ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

10/30/2015

ΕΞΥΠΝΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΟΙΚΙΑΚΗΣ ΗΛΕΚΤΡΙΚΗΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΓΙΑ ΑΤΟΜΑ ΜΕ ΕΙΔΙΚΕΣ ΑΝΑΓΚΕΣ

Χριστόφορος - Βασίλειος Κούτσικος

Α.Μ.: 2013019013

Επιβλέπων Καθηγητής:
Φώτιος Κανέλλος

…….....

Χριστόφορος-Βασίλειος Κ. Κούτσικος

Διπλωματούχος Ηλεκτρονικός Μηχανικός

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να εκφράσω τις ευχαριστίες μου σε όσους με βοήθησαν κατά την εκπόνησή της.

Ευχαριστώ θερμά τον Καθηγητή μου κ. Φώτιο Κανέλλο , για την καθοδήγησή του και την άψογη συνεργασία για την περάτωση της εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τους Καθηγητές κ. Βασίλειο Κουικόγλου και κ. Ιωάννη Μακρή για τη συμμετοχή τους στην τριμελή εξεταστική επιτροπή.

Τέλος, εκφράζω την ευγνωμοσύνη μου στους γονείς μου, Κωνσταντίνο και Ευαγγελία , οι οποίοι με στήριξαν καθ όλη τη διάρκεια των σπουδών μου μέσω της κατανόησης που μου έδειξαν και των θυσιών που έκαναν.

Περίληψη

Από τις αρχές του 19ου αιώνα έχει εμφανιστεί η έννοια της έξυπνης κατοικίας και μέχρι σήμερα ολοένα και εξελίσετε.

Μέσω αυτής της διπλωματικής διατριβής θέλουμε να παρουσιάσουμε την υλοποίηση μιας έξυπνης κατοικίας για Α.με.Α. με ιδιαίτερα χαμηλό κόστος και εύκολα τροποποιήσιμο για τις ανάγκες του εκάστοτε ιδιοκτήτη.

Κάνωντας χρήση της πλατφόρμας arduino, των κατάλληλων αισθητήρων και τη συγγραφή κώδικα δημιουργήσαμε μια πλατφόρμα η οποία θα μπορεί να διευκολύνει την καθημερινότητα των ατόμων αυτών.

Οι ενέργειες που θα μπορούν να εκτελεστούν θα είναι είτε μέσω Η/Υ είτε κάποιου σύγχρονου κινητού τηλεφώνου.

Επιτρέπει λοιπόν στο χρήστη να διαχειρίζεται τις οικιακές συσκευές μέσω δικτύου καθώς επίσης η πλατφόρμα θα γνωστοποιεί στο χρήστη σημαντικές πληροφορίες όπως η θερμοκρασία και η φωτεινότητα στο χώρο και θα μπορεί η συσκευή σε περιπτώσεις έκτατης ανάγκης να ενεργοποιεί κάποια συστήματα.

Λέξεις κλειδιά: Έξυπνο σπίτι , ΑμΕΑ, Αισθητήρες , Arduino, Shield

Πίνακας Περιεχομένων

1 Πίνακας Εικόνων

1 Εισαγωγή

Τις τελευταίες δεκαετίες έχουν αποκτήσει εξαιρετική δυναμική οι τεχνολογίες αυτοματισμών και επικοινωνιών που αφορούν στην έννοια της «έξυπνης κατοικίας». Η εξέλιξη της τεχνολογίας αυτής είναι αλματώδης με αποτέλεσμα η έννοια της «έξυπνης κατοικίας» να διευρύνει τα χαρακτηριστικά και τις δυνατότητές η πορεία αυτή είναι βέβαιο ότι θα συνεχιστεί βαθιά στο μέλλον. Η συνεισφορά της στη διευκόλυνση της καθημερινότητας του ανθρώπου είναι αδιαμφισβήτητη, και το γεγονός αυτό αποκτά ιδιαίτερη ανθρωπιστική διάσταση όταν κατορθώνει να εξυπηρετεί άτομα με ειδικές ανάγκες.

Σκοπός της παρούσας διατριβής είναι η πρόταση και παρουσίαση μιας τέτοιας καινοτομικής εκδοχής της «έξυπνης κατοικίας», σύμφωνα με την οποία ελέγχονται όλες οι συσκευές του σπιτιού, ακόμα και μέσω ενός σύγχρονου κινητού τηλεφώνου, επιδιώκοντας στη διευκόλυνση της καθημερινότητας των ατόμων με ειδικές ανάγκες.

2 Ιστορική αναδρομή

Η ιστορική αφετηρία των έξυπνων κατοικιών τοποθετείται χρονικά στα τέλη του 19ου αιώνα, όταν σε καινούριες κατοικίες εγκαταστάθηκαν ηλεκτρικά και τηλεφωνικά καλώδια.

Ένα μεγάλο άλμα έγινε προς το τέλος της δεκαετίας του ’50 κατά την οποία εισήχθησαν στην αγορά συσκευές τηλεχειρισμού για τον κλιματισμό και τις τηλεοράσεις. Η εταιρία που έθεσε τις βάσεις για το «έξυπνο σπίτι» ήταν η General Electric αναπτύσσοντας μια σειρά από καινοτόμες συσκευές όπως το πλυντήριο πιάτων (1954), αυτοκαθαριζόμενοι φούρνοι (1963),ψηφιακοί συναγερμοί, φορητά ραδιόφωνα και φούρνοι μικροκυμάτων(1978).

Μέχρι το 1965 δεν είχε γίνει επίσημη αναφορά στον όρο «έξυπνη κατοικία» από καμία εταιρία. Την ίδια χρονιά, σε μια παγκόσμια έκθεση παρουσιάστηκε μια πολύ πρωτοποριακή ιδέα για τα δεδομένα της εποχής. Κάθε σπίτι θα διέθετε κεντρικό υπολογιστή ο οποίος θα έλεγχε τα φώτα, τις αυτοματοποιημένες κουρτίνες και θα είχε τον έλεγχο του κλιματισμού σε όλο το σπίτι.

Η πραγματική ανάπτυξη του έξυπνου σπιτιού όμως έγινε τη δεκαετία του ’80 στην Ιαπωνία από τις εταιρίες Toshiba, Hitachi, Panasonicκαι Fujitsu, οι οποίες συνεργάστηκαν με μία ομάδα αρχιτεκτόνων μηχανικών για να κατασκευάσουν το πιο έξυπνο κτίριο στον κόσμο, στο οποίο όλα θα λειτουργούσαν με αυτοματοποιημένο τρόπο.

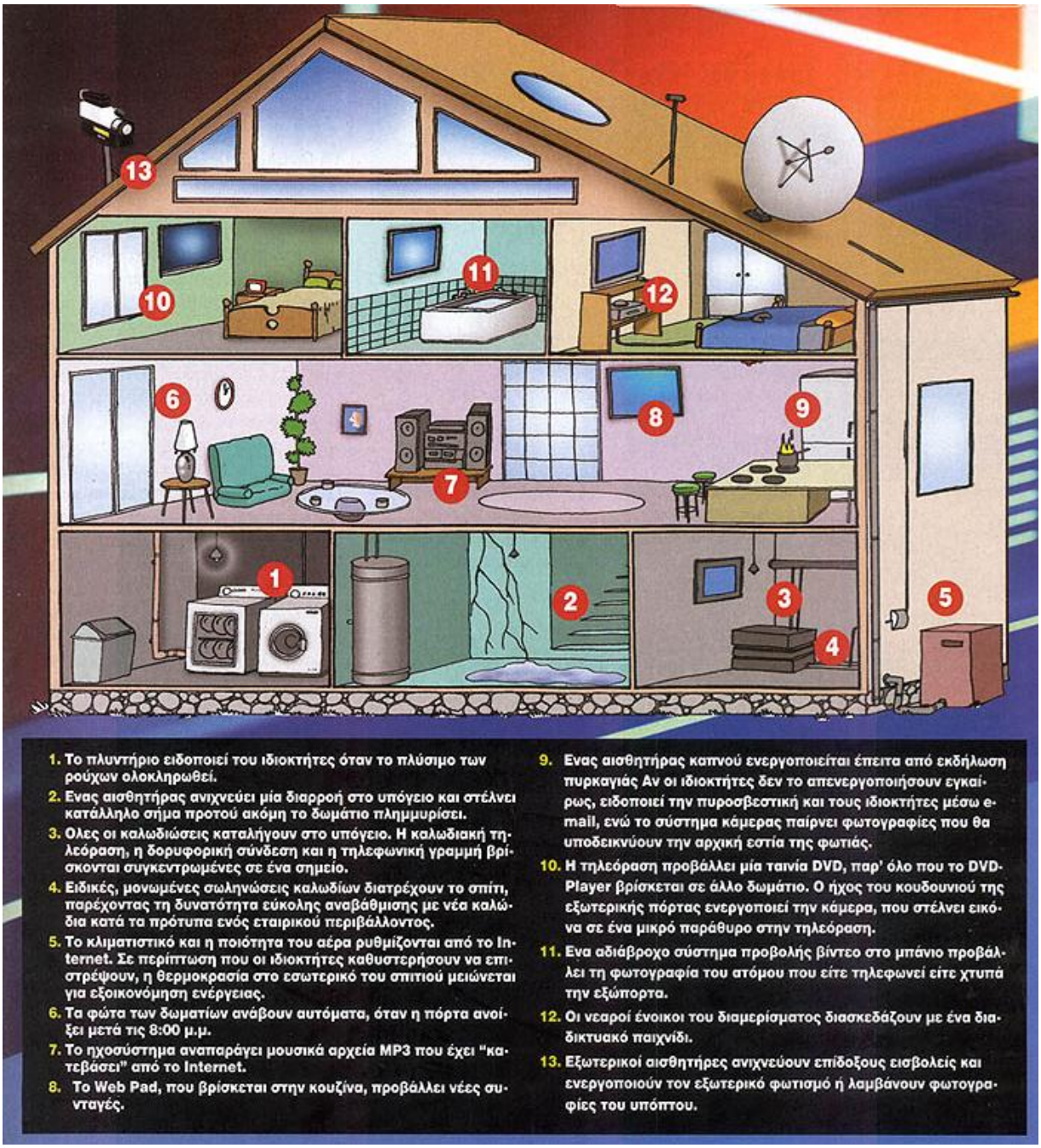
Εκείνη τη δεκαετία έλαβαν χώρα πολλά πειράματα χωρίς κανένα να στεφθεί με επιτυχία μιας και τα μέσα που χρησιμοποιούνταν, όπως οι υπολογιστές, οι ασύρματες συσκευές και το διαδίκτυο, είχαν αρκετά υψηλό κόστος και θεωρούνταν είδος πολυτελείας, εν αντιθέσει με σήμερα.

3 Τί είναι ένα «έξυπνο σπίτι»

Η φράση "έξυπνο σπίτι" ("smarthome") είναι ευρέως διαδεδομένη, αφού χρησιμοποιείται για να περιγράψει οποιαδήποτε οικία που ενσωματώνει σε μικρότερο ή μεγαλύτερο βαθμό τη δυνατότητα ρύθμισης κάποιων παραμέτρων των δυνατοτήτων και των ανέσεων που παρέχει. Στην αγγλική γλώσσα η συγκεκριμένη τεχνολογία αποδίδεται και ως homeautomation" και χρησιμοποιείται για να χαρακτηρίσει οποιοδήποτε σπίτι περιλαμβάνει κάποιου είδους τεχνητή νοημοσύνη. Αξιοποιώντας αυτή την τεχνητή νοημοσύνη, το εγκατεστημένο σύστημα έχει τη δυνατότητα να ρυθμίζει αυτόματα το περιβάλλον και τις παροχές της οικίας, σύμφωνα με τις προκαθορισμένες επιθυμίες ενοίκων. Για να επιτευχθεί ο στόχος αυτός, απαιτείται η ύπαρξη ενός συστήματος ώστε όλες οι ηλεκτρικές και ηλεκτρονικές συσκευές και διατάξεις της οικίας (ή τουλάχιστον ένα μεγάλο μέρος αυτών) να επικοινωνούν με το κεντρικό σύστημα ελέγχου ή και μεταξύ τους, λαμβάνοντας και αποστέλλοντας εντολές και δεδομένα. Στην κατεύθυνση αυτή, η τεχνολογία έχει αναπτύξει πολλά ανταγωνιστικά πρότυπα, τα οποία προσφέρουν διαφορετικά πλεονεκτήματα αλλά έχουν και αντίστοιχα μειονεκτήματα.

