endif // __PRINTF_H__
```

Α.3 Ο Έλεγχος από τον Η/Υ

Κώδικας Α.7 Το βασικό αρχείο κώδικα του module του Η/Υ

```
////////////////////////////////////////////////////////////////////////////////////////////////
// This program communicates serially with the Java Applet                                    //
// and via RF with the thermostat. It receives and sends temperatures                         //
// to and from both                                                                           //
////////////////////////////////////////////////////////////////////////////////////////////////
#include <avr/pgmspace.h>
#include <RF24Network.h>
#include <RF24.h>
#include <SPI.h>
#include "nodeconfig.h"
#include "ftoa.h"


//////////////////////////////////////
// RF24 Radio + Network  //
//////////////////////////////////////
#ifndef PINS_DEFINED
#define __PLATFORM__ "Getting Started board"

const int rf_ce = 9;
const int rf_csn = 10;

#endif

RF24 radio(rf_ce,rf_csn);
RF24Network network(radio);

// Our node address
uint16_t this_node;


//variables for serial communication with java
char message[24];
char *temp1;
char *temp2;
char *temp3;
char *temp4;
double tempFromNode1 = 0;
double tempFromNode2 = 0;
double settemp1 = 0;
double settemp2 = 0;
int ak1 = 0;
int ak2 = 0;
```

```
double dec1 = 0;
double dec2 = 0;
byte space = 0;
byte separator = 0;
bool ok = 0;

// This function reads the serial data in 6 bytes, 4 for the decimal and integer parts
// of the temperatures sent from the PC, one for delimiter and one for ending
// It also runs itself in case new data is sent while running
// If it receives less than 6 bytes it does nothing, if it receives 6<bytes<12 it flushes the
// remaining bytes. If it receives more than 12 bytes, it keeps the 6 it needs
void readSerial()
{
        ok = 0;
        if (Serial.available() >= 6)
        {
                ak1 = Serial.read();
                dec1 = Serial.read();
                separator = Serial.read();
                ak2 = Serial.read();
                dec2 = Serial.read();
                space = Serial.read();
                settemp1 = ak1 + (dec1/100);
                settemp2 = ak2 + (dec2/100);
                readSerial();
                Serial.flush();
                ok = 1;
        }
}

// This function prints the temperature data in the Applet terminal Log
void exportData(){
        Serial.println("Current temperature:");
        Serial.print("Temp1: ");
        Serial.println(tempFromNode1);
        Serial.print("Temp2: ");
        Serial.print(tempFromNode2);
        Serial.print("\n");
        Serial.println("Set temperature:");
        Serial.print("Temp1: ");
        Serial.println(settemp1);
        Serial.print("Temp2: ");
        Serial.println(settemp2);
        Serial.print("- - - - - - - - - - - - - - - - - - - - - - - - - \n");
}
```

```
// This function exports the 4 temperatures received from the thermostat and stores them
void dataThermostat(char *mes)
{
	String msgbuf(mes);
	temp1 = strtok(mes,"-");
	temp2 = strtok(NULL,"-");
	temp3 = strtok(NULL,"-");
	temp4 = strtok(NULL,"-");
	settemp1 = onedecimal ( atof (temp1));
	settemp2 = onedecimal ( atof (temp2));
	tempFromNode1 = onedecimal ( atof (temp3));
	tempFromNode2 = onedecimal ( atof (temp4));
	msgbuf.toCharArray(mes,24);
}


void setup(void)
{
	// Print version and RF data in the Applet Log on startup
	Serial.begin(57600);
	printf_begin();
	printf_P(PSTR("\n\rRF24Network/examples/sensornet/\n\r"));
	printf_P(PSTR("PLATFORM: " __PLATFORM__ "\n\r"),program_version);
	printf_P(PSTR("VERSION: %s\n\r"),program_version);

	// Pull node addr out of eeprom & bring up network
	this_node = nodeconfig_read();
	SPI.begin();
	radio.begin();
	network.begin(/*channel*/ 92, /*node address*/ this_node);
}



void loop(void)
{
	// Pump the network regularly
	network.update();
	// Is there anything ready for us?
	while ( network.available() )
	{
		// If so, grab it and store it
		RF24NetworkHeader header;
		char message[24];
		network.read(header,&message,sizeof(message));
		dataThermostat(message);
	}
	// Stop listening
```