Οι λειτουργίες που μπορεί να ενσωματώνει ένα «έξυπνο σπίτι» σχετίζονται σχεδόν με το σύνολο των ανθρώπινων δραστηριοτήτων όπως εργασία, καθημερινές ασχολίες, ψυχαγωγία (βλ. σχήμα 1). Για παράδειγμα, κατά τη διάρκεια του ύπνου το "έξυπνο" σπίτι θα ελέγχει την θερμοκρασία του δωματίου και την προσαρμόζει με βάση την επιθυμητή για τον ένοικο ή βέλτιστη τιμή. Επίσης, το «έξυπνο σπίτι» θα μπορούσε να παρακολουθείτο δελτίο πρόγνωσης του καιρού μέσω διαδικτύου, ώστε να σχεδιάζει το πότισμα ή μη του κήπου. Ακόμα είναι δυνατό να ρυθμίζει την αφύπνιση ανάλογα με το ημερήσιο πρόγραμμα κάθε ατόμου και να θέτει σε λειτουργία ενεργοβόρες οικιακές συσκευές κατά το νυχτερινό τιμολόγιο ηλεκτρικού ρεύματος, το οποίο είναι οικονομικότερο. Επιπροσθέτως, κατά την ώρα της αφύπνισης των ενοίκων, θα μπορούσε να δυναμώνει σταδιακά την ένταση του (φυσικού) φωτισμού, να ενεργοποιεί την παραγωγή καφέ, να ρυθμίζει την τηλεόραση ή το ραδιόφωνο στο επιθυμητό πρωινό κανάλι ή ραδιοφωνικό σταθμό και να προβάλλει τα νέα που τους ενδιαφέρουν και τα οποία εντόπισε κατά τη διάρκεια της νύκτας στο διαδίκτυο. Στην περίπτωση απουσίας ή εξόδου των ενοίκων από το σπίτι,

θα σβήνει τα φώτα και θα ρυθμίζει τη θέρμανση με τέτοιο τρόπο ώστε να εξοικονομείται ενέργεια, αλλά και θα ενεργοποιεί το σύστημα πυρασφάλειας και το σύστημα συναγερμού. Τέλος, όταν οι ιδιοκτήτες απουσιάζουν επί μακρόν, π.χ. για διακοπές ή οποιοδήποτε άλλο λόγο, θα μπορούσε να ελέγχει την ομαλή λειτουργία



Εικόνα 1 – Παράδειγμα έξυπνου σπιτιού

κάθε υποσυστήματος και να αποστέλλει λεπτομερή μηνύματα μέσω ηλεκτρονικού ταχυδρομείου στους ιδιοκτήτες, περιγράφοντας κάθε πρόβλημα που ενδεχομένως ήθελε προκύψει.

Κορυφαίο χαρακτηριστικό της τεχνολογίας «έξυπνο σπίτι» συνιστά ο απομακρυσμένος έλεγχος, καθώς παρέχει τη δυνατότητα στους ενοίκους να ενημερώνονται για την κατάσταση της κατοικίας και να παρεμβαίνουν στη λειτουργία της από οπουδήποτε κι αν βρίσκονται, αρκεί να διαθέτουν τρόπο επικοινωνιακής πρόσβασης (ενσύρματο ή ασύρματο). Όταν μάλιστα τα άτομα που διαμένουν στο «έξυπνο σπίτι» είναι με ειδικές ανάγκες, ο απομακρυσμένος έλεγχος παροχών, διατάξεων και συσκευών μπορεί να αποβεί σωτήριος σε πιθανή περίπτωση κινδύνου.

Όπως προαναφέρθηκε, ο απομακρυσμένος έλεγχος είναι εφικτός με πολλαπλούς τρόπους επικοινωνίας, δίχως να είναι υποχρεωτικό να υλοποιούνται όλοι. Ο πλέον απλός και διαδεδομένος αφορά την πρόσβαση μέσω τηλεφώνου, είτε με αναγνώριση των φωνητικών εντολών που δίνονται, είτε με τη χρήση του αριθμητικού πληκτρολογίου στις ψηφιακές τηλεφωνικές συσκευές. Εναλλακτικά ή επιπροσθέτως, αξιοποιώντας τις υπηρεσίες του διαδικτύου, πολλά συστήματα είναι σε θέση να προσφέρουν στους ενοίκους τον πλήρη έλεγχο της οικίας τους μέσω ενός εύχρηστου και λειτουργικού γραφικού περιβάλλοντος σε μορφή ιστοσελίδας που αντιπροσωπεύει το σύνολο του οικιακού εξοπλισμού.

Ανακεφαλαιώνοντας, η τεχνολογία του «έξυπνου σπιτιού» προσφέρεται για να αυτοματοποιηθούν ορισμένες ή οι περισσότερες από τις καθημερινές λειτουργίες μιας οικίας ή και για να επιτευχθεί μεγαλύτερη ευχρηστία καθώς και να βελτιωθούν οι δυνατότητες στις υπάρχουσες οικιακές συσκευές. Είναι πλέον αξιοσημείωτη η πληθώρα ανάλογων προϊόντων και υπηρεσιών που υπάρχουν στις μέρες μας και ακόμα πιο εντυπωσιακό είναι το γεγονός ότι λαμβάνουν χώρα συνεχείς βελτιώσεις και εξελίξεις, ώστε να επεκταθούν ακόμη περισσότερο οι δυνατότητες και οι παροχές της τεχνολογίας αυτοματισμού οικίας

4 Τα οφέλη της τεχνολογίας

Η δυνατότητα απομακρυσμένης πρόσβασης στο σπίτι, η ευκολία στο να δοθεί μια εντολή για κάποια λειτουργία, και ο άμεσος, προσωπικός έλεγχος του οικιακού περιβάλλοντος συνιστούν τα κυριότερα πλεονεκτήματα και προτερήματα της συγκεκριμένης τεχνολογίας. Η κεντρική και αυτοματοποιημένη διαχείριση ενδεχομένως του συνόλου των οικιακών συσκευών καθώς και η πολυεπίπεδη επιτήρηση για την αποφυγή διαφόρων κινδύνων και άρα η επίτευξη ασφάλειας, αποτελούν τους πλέον σημαντικούς λόγους για τους οποίους οι καταναλωτές είναι πιθανόν να ξοδέψουν ένα αξιοσέβαστο χρηματικό ποσό για τη μετατροπή ή την εξαρχής κατασκευή του δικού τους ιδανικού «έξυπνου σπιτιού». Όμως, για να εμφανισθεί το ανάλογο αγοραστικό κοινό και για να επιτευχθεί η καθολική αποδοχή της τεχνολογίας από τους τελικούς αγοραστές-χρήστες, είναι σημαντικό τα προσφερόμενα οφέλη να ανταποκρίνονται στις απαιτήσεις των χρηστών και να εναρμονίζονται με τα οικονομικά τους δεδομένα.

Συνοψίζοντας, παράλληλα με τη διαρκή μεταβολή του τρόπου ζωής του σύγχρονου ανθρώπινου, που πάντα περιλαμβάνει εξατομικευμένα ενδιαφέροντα και ανάγκες, ο οικιακός εξοπλισμός γίνεται όλο και πιο σύνθετος, κάνοντας αναγκαία την ύπαρξη προηγμένων συστημάτων ελέγχου και διαχείρισης που θα μπορούν να εξυπηρετήσουν τις νέες συσκευές και διατάξεις. Η υπάρχουσα καλωδιακή υποδομή των κατοικιών ήδη θεωρείται ανεπαρκής, ενώ στο μέλλον το χάσμα μεταξύ των δυνατοτήτων που μπορούν να προσφέρουν οι οικιακές συσκευές και των δυνατοτήτων που είναι δυνατόν να ενσωματωθούν στις υπάρχουσες καλωδιώσεις θα αυξάνεται συνεχώς. Για τους παραπάνω λόγους, αρκετές εταιρίες ήδη αναπτύσσουν και παρουσιάζουν στην παγκόσμια αγορά μία σειρά νέων τεχνολογιών, οι οποίες σχετίζονται τόσο με την υλική υποδομή των νέων δικτύων που θα προκύψουν, όσο και με τα πρωτόκολλα επικοινωνίας που θα εφαρμοστούν στα νέα αυτά δίκτυα. Η χρονική συγκυρία δεν θα μπορούσε να είναι καλύτερη, καθώς τις τελευταίες δεκαετίες το βιοτικό επίπεδο των ανθρώπων έχει αυξηθεί σημαντικά, με αποτέλεσμα η μετάβαση στη νέα εποχή των αυτοματοποιημένων οικιών να προκύπτει ως φυσικό επακόλουθο ακόμη και για τους πιο «τεχνολογικώς» μετριοπαθείς ανθρώπους.

5 Άτομα με ειδικές ανάγκες και «έξυπνο σπίτι»

Η πλειονότητα των μελετών που έχουν εκπονηθεί τα τελευταία χρόνια στην Ελλάδα καταδεικνύουν ότι τα υψηλότερα ποσοστά κοινωνικού αποκλεισμού παρατηρούνται στα άτομα με χρόνια προβλήματα υγείας και αναπηρίες. Σε έρευνα της Εθνικής Στατιστικής Υπηρεσία της Ελλάδας που διενήργησε το 2002, ποσοστό 64,2% των ατόμων με ειδικές ανάγκες ή με αναπηρία (Α.με.Α.) εμφανίζει μακροχρόνια ή μόνιμα προβλήματα υγείας ή/και αναπηρίας, ενώ σε ποσοστό πάνω από 50% είναι άνθρωποι ηλικίας μεγαλύτερης των 65 ετών.

Σε ότι αφορά το ανθρωποποίητο και δομημένο περιβάλλον, η πρόσβαση σε αυτό των Α.με.Α. είναι σήμερα το μέτρο που καθορίζει ένα φιλικό και ασφαλές περιβάλλον, απαραίτητο για την ποιότητα ζωής όλων των μελών της κοινωνίας, την αναβαθμίζει και της προσδίδει αξία. Μάλιστα, σε ότι αφορά την πρόσβαση και την παρουσία ατόμων με ειδικές ανάγκες ή με αναπηρία σε δημόσιους χώρους και κτίρια, υφίσταται ρυθμιστικό πλαίσιο (Κανονισμός 61, 1999) που καθορίζει τις παραμέτρους και τα στοιχεία αυτού του δομημένου περιβάλλοντος, ώστε να είναι πρωτίστως προσβάσιμο, αλλά και φιλικό και ασφαλές για Α.με.Α.

Και ενώ, ορθώς, πολλά λέγονται και γίνονται για ένα αστικό και κοινωνικό περιβάλλον που δεν αποκλείει και δεν δυσχεραίνει τα άτομα με ειδικές ανάγκες, λιγότερη έμφαση δίνεται στην ποιότητα ζωής των ανθρώπων αυτών εντός των οικιών τους, που κατά τεκμήριο, διαμένουν τον περισσότερο χρόνο. Το έλλειμα αυτό έρχεται να καλύψει η τεχνολογία της «αυτοματοποιημένης οικίας».

Το «έξυπνο σπίτι» πρέπει να είναι κατασκευασμένο σύμφωνα με τις εκάστοτε ανάγκες του ανθρώπου που ζει σε αυτό και επαυξημένα για Α.με.Α. πρέπει να είναι πολύ πιο εξελιγμένο και πιο εύχρηστο από οποιοδήποτε άλλο και να διαφοροποιείται ανάλογα με το είδος της αναπηρίας (κίνηση, όραση, ακοή, νόηση).

6 Περιγραφή του αναπτυχθέντος έξυπνου συστήματος

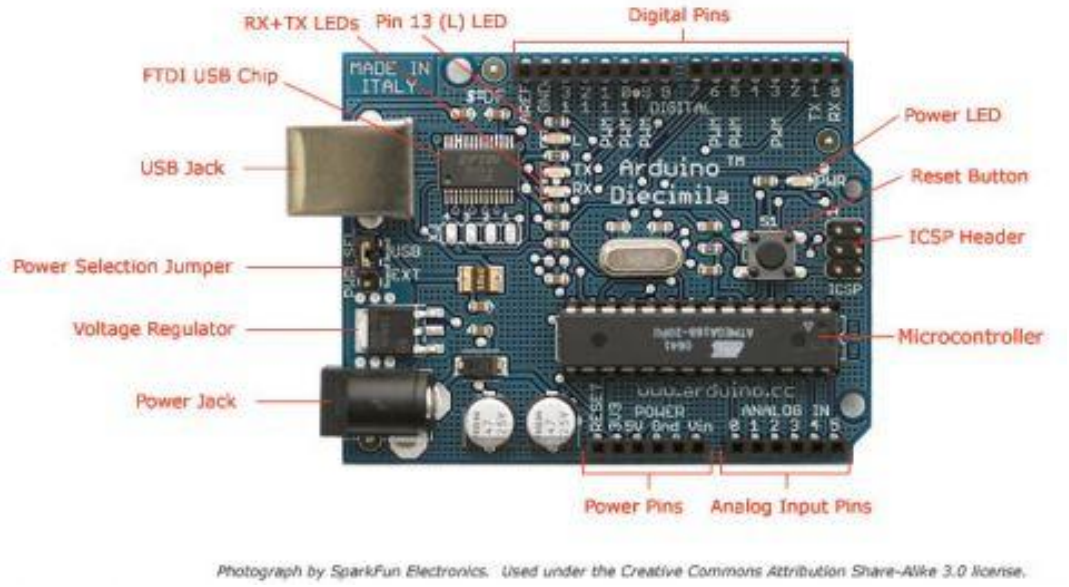
Το σύστημα έξυπνης διαχείρισης οικίας, που παρουσιάζεται πιλοτικά στα πλαίσια της παρούσας εργασίας, αποσκοπεί στον έλεγχο των οικιακών συσκευών μέσω οποιασδήποτε συσκευής διαθέτει πρόσβαση σε ασύρματο δίκτυο, ακόμα και με χρήση ενός κινητού τηλεφώνου. Έτσι μπορούμε να κάνουμε πιο εύκολη την καθημερινότητα ατόμων που έχουν παραδείγματος χάριν κινητικά προβλήματα, προβλήματα ακοής ή γενικότερα άτομα με ειδικές ανάγκες.

Στην πράξη δίνεται η δυνατότητα καθολικού χειρισμού μέσω ενός browser. Επίσης ανά πάσα στιγμή επιτρέπεται ο χρήστης να έχει εικόνα των λειτουργιών που εξελίσσονται στην οικία του, καθώς και να ορίζει ενέργειες που θα γίνονται αυτόματα, όπως για παράδειγμα:

- ✓ Έλεγχος της θερμοκρασίας.
- ✓ Ενεργοποίηση/απενεργοποίηση του θερμοσίφωνα.
- ✓ Άνοιγμα/κλείσιμο των διακοπτών φωτισμού.
- ✓ Έλεγχος της διαρροής υγραερίου και παροχή ηχητικής ειδοποίησης.
- ✓ Αυτόματο άνοιγμα των λαμπτήρων όταν ο φωτισμός δεν επαρκεί.
- ✓ Αυτόματο άνοιγμα/κλείσιμο κλιματισμού ανάλογα με τη θερμοκρασία του χώρου.
- ✓ Αυτόματη ενεργοποίηση της πυρόσβεσης όταν η θερμοκρασία ξεπεράσει τους 50^{o}C.
- ✓ Έλεγχος στάθμης σε κάποια δεξαμενή (νερού, λέβητα, κλπ).

Για την πιλοτική υλοποίηση αυτής της ιδέας γίνεται χρήση ενός μικροελεγκτή arduino ο οποίος αποτελεί και την «καρδιά» του αναπτυχθέντος συστήματος, καθώς και ομάδα αισθητήρων που θα αναφερθούν αναλυτικά παρακάτω.

7 Arduino



Εικόνα 2- Πλακέτα Arduino

Το Arduino αποτελεί μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα που διαθέτει έναν ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων και έχει τη δυνατότητα σύνδεσης με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider.

a. Ιστορικά στοιχεία

Το 2005, ξεκίνησε ένα σχέδιο προκειμένου να φτιαχτεί μία συσκευή για τον έλεγχο προγραμμάτων διαδραστικών σχεδίων από μαθητές, η οποία θα κόστιζε λιγότερο σε σχέση με άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνη την

χρονική περίοδο. Οι Massimo Banzi και David Cueartielles ονόμασαν το έργοArduino της Ivrea και ξεκίνησαν να παράγουν πλακέτες σε ένα μικρό εργοστάσιο στην Ivrea, κωμόπολη της επαρχίας Τορίνο στην περιοχή Πεδεμόντιο της βορειοδυτικής Ιταλίας - την ίδια περιοχή στην οποία στεγαζόταν η εταιρία υπολογιστών Olivetti.

Το Arduino δομήθηκε γύρω από το έργο Wiring του Hernando Barragan. Το Wiring ήταν η ακαδημαϊκήεργασία του Hernando στο Interaction Design Institute της Ivrea. Σκόπευε να ήταν μια ηλεκτρονική μορφή της επεξεργασίας που χρησιμοποιούνταν στα προγραμματιστικά περιβάλλοντα και είχε ως πρότυπο την Processing syntax.

- Τον Σεπτέμβρη του 2006ανακοινώθηκε το Arduino Mini
- Τον Οκτώβρη του 2008 ανακοινώθηκε ανακοινώθηκετο Arduino Duemilanove. Αρχικά βασίστηκε στο Atmel Atmega168, αλλά μετά στάλθηκε με το ATmega328.
- Τον Μάρτη του 2009 ανακοινώθηκε το Arduino Mega. Ήταν βασισμένο στο Atmel ATmega1280.
- Από τον Μάη του 2011 περισσότερα από 300,000 Arduino ήταν σε χρήση σε όλο τον κόσμο.
- Τον Ιούλη του 2012, ανακοινώθηκε το Arduino Leonardo. Ήταν βασισμένο στο Atmel ATmega32u4.
- Τον Οκτώβριο του 2012, ανακοινώθηκε το Arduino Due. Ήταν βασισμένο στο Atmel SAM3X8E, που είχε πυρήνα ARM Cortex-M3.
- Τον Νοέμβριο του 2012 ανακοινώθηκε το Arduino Micro. Ήταν βασισμένο στο Atmel ATmega32u4.
- Τον Μάη του 2013ανακοινώθηκε το Arduino Robot. Ήταν βασισμένο στο Atmel ATmega32u4 και ήταν το πρώτο επίσημο Arduino με ρόδες.
- Τον Μάη του 2013 ανακοινώθηκε το Arduino Yun. Ήταν Βασισμένο στο ATmega32u4 και στο Atheros AR9331 και ήταν το πρώτο προϊόν wifi που συνδύαζε το Arduino με την Linux.

b. Πλατφόρμα

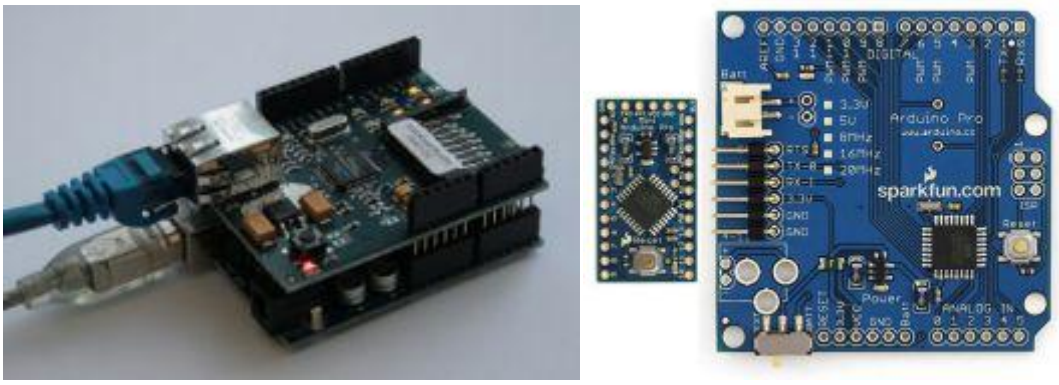
Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό αντηχητή σε κάποιες παραλλαγές). Ο μικροελεγκτής είναι προγραμματισμένος από τον κατασκευή του με ένα bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής.

Γενικά όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων RS-232 και TTL.

Οι πλακέτες Arduino που κυκλοφορούν σήμερα στην αγορά, συμπεριλαμβανόμενης και της Diecimila, προγραμματίζονται μέσω USB, εφαρμόζοντας ένα τσιπ προσαρμογέα USB-to-serial όπως το FTDI FT232. Κάποιες παραλλαγές, όπως το Arduino mini και το ανεπίσημο Boarduino, χρησιμοποιούν προσαρμογέα USB-to-serial σε μορφή πλακέτας ή καλωδίου.

Η πλακέτα του Arduino έχει εκτεθειμένες τις περισσότερες επαφές εισόδου/εξόδου για χρήση με άλλα κυκλώματα. Το Diecimila, για παράδειγμα, παρέχει 14 ψηφιακές επαφές εισόδου/εξόδου, από τις οποίες οι 6 μπορούν να παράγουν σήματα PWM, και 6 αναλογικές εισόδους. Αυτές οι επαφές είναι διαθέσιμες στην κορυφή της πλακέτας μέσω θηλυκών συνδέσεων μεγέθους 0,1 ιντσών.

Επίσης είναι διαθέσιμες στο εμπόριο διάφορες plug-in πλακέτες εφαρμογών γνωστές σαν “shields” («επεκτάσεις»).



Εικόνα 3 - Πλακέτα Εφαρμογής

c. Επεκτάσεις Arduino

Η πλατφόρμα του arduino κατασκευάστηκε με σκοπό να μπορεί με χαμηλό κόστος να μπορεί ο εκάστοτε χρήστης ανάλογα με τις απαιτήσεις του να την τροποποιεί.

Αυτό επιτυγχάνεται μέσω των επεκτάσεων που τοποθετούνται απλά πάνω στην πλατφόρμα και δίνουμε νέες δυνατότητες στην πλατφόρμα, όπως σύνδεση στο διαδίκτυο(μέσω Ethernet, wi-fi ή GSM) διαχείριση μοτέρ ισχύος, επέκταση αποθηκευτικού χώρου μέσω καρτών SD ή usb, καθώς και πολλές άλλες δυνατότητες ανάλογα τιε επιθυμίες του χρήστη.

Το θετικό είναι ότι δεν υπάρχει περιορισμός στην χρήση των επεκτάσεων μέρος των οποίων παρουσιάζεται αναλυτικά παρακάτω.

i. Arduino GSM Shield

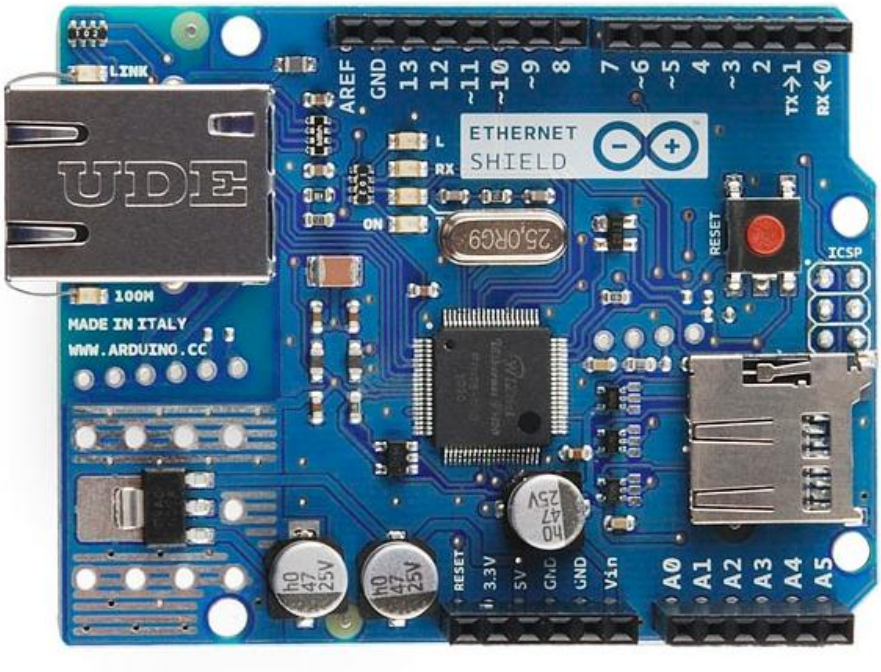
Εικόνα 4 – Arduino GSM Shield

Το Arduino GSM shieldεπιτρέπει στο Arduino να συνδεθεί στο διαδίκτυο, εκτελεί και λαμβάνει φωνητικές κλήσεις και αποστέλλει/λαμβάνει μηνύματα SMS. Για τη διασύνδεση με το δίκτυο κινητής τηλεφωνίας, απαιτείται η ύπαρξη μιας κάρτας SIM που παρέχεται από τον λειτουργό του δικτύου.

Το Παγκόσμιο Σύστημα Κινητών Επικοινωνιών (Global System for Mobile communications συντ. GSM) που χρησιμοποιεί για τη διασύνδεση είναι ένα κοινό Ευρωπαϊκό ψηφιακό σύστημα κινητής τηλεφωνίας.

Το GSM είναι ένα κυψελοειδές ψηφιακό σύστημα κινητής τηλεφωνίας δεύτερης γενιάς (2G). Χρησιμοποιεί ηλεκτρομαγνητικά σήματα και την τεχνική πολλαπλής πρόσβασης με διαχωρισμό του διαθέσιμου φάσματος συχνοτήτων σε ένα πλήθος καναλιών και την διαίρεση αυτών σε χρονοθυρίδες για την μετάδοση των σημάτων.

ii. Arduino Ethernet Shield



Εικόνα 5 - ArduinoEthernetShield

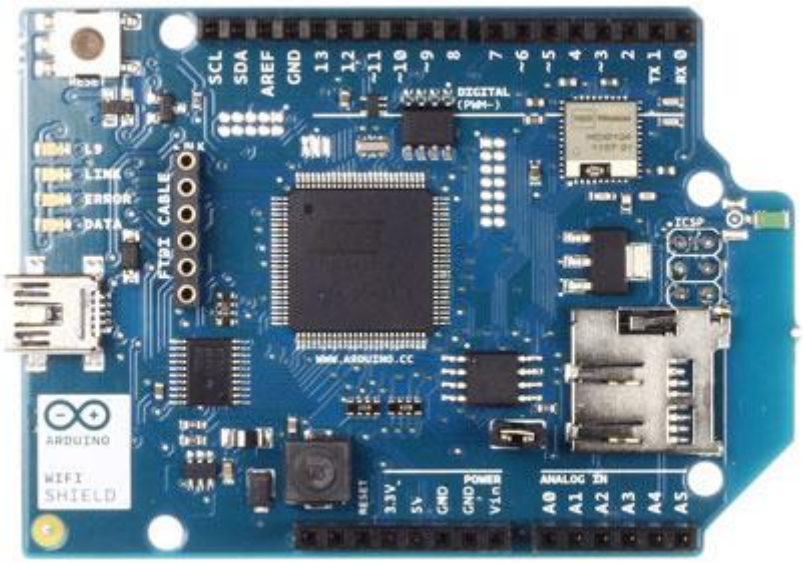
Το Arduino Ethernet shield συνδέει το Arduino στο διαδίκτυο σε λίγα λεπτά. Συνδέοντας απλώς αυτό το shieldστο Arduino μπορείτε να συνδεθείτε στο δίκτυο με ένα καλώδιο.

Επιπλέον δίνει την δυνατότητα επέκτασης μνήμης μέσω του ενσωματωμένου microSD cardslot ώστε ο χρήστης να έχει τη δυνατότητα να κάνει διαμοιρασμό των αρχείων που έχει αποθηκεύσει στην κάρτα μνήμης, μέσω δικτύου.