```
        // Read the serial data from pc if any and if read, send the temperatures
        // set from PC to the thermostat
        readSerial();
        if (ok)
        {
                char message[12];
                floatsToMsg(message, settemp1, settemp2, 2);
                RF24NetworkHeader header(/*to node*/ 02);
                network.write(header,&message,sizeof(message));
        }
        exportData();
        delay(5000);


        // Listen for a new node address
        //  nodeconfig_listen();
}
```

```
// Remove second decimal from temperature
double onedecimal(double f)
{
	double minus = 10*f;
	int cut = minus;
	double buf = cut;
	double ret = buf/10;
	return ret;
}

// Save temperatures to string to send to thermostat
char *floatsToMsg(char *a, double f, double d, int precision)
{
	long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

	char *ret = a;
	long heiltal = (long)f;
	itoa(heiltal, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal = abs((long)((f - heiltal) * p[precision]));
	itoa(desimal, a, 10);
	while (*a != '\0') a++;
	*a++ = '-';
	long heiltal2 = (long)d;
	itoa(heiltal2, a, 10);
	while (*a != '\0') a++;
	*a++ = '.';
	long desimal2 = abs((long)((d - heiltal2) * p[precision]));
	itoa(desimal2, a, 10);
	return ret;
}
```

Α.4 Διευθυνσιοδότηση των κόμβων

Τα δυο παρακάτω αρχεία, ένα cpp και ένα header περιλαμβάνονται ως είναι και στους τρεις κώδικες που παρατέθηκαν και χρησιμοποιούνται για την αποθήκευση και την αναγνώριση της διεύθυνσης των κόμβων.

Κώδικας Α.9 Το αρχείο nodeconfig.cpp για τη διευθυνσιοδότηση των nodes

```
/*
Copyright (C) 2011 James Coliz, Jr. <maniacbug@ymail.com>

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
version 2 as published by the Free Software Foundation.
*/

#include "RF24Network_config.h"
#include <avr/eeprom.h>
#include <avr/pgmspace.h>
#include "nodeconfig.h"

// Where in EEPROM is the address stored?
uint8_t* address_at_eeprom_location = (uint8_t*)10;

// What flag value is stored there so we know the value is valid?
const uint8_t valid_eeprom_flag = 0xdf;

// What are the actual node values that we want to use?
// EEPROM locations are actually just indices into this array
const uint16_t node_address_set[10] = { 00, 02, 05, 012, 015, 022, 025, 032, 035, 045 };

uint8_t nodeconfig_read(void)
{
        uint8_t result = 0;

        // Look for the token in EEPROM to indicate the following value is
        // a validly set node address
        if ( eeprom_read_byte(address_at_eeprom_location) == valid_eeprom_flag )
        {
                // Read the address from EEPROM
                result = node_address_set[ eeprom_read_byte(address_at_eeprom_location+1) ];
                printf_P(PSTR("ADDRESS: %u\n\r"),result);
        }
        else
        {
                printf_P(PSTR("*** No valid address found.  Send 0-9 via serial to set node
```

```
                address\n\r"));
                while(1)
                {
                        nodeconfig_listen();
                }
        }

        return result;
}

void nodeconfig_listen(void)
{
        //
        // Listen for serial input, which is how we set the address
        //
        if (Serial.available())
        {
                // If the character on serial input is in a valid range...
                char c = Serial.read();
                if ( c >= '0' && c <= '9' )
                {
                        // It is our address
                        eeprom_write_byte(address_at_eeprom_location,valid_eeprom_flag);
                        eeprom_write_byte(address_at_eeprom_location+1,c-'0');

                        // And we are done right now (no easy way to soft reset)
                        printf_P(PSTR("\n\rManually reset index to: %c, address 0%o\n\rPress
                        RESET to continue!"),c,node_address_set[c-'0']);
                        while(1);
                }
        }
}
```

```
/*
 Copyright (C) 2011 James Coliz, Jr. <maniacbug@ymail.com>

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 version 2 as published by the Free Software Foundation.
 */

#ifndef __NODECONFIG_H__
#define __NODECONFIG_H__

uint8_t nodeconfig_read(void);
void nodeconfig_listen(void);

#endif // __NODECONFIG_H__
```

A.5 To Java Applet

Κώδικας Α.11 Το αρχείο GUI.java για την κατασκευή του γραφικού περιβάλλοντος

```
package RfControlPanel;

import java.awt.Color;
import javax.swing.*;


public class GUI extends javax.swing.JFrame {
        //Communicator object
        Communicator communicator = null;
        ActionList actionList = null;