Τα τεχνικά χαρακτηριστικά του shield είναι:

- Ethernet Controller: W5100 με εσωτερική 16K ρυθμιστικό
- Ταχύτητα σύνδεσης: 10/100Mb
- Σύνδεση με Arduino στη θύρα SPI.
- Onboard micro-SD card slot

iii. Arduino WiFi Shield



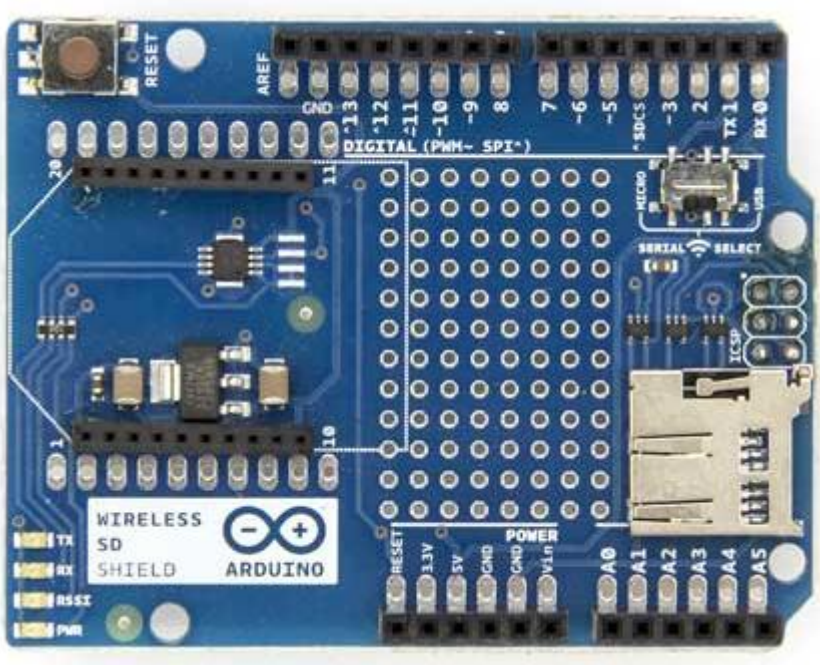
Εικόνα6- Arduino WiFi Shield

Το Arduino WiFi συνδέει το Arduino στο διαδίκτυο με ασύρματο τρόπο. Η λογική είναι η ίδια με αυτή τουEthernet shield απλά στην προκειμένη περίπτωση συνδεόμαστε ασύρματα στο δίκτυο.

Τα τεχνικά χαρακτηριστικά του shield είναι:

- Σύνδεση μέσω: 802.11b/g δίκτυα
- Τύποι κρυπτογράφησης: WEP και WPA2 Personal
- Σύνδεση με Arduino στη θύρα SPI
- Σύνδεση FTDI για σειριακό εντοπισμό σφαλμάτων του WiFi
- Mini-USB για την ενημέρωση WiFi ασπίδα firmware

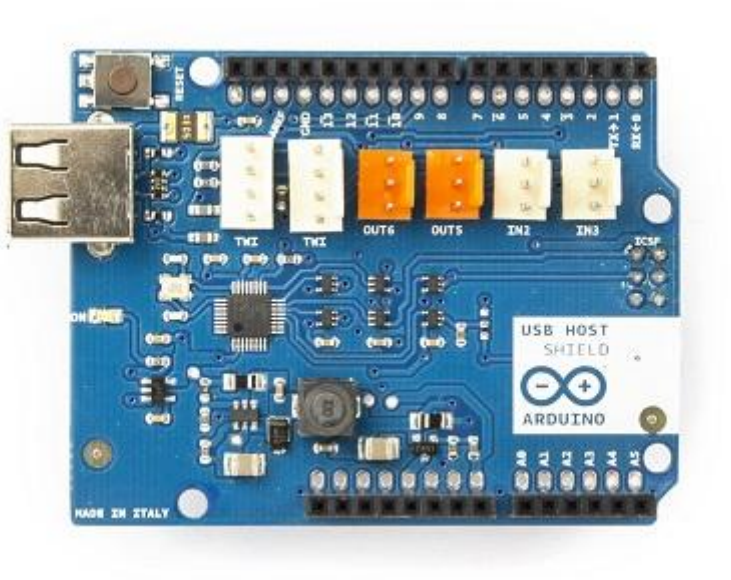
iv. Wireless SD Shield



Εικόνα7 – Arduino Wireless SD Shield

Το Wireless SD shield επιτρέπει στο Arduino την ασύρματη επικοινωνία, χρησιμοποιώντας μια ασύρματη μονάδα. Βασίζεται στις ενότητες Xbee από την Digi, αλλά μπορεί να χρησιμοποιήσει οποιαδήποτε ενότητα με το ίδιο αποτύπωμα. Η μονάδα μπορεί να επικοινωνήσει μέχρι 100 μέτρα σε εσωτερικούς χώρους ή 300 πόδια σε εξωτερικούς χώρους (με line-of-sight). Μπορεί να χρησιμοποιηθεί ως αντικατάσταση σειριακή/usb.

v. Arduino USB Host Shield



Εικόνα7 – Arduino USB Host Shield

Το Arduino USB Host Shield επιτρέπει να συνδεθεί μια συσκευή USBστο Arduino. Βασίζεται στον ελεγκτή MAX3421E ο οποίος είναι ένας ελεγκτής USB περιφερειακών/υποδοχής που περιέχει την ψηφιακή λογική και το αναλογικό κύκλωμα που απαιτείται για την εφαρμογή full-speed USB περιφερειακών ή πλήρους υποδοχής/χαμηλής ταχύτητας συμβατές με USB προδιαγραφές rev 2.0. Το shield είναι TinkerKit συμβατό, πράγμα που σημαίνει ότι μπορείτε να δημιουργήσετε γρήγορα κάποιο έργο, συνδέοντας μονάδες TinkerKit πάνω στην πλακέτα.

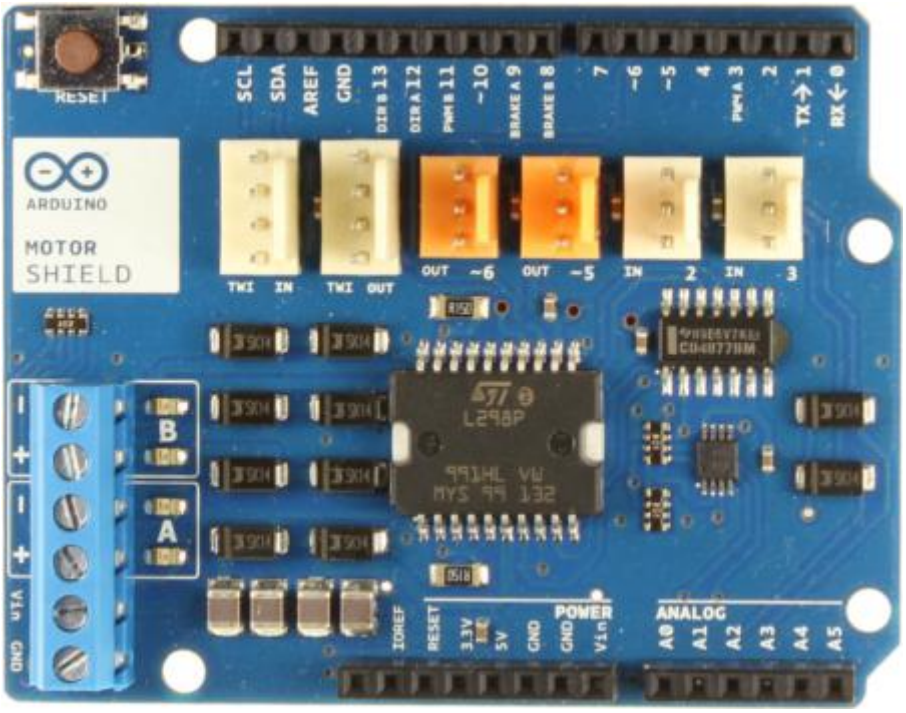
Οι ακόλουθες κατηγορίες συσκευών υποστηρίζονται από το shield:

- HID συσκευές: πληκτρολόγια, ποντίκια, χειριστήρια, κλπ.
- Ελεγκτές παιχνιδιών: Sony PS3, Nintendo Wii, Xbox360.
- USB σε σειριακή μετατροπείς: FTDI, PL-2303, ACM, καθώς και ορισμένα κινητά τηλέφωνα και δέκτες GPS.
- ADK-ικανό Android τηλέφωνα και τραπέζια.
- Ψηφιακές φωτογραφικές μηχανές: Canon EOS, Powershot, Nikon φωτογραφικές μηχανές DSLR και P & S, καθώς και γενικές PTP.

- Συσκευές μαζικής αποθήκευσης: USB sticks, συσκευές ανάγνωσης καρτών μνήμης, εξωτερικοί σκληροί δίσκοι, κλπ
- Dongles Bluetooth.

Το Arduino επικοινωνεί με το MAX3421E χρησιμοποιώντας τον δίαυλο SPI (μέσα από την επικεφαλίδα ICSP).

vi. Arduino Motor Shield



Εικόνα 8 - ArduinoMotorShield

Το Arduino Motorshield βασίζεται στο L298, το οποίο είναι μια γέφυρα διπλού οδηγού σχεδιασμένη για να οδηγεί επαγωγικά φορτία όπως relay, πηνία, DC και step-up κινητήρες. Αυτό επιτρέπει την οδήγηση δύο κινητήρων συνεχούς ρεύματος με Arduino, ελέγχοντας ξεχωριστά την ταχύτητα και την κατεύθυνση του καθενός. Επίσης, έχει τη δυνατότητα να μετρήσει το ρεύμα απορρόφησης κάθε κινητήρα, μεταξύ άλλων χαρακτηριστικών γνωρισμάτων.

8 Αισθητήρες

Τα διάφορα «ερεθίσματα» από το περιβάλλον η πλατφόρμα του arduino μπορεί να τα αντιλαμβάνεται χρησιμοποιώντας μια σειρά από αισθητήρες που παρατίθενται παρακάτω.

Ανάλογα τον τύπο του αισθητήρα δίνουμε τη δυνατότητα στην πλατφόρμα να αποκτήσει ιδιότητες όπως,

- ✓ Μέτρηση της θερμοκρασίας, φωτεινότητας
- ✓ υπολογισμός στάθμης υγρού σε ένα δοχείο
- ✓ διαρροή αερίου
- ✓ Ηχητική ειδοποίηση

a. Ultrasonic sensor

Εικόνα 9 - Ultrasonic*sensor*

Ο αισθητήρας υπερήχων HC-SR04 είναι ένας αισθητήρας μέτρησης της απόστασης. Ουσιαστικά αποτελεί τα «μάτια» του Arduino καθώς του επιτρέπει τη χωρική συνειδητοποίηση.

b. Gas sensor

Εικόνα 10 - Gassensor

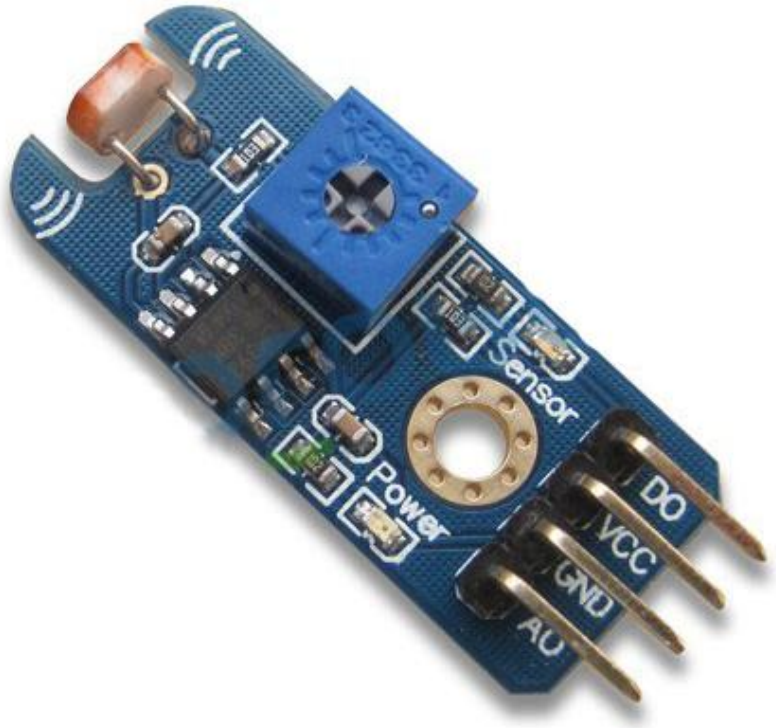
Η σειρά MQ των αισθητήρων αερίου χρησιμοποιεί ένα μικρό "καλοριφέρ" με έναν ηλεκτροχημικό αισθητήρα. Μπορεί να χρησιμοποιηθεί μόνο σε εσωτερικό χώρο και σε θερμοκρασία δωματίου. Η ευαισθησία του αισθητήρα βαθμονομείται σε αντιστοιχία το είδος του αερίου μέσω του ενσωματωμένου trimmer.Η έξοδος είναι ένα αναλογικό σήμα και μπορεί να διαβαστεί με μια αναλογική είσοδο του Arduino.

c. Temperature sensor

Εικόνα 11 – Temperature sensor

Ο αισθητήρας θερμοκρασίας χρησιμοποιεί ένα transistor lm35. Το lm35 έχει 3 Pins, Vin,Vout, Ground. Στο Vinδίνεται τάση από 4V έως 8V και στο Vout εμφανίζονται τάσεις που αντιστοιχούν στην θερμοκρασία του χώρου. Παραδείγματος χάριν τα 0.20Vαντιστοιχούν σε 20^{o}C.

d. Light sensor



Εικόνα 12 – Light sensor

Ο αισθητήρας φωτός διαθέτει μια φωτό-αντίσταση ως ελεγκτή, η οποία ανάλογα με την φωτεινότητα μεταβάλει την χωρητικότητά της και μπορεί να χρησιμοποιηθεί ως διακόπτης για αυτόματο άνοιγμα/κλείσιμο των λαμπτήρων.

e. Sound sensor

Εικόνα 13 - Soundsensor

Ο αισθητήρας ήχου διαθέτει ένα μικρόφωνο ως ελεγκτή, το οποίο ανάλογα αν υπάρχει ήχος ή όχι μπορεί να οπλίζει ή να αφοπλίζει αντιστοίχως ένα relayπου θα ανοιγοκλείνει κάποια συσκευή.

Παρέχει τη δυνατότητα ρύθμισης της ευαισθησίας μέσω του ενσωματωμένου trimmerπου διαθέτει.

9 Λογισμικό

Το Integrated Development Environment (IDE) του Arduino είναι αναπτυξιακό περιβάλλον λογισμικού ανοικτού κώδικα και έχει τη δυνατότητα να τρέξει σε ποικίλες πλατφόρμες είναι δε βασισμένο στη γλώσσαJava. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστή, και έχει την ικανότητα να φορτώνει με ευκολία το πρόγραμμα από τον υπολογιστή στην πλακέτα διαμέσου μιας σειριακής θύρας.

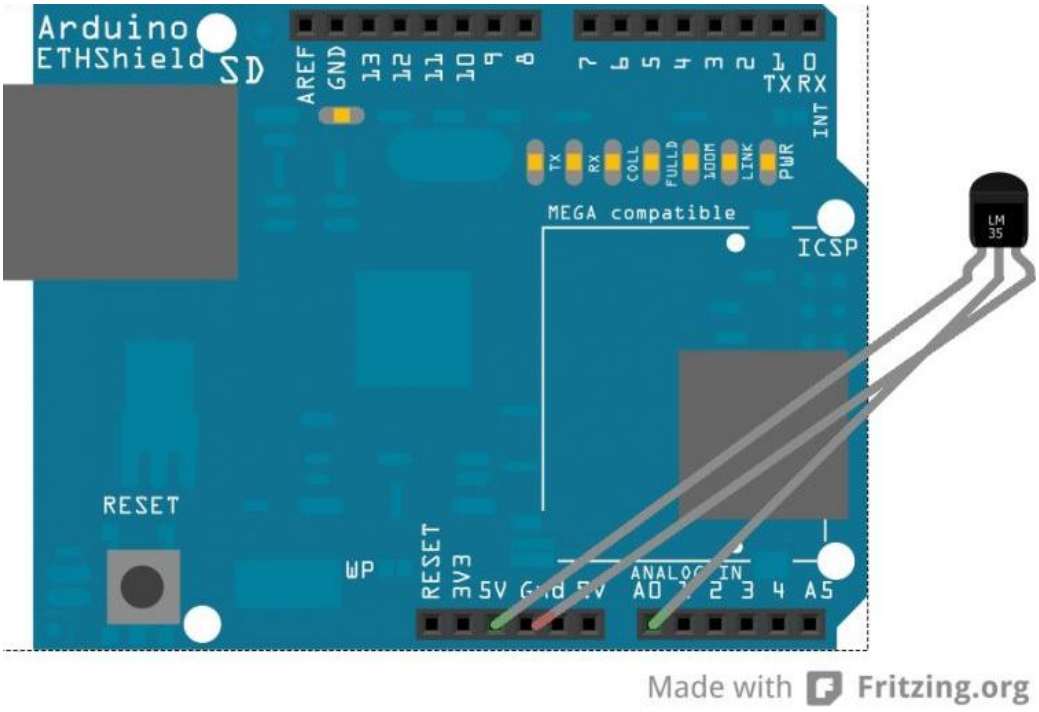
Εικόνα 14- Γραφικό περιβάλλον Arduino

Το περιβάλλον ανάπτυξης βασίζεται στην Processing. Η Processingαποτελεί ένα περιβάλλον ανάπτυξης που έχει σχεδιαστεί με τέτοιο τρόπο ώστε να εισαγάγει στον προγραμματισμό νέους χρήστες, μη εξοικειωμένους με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring, μια γλώσσα που έχει κοινά στοιχεία με τη γλώσσα C, η οποία παρέχει παρόμοια λειτουργικότητα για μια πλακέτα με κάπως πιο περιορισμένη σχεδίαση, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στην Processing.

10 Η λογική της σχεδίασης του συστήματος

Η λογική είναι η εξής:

Οι αισθητήρες παίρνουν δεδομένα και τα στέλνουν στο Arduino, το οποίο με τη σειρά του τα ανεβάζει μέσω της Ethernetshield σε έναν HTTP Server.



Εικόνα 15 - Arduino Ethernet Shield

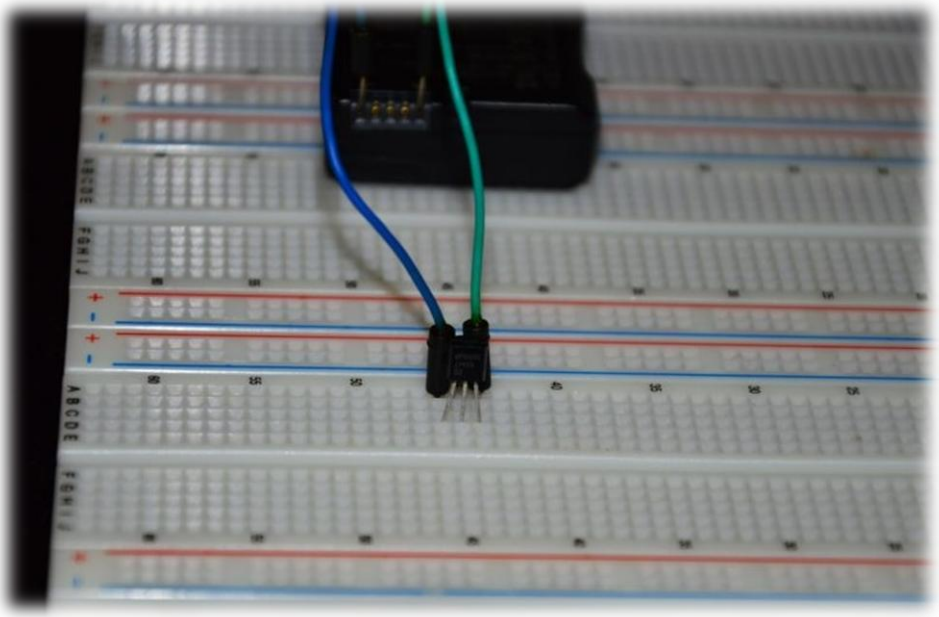
11 Προγραμματισμός arduino ώστε να επικοινωνεί με τον αισθητήρα θερμότητας

Όλη η διαδικασία είναι χωρισμένη σε επιμέρους τμήματα, αφενός για να είναι πιο εύκολο να στηθεί η τελική πλατφόρμα και αφετέρου για να είναι ευκολότερη η διαχείριση. Έτσι λοιπόν το πρώτο στάδιο είναι να γίνουν οι απαραίτητες ενέργειες όσον αφορά το hardware για να συνδεθεί ο ελεγκτής με τον αισθητήρα θερμότητας και ακολούθως να συγγραφεί ο απαραίτητος κώδικας για την καταγραφή θερμότητας στο χώρο που έχει εγκατασταθεί ο αισθητήρας.

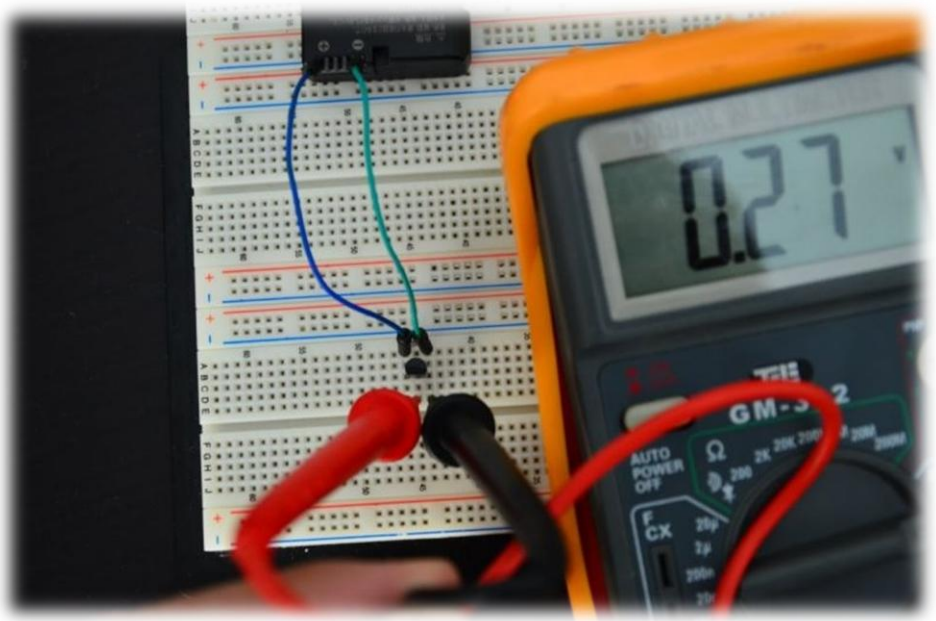
Ο LM35 είναι ένας αισθητήρας θερμοκρασίας, ο οποίος έχει 3 pins, ένα V(in), ένα V(out), και ένα Ground (βλ. εικόνα 17).

Εικόνα 16 - LM35

Ως τάση εξόδου V(in)μπορεί να δεχτεί από 4V μέχρι 20V. Η τάση εξόδου V(out) που δίνει εξαρτάται από την θερμοκρασία. Για παράδειγμα τάση εξόδου 0.20V αντιστοιχεί σε 20 βαθμούς Κελσίου.



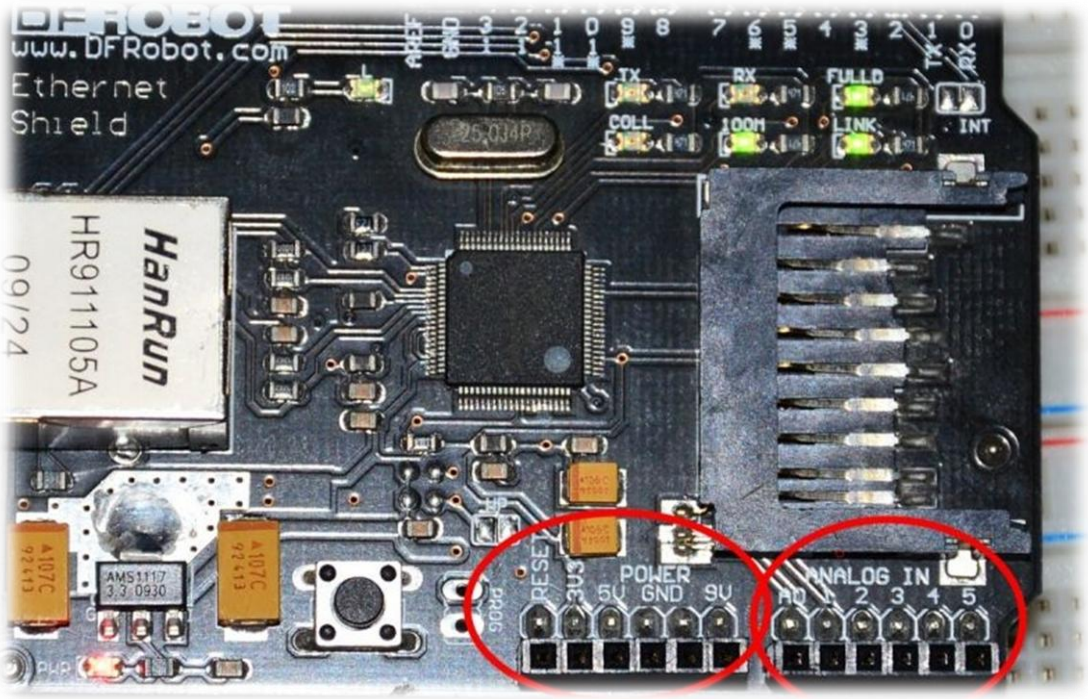
Εικόνα 17 - Συνδέσεις



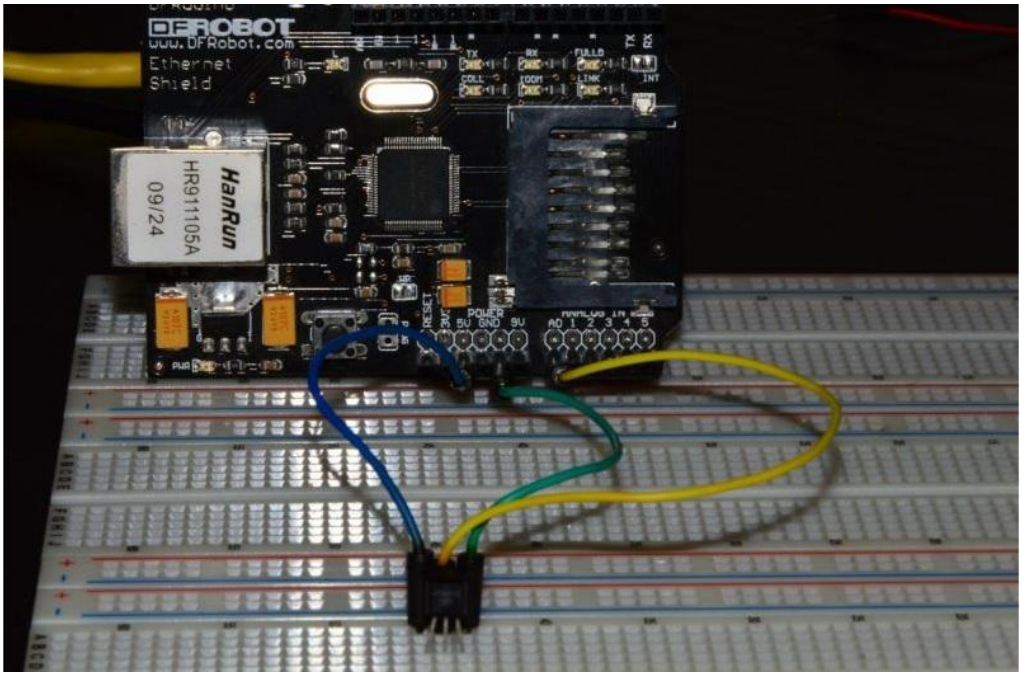
Εικόνα 18 - Μετρήσεις

Για να συνδεθεί ο αισθητήρας που φαίνεται παραπάνω (LM35) στο Arduino, λαμβάνουμε την τάση εισόδου από την πλακέτα και την τάση εξόδου στέλνουμε πίσω σε αυτήν.