        /** Creates new form GUI */
        public GUI() {
                initComponents();
                createObjects();
                communicator.searchForPorts();
                actionList.toggleControls();
           }

        private void createObjects()
        {
                communicator = new Communicator(this);
                actionList = new ActionList(this);
        }

        @SuppressWarnings("unchecked")

        private void initComponents()
        {

                jScrollPane1 = new javax.swing.JScrollPane();
                jTextArea1 = new javax.swing.JTextArea();
                jLabel1 = new javax.swing.JLabel();
                jLabel2 = new javax.swing.JLabel();
                jLabel3 = new javax.swing.JLabel();
                jLabel4 = new javax.swing.JLabel();
                lblLeft = new javax.swing.JLabel();
                raiseTempOne = new javax.swing.JButton();
                redTempOne = new javax.swing.JButton();
                raiseTempTwo = new javax.swing.JButton();
                redTempTwo = new javax.swing.JButton();
                updatebutton = new javax.swing.JButton();
```

```
setbutton = new javax.swing.JButton();
lblRight = new javax.swing.JLabel();
cboxPorts = new javax.swing.JComboBox();
jLabel5 = new javax.swing.JLabel();
btnConnect = new javax.swing.JButton();
btnDisconnect = new javax.swing.JButton();
jLabel13 = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
txtLog = new javax.swing.JTextArea();

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("RF Thermostat Control Panel");
setIconImage(new ImageIcon("icon2.png").getImage());

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel1.setText("RF Thermostat Control Panel");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel2.setText("Current Temp");

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setText("Temp 1");

jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel4.setText("Temp 2");

lblLeft.setFont(new java.awt.Font("Tahoma", 1, 18));
lblLeft.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblLeft.setText("0");


updatebutton.setText("Update Info");
updatebutton.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        updatebuttonActionPerformed(evt);
        }
});

setbutton.setText("Set Temperatures");
setbutton.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        setbuttonActionPerformed(evt);
```

```
                }
        });


raiseTempOne.setText("temp1 +");
raiseTempOne.addActionListener(new java.awt.event.ActionListener()
        {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                raiseTempOneActionPerformed(evt);
        }
});

redTempOne.setText("temp1 -");
redTempOne.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        redTempOneActionPerformed(evt);
        }
});

raiseTempTwo.setText("temp2 +");
raiseTempTwo.addActionListener(new java.awt.event.ActionListener()
        {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                raiseTempTwoActionPerformed(evt);
        }
});


lblRight.setFont(new java.awt.Font("Tahoma", 1, 18));
lblRight.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblRight.setText("0");

redTempTwo.setText("temp2 -");
redTempTwo.addActionListener(new java.awt.event.ActionListener()
        {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
                redTempTwoActionPerformed(evt);
        }
});

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel5.setText("Select the Arduino COM Port");

btnConnect.setText("Connect");
btnConnect.addActionListener(new java.awt.event.ActionListener()
```

```
        {
                public void actionPerformed(java.awt.event.ActionEvent evt)
                {
                        btnConnectActionPerformed(evt);
                }
        });

        btnDisconnect.setText("Disconnect");
        btnDisconnect.addActionListener(new java.awt.event.ActionListener()
                {
                        public void actionPerformed(java.awt.event.ActionEvent evt)
                        {
                        btnDisconnectActionPerformed(evt);
                        }
                });



        jLabel13.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel13.setText("Log");

        txtLog.setColumns(20);
        txtLog.setEditable(false);
        txtLog.setLineWrap(true);
        txtLog.setRows(5);
        txtLog.setFocusable(false);
        jScrollPane2.setViewportView(txtLog);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel1)
        .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addComponent(cboxPorts, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnConnect)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnDisconnect))
        .addComponent(jLabel5)
        .addComponent(jLabel2)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
```

```
        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,layout.createSequentialGrou
p()
        .addComponent(jLabel3,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel4,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(redTempOne)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
        .addComponent(lblLeft,                javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(raiseTempOne, javax.swing.GroupLayout.Alignment.LEADING)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(redTempTwo)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
        .addComponent(lblRight,               javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(raiseTempTwo, javax.swing.GroupLayout.Alignment.LEADING)))))
        .addGroup(layout.createSequentialGroup()
        .addComponent(setbutton)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel13)
        .addComponent(jScrollPane2,    javax.swing.GroupLayout.PREFERRED_SIZE,    333,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addContainerGap())
        );

        layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
        .addComponent(jLabel5)
        .addComponent(jLabel13))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
        .addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.TRAILING)
```