Δίνονται5V στα ακριανά pins του LM35 (μπλε και πράσινο καλώδιο στην εικόνα 20), και παίρνεται η έξοδος από το μεσαίο pin, η οποία οδηγείται στο Analog Input 0 του Arduino.

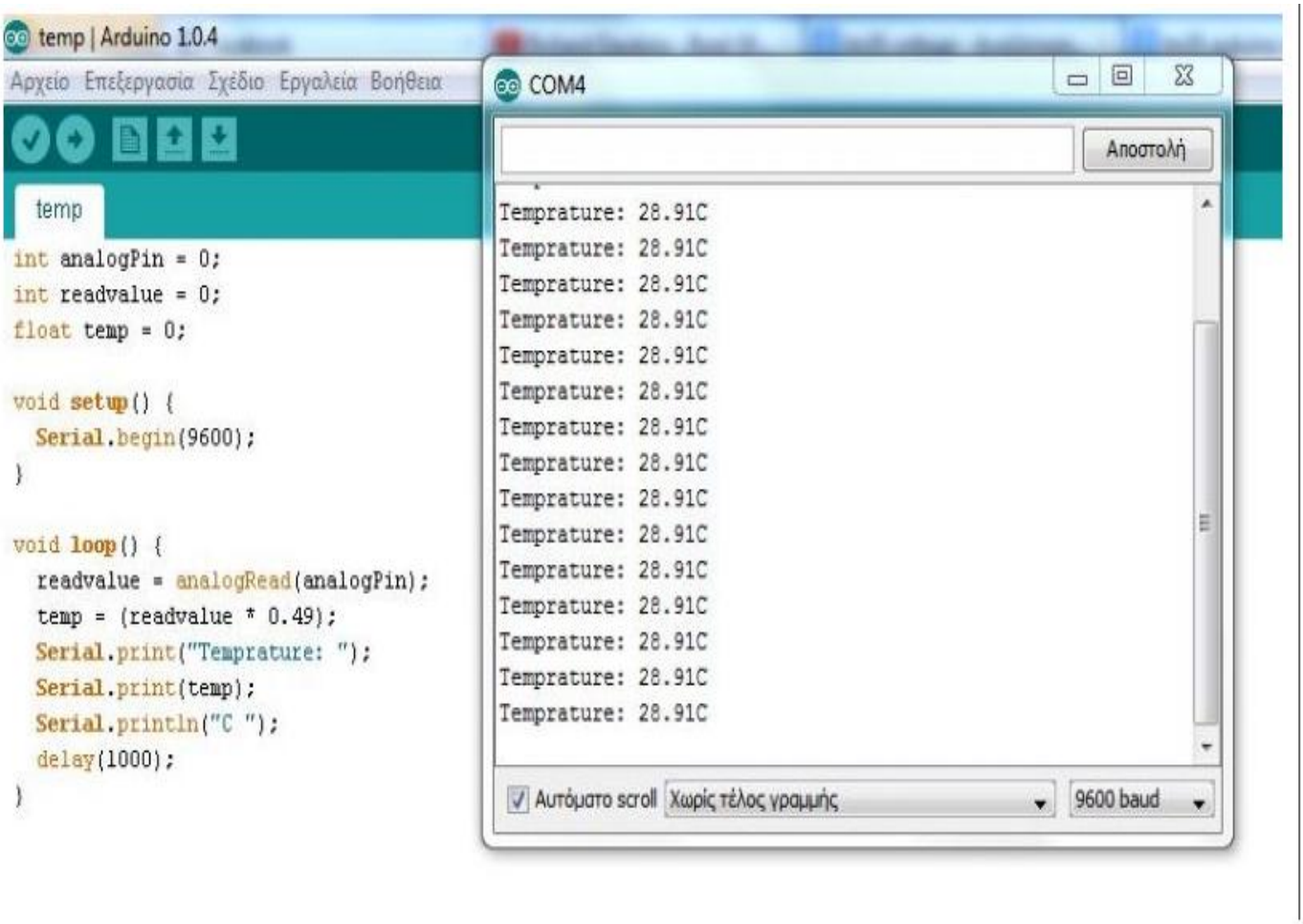


Εικόνα 19 - Πλακέτα



Εικόνα 20 - Συνδέσεις πλακέτας

Στην εικόνα 22παρουσιάζεται η ένδειξη του αισθητήρα στο Serial output του υπολογιστή και το αντίστοιχο μέρος κώδικα.

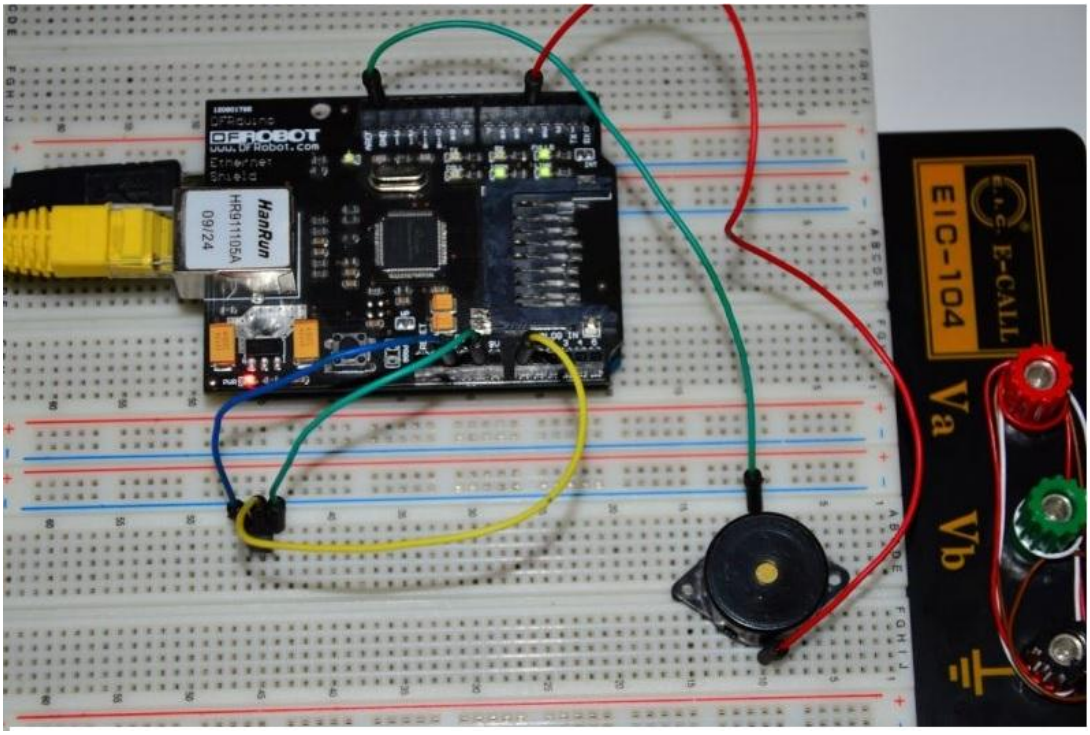


Εικόνα 21 - Προγραμματισμός ελεγκτή& Εμφάνιση τιμών θερμοκρασίας

Όπως φαίνεται και στο παραπάνω σχήμα, ο ελεγκτής έχει προγραμματιστεί με επιτυχία να μετρά τη θερμοκρασία του χώρου και να μεταφέρει τις τιμές στην οθόνη του υπολογιστή.

Ο λόγος που προγραμματίστηκε ο ελεγκτής να κάνει καταγραφή της θερμοκρασίας είναι κατά πρώτον για να επιτρέπεται να του ορίζεται πότε θα ανοίγει ο κλιματισμός και κατά δεύτερον για να ενεργοποιείται το σύστημα πυρόσβεσης σε περίπτωση που υπάρξει φωτιά.

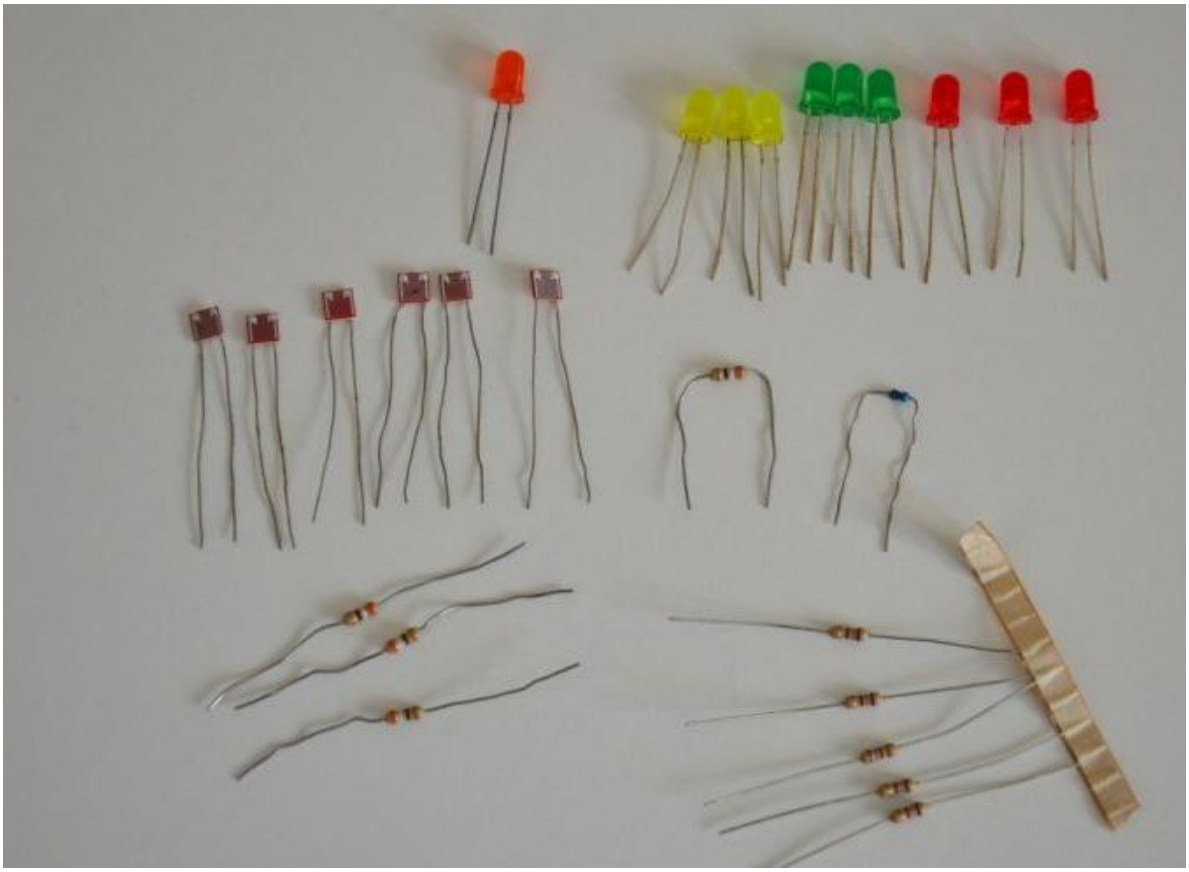
Πειραματικά προστέθηκαν στον παραπάνω κώδικα οι απαραίτητες εντολές έτσι ώστε όταν η θερμοκρασία ξεπεράσει τους 30 βαθμούς Κελσίου να ηχεί ένα buzzer, το οποίο συνδέθηκε στο digital pin 4 και στο ground. Σε πρακτική εφαρμογή φυσικά θα αντικατασταθεί με ένα relayισχύος το οποίο θα ενεργοποιεί και θα απενεργοποιεί αντίστοιχα τον κλιματισμό.



Εικόνα 22 - Συνδέσεις Arduino

12 Προγραμματισμός arduino ώστε να επικοινωνεί με τον αισθητήρα φωτός

Ακολούθως, προστίθεται η χρήση ενός ακόμα αισθητήρα, και συγκεκριμένα ενός photoresistor που μετράει το φως σε ένα δωμάτιο.