```
        .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
        .addComponent(cboxPorts,                javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(btnConnect)
        .addComponent(btnDisconnect))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel2)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
        .addComponent(jLabel3)
        .addComponent(jLabel4))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addComponent(raiseTempOne)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblLeft)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(redTempOne))
        .addGroup(layout.createSequentialGroup()
        .addComponent(raiseTempTwo)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblRight)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(redTempTwo)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
        .addComponent(setbutton))))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);

        pack();
        }// </editor-fold>//GEN-END:initComponents


        private void raiseTempOneActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnLeftAccelActionPerformed
                actionList.setTempSet1(actionList.raise(actionList.getTempSet1()));
                actionList.updateLabels();
        }//GEN-LAST:event_btnLeftAccelActionPerformed

        private void redTempOneActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnLeftDecelActionPerformed
                actionList.setTempSet1(actionList.red(actionList.getTempSet1()));
                actionList.updateLabels();
        }//GEN-LAST:event_btnLeftDecelActionPerformed
```

```
        private void raiseTempTwoActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnRightAccelActionPerformed
                actionList.setTempSet2(actionList.raise(actionList.getTempSet2()));
                actionList.updateLabels();
        }//GEN-LAST:event_btnRightAccelActionPerformed

        private void redTempTwoActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnRightDecelActionPerformed
                actionList.setTempSet2(actionList.red(actionList.getTempSet2()));
                actionList.updateLabels();
        }//GEN-LAST:event_btnRightDecelActionPerformed

        private void updatebuttonActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_updatebuttonActionPerformed
                actionList.updateLabels();
        }//GEN-LAST:event_updatebuttonActionPerformed

        private void setbuttonActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_updatebuttonActionPerformed
                actionList.updateSerial();
        }//GEN-LAST:event_setbuttonActionPerformed

        private void btnConnectActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnConnectActionPerformed
                communicator.connect();
                if (communicator.getConnected() == true)
                {
                        if (communicator.initIOStream() == true)
                        {
                                communicator.initListener();
                                actionList.updateLabels();
                        }
                }
        }//GEN-LAST:event_btnConnectActionPerformed

        private void btnDisconnectActionPerformed(java.awt.event.ActionEvent evt)
        {//GEN-FIRST:event_btnDisconnectActionPerformed
                communicator.disconnect();
        }//GEN-LAST:event_btnDisconnectActionPerformed

public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable()
        {
                public void run()
                {
                        new GUI().setVisible(true);
                }
        });
}
```

```
    public javax.swing.JButton btnConnect;
    public javax.swing.JButton btnDisconnect;
    public javax.swing.JButton raiseTempOne;
    public javax.swing.JButton redTempOne;
    public javax.swing.JButton raiseTempTwo;
    public javax.swing.JButton redTempTwo;
    public javax.swing.JButton setbutton;
    public javax.swing.JButton updatebutton;
    public javax.swing.JComboBox cboxPorts;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    public javax.swing.JLabel lblLeft;
    public javax.swing.JLabel lblRight;
    public javax.swing.JTextArea txtLog;
    // End of variables declaration//GEN-END:variables
}
```

```
package RfControlPanel;

import gnu.io.*;
import java.awt.Color;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.ObjectOutputStream;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.TooManyListenersException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class Communicator implements SerialPortEventListener
{
	//passed from main GUI
	GUI window = null;

	//for containing the ports that will be found
	private Enumeration ports = null;
	//map the port names to CommPortIdentifiers
	private HashMap portMap = new HashMap();

	//this is the object that contains the opened port
	private CommPortIdentifier selectedPortIdentifier = null;
	private SerialPort serialPort = null;

	//input and output streams for sending and receiving data
	private InputStream input = null;
	private OutputStream output = null;

	//just a boolean flag that i use for enabling
	//and disabling buttons depending on whether the program
	//is connected to a serial port or not
	private boolean bConnected = false;

	//the timeout value for connecting with the port
	final static int TIMEOUT = 2000;

	//some ascii values for for certain things
	final static int SPACE_ASCII = 32;
	final static int DASH_ASCII = 45;
	final static int NEW_LINE_ASCII = 10;

	//a string for recording what goes on in the program
	//this string is written to the GUI
```

```
String logText = "";

public Communicator(GUI window)
{
	this.window = window;
}

//search for all the serial ports
//pre: none
//post: adds all the found ports to a combo box on the GUI
public void searchForPorts()
{
	ports = CommPortIdentifier.getPortIdentifiers();

	while (ports.hasMoreElements())
	{
		CommPortIdentifier curPort = (CommPortIdentifier)ports.nextElement();