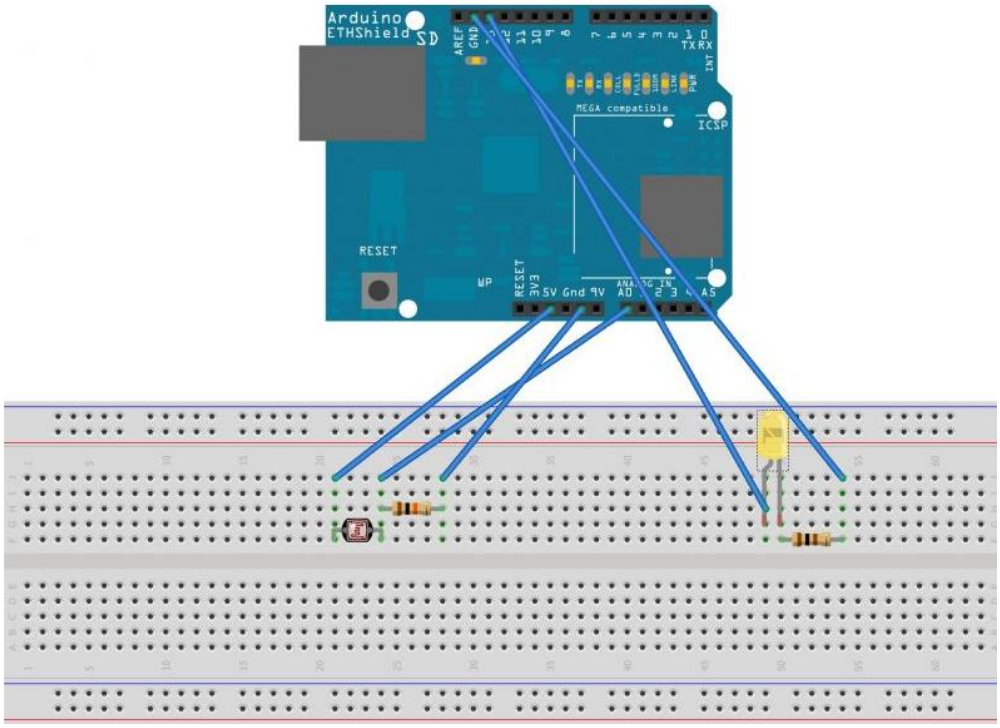


Εικόνα 23- Εξαρτήματα (Αντιστάσεις, Φωτοαντιστάσεις και LED)

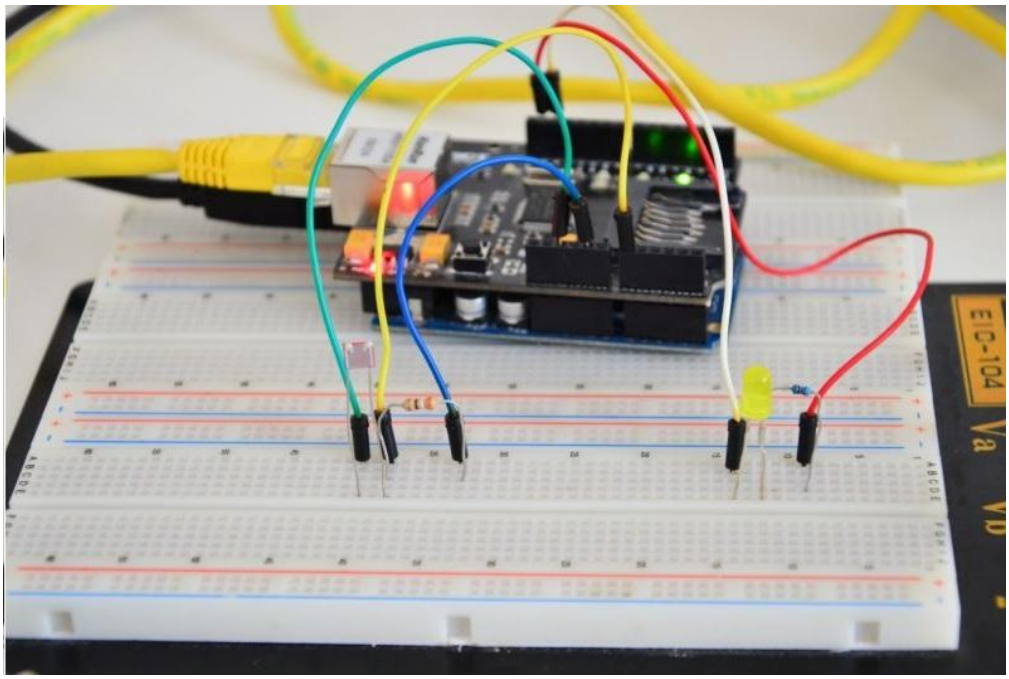
Σε αυτό το σημείο είναι επιθυμητό να φτιάξουμε ένα σύστημα, το οποίο θα μετρά την φωτεινή ένταση σε ένα δωμάτιο. Αν η τιμή της πέσει κάτω από κάποιο όριο που οριστεί, θα ενεργοποιεί αυτόματα τον φωτισμό του δωματίου (τεχνητό ή φυσικό).

Στην πιλοτική πειραματική διάταξη ενεργοποιείται ένα led, ενώ στην πλήρη ανάπτυξη θα στέλνεταιtrigger 5V σε ένα ac relay ώστε να ανάβει τα φώτα σε ένα δωμάτιο ή να ανοίγει τα καλύμματα των παραθύρων ή θα κλείνει τα σκίαστρα.

13 Υλοποίηση κυκλώματος



Εικόνα 24 - Συνδεσμολογία κυκλώματος (α)



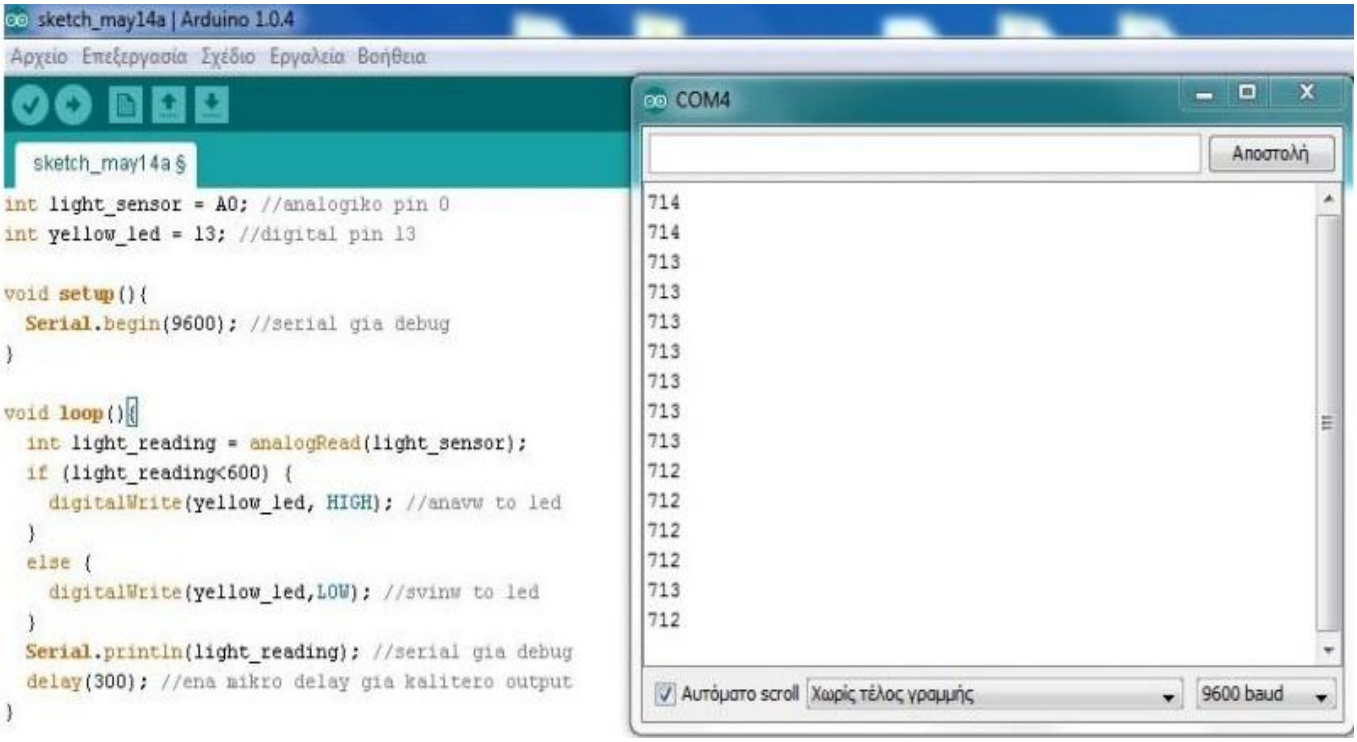
Εικόνα 25 - Συνδεσμολογία κυκλώματος (β)

Αρχικά δηλώνονται οι μεταβλητές light_sensor_1 και led στις οποίες αντιστοιχείτο αναλογικό Pin1, και το ψηφιακό PIN 13 του Arduino.

Στην συνέχεια στη μεταβλητή light_read, τοποθετείται η μέτρηση που διαβάζεται από τον αισθητήρα φωτός, που είναι συνδεδεμένος στο αναλογικό Pin0.

Ακόμα, ορίζεται αν η τιμή της μεταβλητής πέσει κάτω από ένα όριο if(light_reading<600), να ανάβει το LED, αλλάζοντας την κατάστασή του σε “high”.

Τέλος, ορίζεται να εμφανίζεται η τιμή της μεταβλητής light_reading στο Serial Monitor για Debugging.



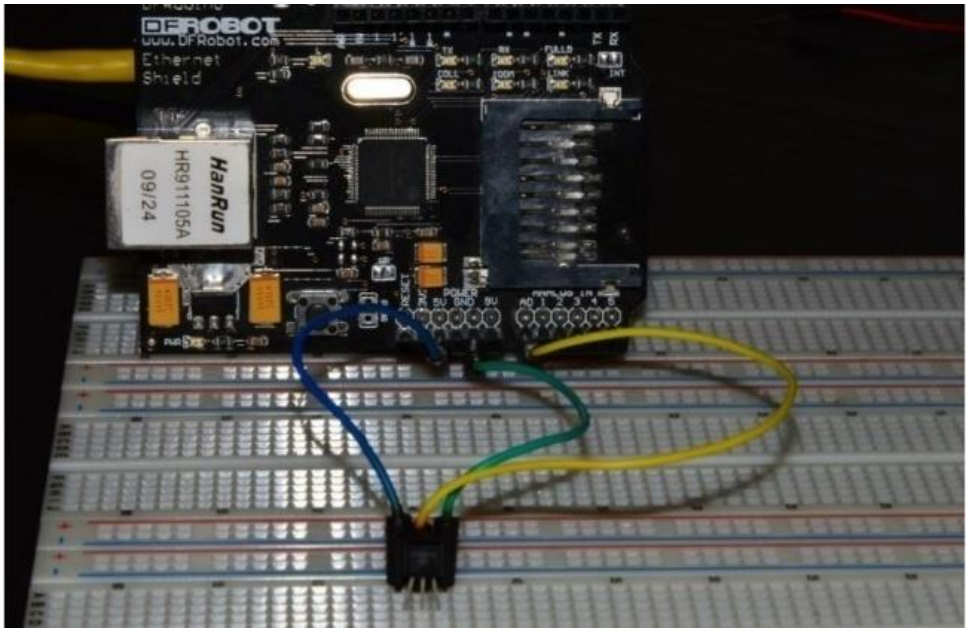
Εικόνα 26 - Στιγμιότυπο προγραμματισμού

Το τελευταίο βήμα που απομένει για τον ελεγκτή είναι να επικοινωνεί με το routerώστε να είναι εφικτή η διαχείριση των δυνατοτήτων που έχουν εγκατασταθεί μέσω της εφαρμογής που θα αναπτυχθεί..

Έτσι λοιπόν, επιτρέπεται να δημιουργήσουμε ένα HTTP server μέσω ενός shield που παρέχει η κατασκευάστρια εταιρεία, στο οποίο θα παρέχονται οι πληροφορίες από τους αισθητήρες.

Συνδέεται το Arduino στον δρομολογητή με καλώδιοEthernet, στήνεται ο WebServer, και προγραμματίζεται το Arduino να δείχνει την ένδειξη από τον αισθητήρα θερμότητας.

Πλέον, ανοίγοντας έναν οποιονδήποτε περιηγητή είναι εφικτή η πρόσβαση στην IP διεύθυνση που ορίστηκε στο Arduino, για να ελεγχθεί η ένδειξη της θερμοκρασίας.



Εικόνα 27 - Συνδεσμολογίες (α)

14 Στήσιμο του Δικτύου διαχείρισης – επικοινωνίας μεταξύ Arduino και συσκευών

Το Arduino με την Ethernet shield πρέπει να συνδεθεί στο router του σπιτιού με καλώδιο Ethernet. Στην περίπτωση όμως που κάτι τέτοιο δεν είναι επιθυμητό, υπάρχει η δυνατότητα να χρησιμοποιηθεί ένα A/P σε client mode, ώστε το Arduino να αποκτήσει πρόσβαση στο δίκτυο από "απόσταση".

Φυσικά πριν συνδεθεί ασύρματα το A/P σε client mode πρέπει να ληφθούν μέτρα για την ασφάλεια του ασύρματου δικτύου, καθώς δεν είναι επιθυμητό ο καθένας να έχει πρόσβαση στο αναπτυχθέν σύστημα.