		//get only serial ports
		if (curPort.getPortType() == CommPortIdentifier.PORT_SERIAL)
		{
			window.cboxPorts.addItem(curPort.getName());
			portMap.put(curPort.getName(), curPort);
		}
	}
}

//connect to the selected port in the combo box
//pre: ports are already found by using the searchForPorts method
//post: the connected comm port is stored in commPort, otherwise,
//an exception is generated
public void connect()
{
	String selectedPort = (String)window.cboxPorts.getSelectedItem();
	selectedPortIdentifier = (CommPortIdentifier)portMap.get(selectedPort);

	CommPort commPort = null;

	try
	{
		//the method below returns an object of type CommPort
		commPort = selectedPortIdentifier.open("RfControlPanel", TIMEOUT);
		//the CommPort object can be casted to a SerialPort object
		serialPort = (SerialPort)commPort;

		//for controlling GUI elements
		setConnected(true);

		//logging
		logText = selectedPort + " opened successfully.";
```

```
                        window.txtLog.setForeground(Color.black);
                        window.txtLog.append(logText + "\n");
serialPort.setSerialPortParams(57600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialP
ort.PARITY_NONE);

                        //enables the controls on the GUI if a successful connection is made
                        window.actionList.toggleControls();
                }
                catch (PortInUseException e)
                {
                        logText = selectedPort + " is in use. (" + e.toString() + ")";

                        window.txtLog.setForeground(Color.RED);
                        window.txtLog.append(logText + "\n");
                }
                catch (Exception e)
                {
                        logText = "Failed to open " + selectedPort + "(" + e.toString() + ")";
                        window.txtLog.append(logText + "\n");
                        window.txtLog.setForeground(Color.RED);
                }
        }

        //open the input and output streams
        //pre: an open port
        //post: initialized intput and output streams for use to communicate data
        public boolean initIOStream()
        {
                //return value for whather opening the streams is successful or not
                boolean successful = false;

                try
                {
                        //
                        input = serialPort.getInputStream();
                        output = serialPort.getOutputStream();
                        writeData(0, 0);

                        successful = true;
                        return successful;
                }
                catch (IOException e)
                {
                        logText = "I/O Streams failed to open. (" + e.toString() + ")";
                        window.txtLog.setForeground(Color.red);
                        window.txtLog.append(logText + "\n");
                        return successful;
                }
        }
```

```
//starts the event listener that knows whenever data is available to be read
//pre: an open serial port
//post: an event listener for the serial port that knows when data is recieved
public void initListener()
{
	try
	{
		serialPort.addEventListener(this);
	serialPort.notifyOnDataAvailable(true);
	}
	catch (TooManyListenersException e)
	{
		logText = "Too many listeners. (" + e.toString() + ")";
		window.txtLog.setForeground(Color.red);
		window.txtLog.append(logText + "\n");
	}
}

//disconnect the serial port
//pre: an open serial port
//post: clsoed serial port
public void disconnect()
{
	//close the serial port
	try
	{
		writeData(0, 0);

		serialPort.removeEventListener();
		serialPort.close();
		input.close();
		output.close();
		setConnected(false);
		window.actionList.toggleControls();

		logText = "Disconnected.";
		window.txtLog.setForeground(Color.red);
		window.txtLog.append(logText + "\n");
	}
	catch (Exception e)
	{
	logText = "Failed to close " + serialPort.getName() + "(" + e.toString() + ")";
	window.txtLog.setForeground(Color.red);
	window.txtLog.append(logText + "\n");
	}
}
```

```
	final public boolean getConnected()
	{
		return bConnected;
	}

	public void setConnected(boolean bConnected)
	{
		this.bConnected = bConnected;
	}

	//what happens when data is received
	//pre: serial event is triggered
	//post: processing on the data it reads
	public void serialEvent(SerialPortEvent evt)
	{
	if (evt.getEventType() == SerialPortEvent.DATA_AVAILABLE)
	{
		try
		{
			byte singleData = (byte)input.read();

			if (singleData != NEW_LINE_ASCII)
			{
				logText = new String(new byte[] {singleData});
				window.txtLog.append(logText);
			}
			else
			{
				window.txtLog.append("\n");
			}
		}
		catch (Exception e)
		{
			logText = "Failed to read data. (" + e.toString() + ")";
			window.txtLog.setForeground(Color.red);
			window.txtLog.append(logText + "\n");
		}
		try
		{
	 window.txtLog.scrollRectToVisible(window.txtLog.modelToView(window.txtLog.getDo
cument().getLength ()));
		}
		catch (javax.swing.text.BadLocationException err) {}
	}
	}

	//method that can be called to send data
	//pre: open serial port
	//post: data sent to the other device
```

```
	public void writeData(double tempset1, double tempset2)
	{
		try
		{
			int ak1 = (int) tempset1;
			double decimal1 = 100*(tempset1 - ak1);
			int deccent1 = (int) decimal1;
			output.write(ak1);
			output.flush();
			output.write(deccent1);
			output.flush();
			//this is a delimiter for the data
			output.write(DASH_ASCII);
			output.flush();

			int ak2 = (int) tempset2;
			double decimal2 = 100*(tempset2 - ak2);
			int deccent2 = (int) decimal2;
			output.write(ak2);
			output.flush();
			output.write(deccent2);
			output.flush();
			//will be read as a byte so it is a space key
			output.write(SPACE_ASCII);
			output.flush();
		}
		catch (Exception e)
		{
			logText = "Failed to write data. (" + e.toString() + ")";
			window.txtLog.setForeground(Color.red);
			window.txtLog.append(logText + "\n");
		}
	}

	public void getCurrTime()
	{
		Calendar cal = Calendar.getInstance();
		cal.getTime();
		SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
		window.txtLog.append( sdf.format(cal.getTime()) );
	}
}
```

```
package RfControlPanel;

import java.awt.event.ActionEvent;
import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.JButton;

public class ActionList {
	GUI window = null;

	public double tempset1 = 18;
	public double tempset2 = 18;

	private static final double INCREMENT = 0.5;


	public ActionList(GUI window)
	{
		this.window = window;
	}


	public void toggleControls()
	{
		if (window.communicator.getConnected() == true)
		{
			window.raiseTempOne.setEnabled(true);
			window.redTempOne.setEnabled(true);
			window.raiseTempTwo.setEnabled(true);
			window.redTempTwo.setEnabled(true);

			window.btnDisconnect.setEnabled(true);
			window.btnConnect.setEnabled(false);
			window.cboxPorts.setEnabled(false);

			window.setbutton.setEnabled(true);
			window.updatebutton.setEnabled(true);
		}
		else
		{
			window.raiseTempOne.setEnabled(false);
			window.redTempOne.setEnabled(false);
			window.raiseTempTwo.setEnabled(false);
			window.redTempTwo.setEnabled(false);

			window.btnDisconnect.setEnabled(false);
			window.btnConnect.setEnabled(true);
```

```
                window.cboxPorts.setEnabled(true);

                window.setbutton.setEnabled(false);
                window.updatebutton.setEnabled(false);
        }
}

//defining the action
Action raiseTempOne = new AbstractAction()
{
        public void actionPerformed(ActionEvent evt)
        {
                tempset1 = raise(tempset1);
                updateLabels();
        }
};

Action redTempOne = new AbstractAction()
{
        public void actionPerformed(ActionEvent evt)
        {
                tempset1 = red(tempset1);
                updateLabels();
        }
};

Action raiseTempTwo = new AbstractAction()
{
        public void actionPerformed(ActionEvent evt)
        {
                tempset2 = raise(tempset2);
                updateLabels();
        }
};

Action redTempTwo = new AbstractAction()
{
        public void actionPerformed(ActionEvent evt)
        {
        tempset2 = red(tempset2);
        updateLabels();
        }
};
```

```
public void updateLabels()
{
	window.lblLeft.setText(String.valueOf(tempset1));
	window.lblRight.setText(String.valueOf(tempset2));
}


public void updateSerial()
{
	window.communicator.writeData(tempset1, tempset2);
	window.txtLog.append("Temperatures set via PC user at ");
	window.communicator.getCurrTime();
	window.txtLog.append("\n");
	window.txtLog.append("- - - - - - - - - - - - - - - - - - - - - - - - - \n");
}

public double raise(double temperature)
{
	if (temperature < 30)
	{
		temperature += INCREMENT;
	}
return temperature;
}

public double red(double temperature)
{
	if (temperature > 5)
	{
		temperature -= INCREMENT;
	}
	return temperature;
}

final public double getTempSet1()
{
	return tempset1;
}

public void setTempSet1(double value)
{
	tempset1 = value;
}

final public double getTempSet2()
{
	return tempset2;
}
```

```
	public void setTempSet2(double value)
	{
		tempset2 = value;
	}
}
```