Εικόνα 28 - Συνδεσμολογίες (β)

15 Τροφοδοσία

Το Arduino τροφοδοτείται από την θύρα USB του υπολογιστή κατά τη διάρκεια του προγραμματισμού του. Η χρήση του όμως (αφού προγραμματιστεί) θα πρέπει να είναι ανεξάρτητη από τον υπολογιστή. Συνεπώς πρέπει να πραγματοποιηθεί η τροφοδοσία του με διαφορετικό τρόπο.

Η μία επιλογή είναι να χρησιμοποιηθεί η κλασσική διεπαφή που υπάρχει σχεδόν σε όλα τα κινητά, η οποία δίνει σύνδεση USB female. Εναλλακτικά μπορεί να χρησιμοποιηθεί τροφοδοτικό DC 12V, με βύσμα 2.1mm.

Όποια και από τις δύο εναλλακτικές και αν επιλεγεί, θα πρέπει να είναι ικανή να δώσει ρεύμα τουλάχιστον 500mA έως 1A, καθότι στην συνέχεια πρέπει να τροφοδοτηθούν διάφορα ηλεκτρονικά κυκλώματα από το Arduino.

Εικόνα 29 - Τροφοδοτικά

16 Σύνδεση relayμε συσκευές

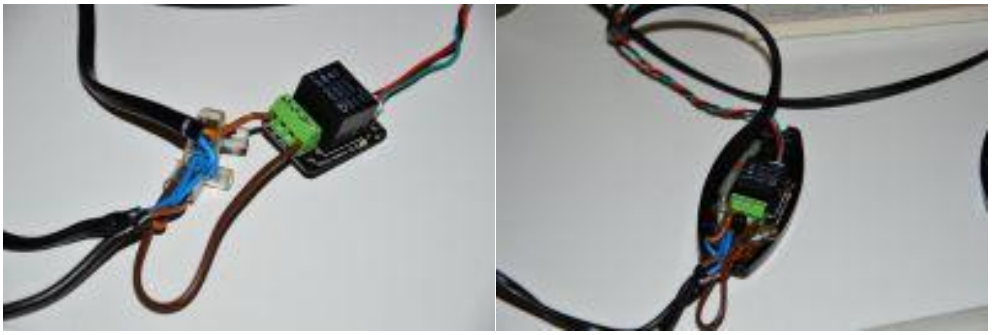
Η σύνδεση των relay με τις συσκευές που κάνουν χρήση 220V AC είναι μια απλή συνδεσμολογία αλλά απαιτεί μεγάλη προσοχή δεδομένου ότι μια λανθασμένη σύνδεση μπορεί να προκαλέσει τραυματισμό ή να δημιουργήσει πυρκαγιά από κάποιο βραχυκύκλωμα.

Το relayσυνδέεται με τρία καλώδια στο Arduino. Το μαύρο καλώδιο συνδέεται στο PIN GND, το πράσινο στο PIN τροφοδοσίας +5V, ενώ το κόκκινο στο PIN 6 το οποίο θα λειτουργεί ως trigger.

Όταν ορίζεται η κατάσταση του PIN6 ως "HIGH", το οποίο δίνει +5V, το relay κλείνει το κύκλωμα, άρα αρχίζει να λειτουργεί η συσκευή. Αντιθέτως ορίζοντας την κατάστασή του ως "LOW", το relay ανοίγει το κύκλωμα και η συσκευή απενεργοποιείται.

Για να συνδεθεί μια συσκευή που διαθέτει διακόπτη ακολουθείται η εξής διαδικασία:

- Αρχικά αποσυνδέεται η συσκευή από την τροφοδοσία.
- Στην συνέχεια ανοίγεται η θήκη του διακόπτη και αφαιρείται ο μηχανισμός.
- Έπειτα για συγκεκριμένη τοποθέτηση του φις, εντοπίζεται το καλώδιο φάσης και το neutral.
- Ακολούθως πρέπει να "παρακαμφθεί" το καλώδιο φάσης, το οποίο συνδέεται στο relay. Το καλώδιο που "έρχεται" από την πρίζα, συνδέεται στο "NO", ενώ αυτό που πηγαίνει στην συσκευή συνδέεται στο COM.



Εικόνα 30 - Συνδέσεις

17 Γραφική απεικόνιση του server

Για την είσοδο στην εφαρμογή χρειάζεται απλώς ασύρματη ή ενσύρματη σύνδεση στο δίκτυο και η πληκτρολόγηση στο browser της διεύθυνσης του webserver, δηλαδή«192.168.1.220».

Τότε εμφανίζεται στην οθόνη η παρακάτω εικόνα με τα sliders μέσω των οποίων ενεργοποιούνται ή απενεργοποιούνται οι συνδεδεμένες συσκευές.

Εικόνα 31 - Σύνδεση στο δίκτυο

18 Προοπτικές εξέλιξης

Η υλοποίηση που παρουσιάστηκε πρέπει να θεωρείται πιλοτική και έχει αρκετές προοπτικές εξέλιξης καθώς και επιδέχεται αρκετών βελτιωτικών τροποποιήσεων.

Η απλούστερη αλλαγή που μπορεί να εφαρμοστεί είναι να προστεθούν ή να αφαιρεθούν διαχειριζόμενες συσκευές ανάλογα με τις απαιτήσεις του εκάστοτε συστήματος και ανάλογα με τις ανάγκες των ανθρώπων που αυτό θα εξυπηρετεί. Για παράδειγμα, θα μπορούσε να συνδεθεί σύστημα συναγερμού, ώστε να μπορεί να οπλιστεί/αφοπλιστεί από απόσταση. Επιπλέον θα μπορούσε να συνδεθεί ένα κλειστό κύκλωμα παρακολούθησης ώστε να υπάρχει οπτική πρόσβαση σε αυτό οποιαδήποτε χρονική στιγμή. Ένα τέτοιο σύστημα θα συνεισέφερε σημαντικά στην καθημερινότητα ενός ατόμου με προβλήματα ακοής, στην περίπτωση που κάποιος τον επισκεπτόταν στο σπίτι του, καθώς ο κάτοικος θα μπορούσε να βλέπει κάποια οπτική (φωτεινή) ειδοποίηση όταν ο επισκέπτης χτυπάει το κουδούνι.

Εφικτή θα ήταν η δημιουργία μιας δικλίδας ασφαλείας για την σύνδεση στον serverώστε να μην μπορεί να συνδεθεί οποιοσδήποτε χρήστης στο σύστημα. Εφόσον υπάρξει τέτοιου είδους ασφάλεια θα μπορέσει να συνδεθεί η πλατφόρμα του server με μια εφαρμογή γραφικής απεικόνισης android ώστε να επιτευχθεί αμεσότερη πρόσβαση στο σύστημα.

Επίσης με απλές μετατροπές θα ήταν εύκολο να τροποποιηθεί ένα αυτοκίνητο ώστε να είναι ελεγχόμενο μέσω μιας τηλεφωνικής συσκευής, για άτομα με κινητικές προβλήματα.

Με τη χρήση σερβομηχανισμών και αισθητήρων δύναται να διαχειριστούμε τις μονάδες «body» που αναλαμβάνουν τη διαχείριση του φωτισμού του οχήματος και γενικότερα των συσκευών που βρίσκονται στην καμπίνα.

Με χρήση κατάλληλων αισθητήρων υπερύθρων και sonar δίνεται στην πλατφόρμα του arduino η δυνατότητα να μετρά την απόσταση του οχήματος από κάποιο εμπόδιο ή προπορευόμενο όχημα και να επιβραδύνει ή να σταματά εντελώς το όχημα.

Φυσικά μια τέτοια υλοποίηση μπορεί να πραγματοποιηθεί σε πειραματικό επίπεδο μόνο, μιας και τίθενται πολλά θέματα ασφαλείας. Επιπρόσθετα η ανθρώπινη παρέμβαση και επαγρύπνηση κατά την οδήγηση θα είναι απαραίτητη μιας και η εφαρμογή θα αφορά σε σύστημα υποβοήθησης και όχι αντικατάστασης του οδηγού.

19 Συμπεράσματα

Η τεχνολογία προχωρά με γοργούς ρυθμούς και με ορθή χρήση της μπορούμε να διευκολύνουμε συνεχώς την καθημερινότητά μας.

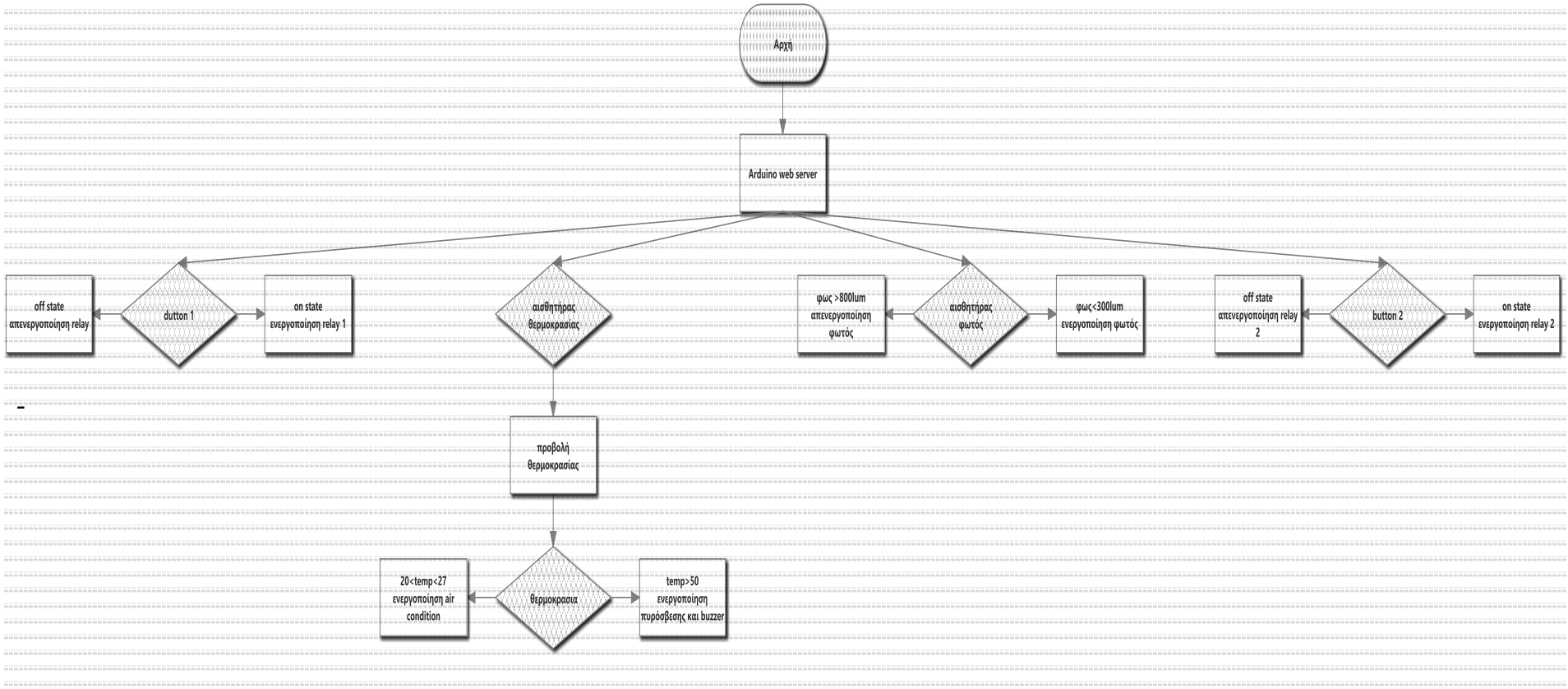
Ιδιαίτερη σημασία όμως έχει να διευκολύνουμε την καθημερινότητα ανθρώπων με ειδικές ανάγκες. Με την παρουσιασθείσα πιλοτική υλοποίηση, πέραν της επιδεικνυόμενης τεχνολογίας, επιθυμούμε να διαδώσουμε το μήνυμα η τεχνολογία πρέπει να αποσκοπεί στην ισότητα και την ισοτιμία των μελών μιας κοινωνίας.

Η μελέτη κατέδειξε ότι με χαμηλό κόστος υλικών, την επιλογή της χαμηλού κόστους πλατφόρμας του Arduino, με τη μεγάλη ποικιλία από επεκτάσεις και τη δυνατότητα προγραμματισμού της ακόμη και από άτομα που δεν είναι ειδικοί, επιτυγχάνεται η σχεδίαση και ανάπτυξη έξυπνου συστήματος διαχείρισης οικιακής ηλεκτρικής εγκατάστασης για άτομα με ειδικές ανάγκες, το οποίο μπορεί με ευκολία να τροποποιείται και να προσαρμόζεται προσθέτοντας νέους αισθητήρες ή ενεργοποιητές, ώστε να ανταποκρίνεται στις όποιες επιμέρους απαιτήσεις των Α.με.Α.

Επίσης το μικρό μέγεθος της κατασκευής επιτρέπει την χωρίς περαιτέρω έξοδα προσαρμογή στον ηλεκτρολογικό πίνακα.

20 Υπόμνημα Κώδικα

20.1Διάγραμμα ροής προγράμματος -Ψευδοκώδικας



Εικόνα 32 - Διάγραμμα ροής προγράμματος

Start //Αρχή προγράμματος

Go to ardunio server // παραπέμπουμε στηχρήση το server

Run

Loop(1) //επανάληψη (1)

Check tempsensor //έλεγχος θερμοκρασίας

If(temp>50 Cdegrees) // αν η θερμοκρασία είναι μεγαλύτερη από 50 βαθμούς

Buzzerstart and printthemessage «Προσοχή υψηλή θερμοκρασία» // ηχεί το buzzer και εμφανίζεται το μήνυμα

If not(check and printt emp) // ή επανέλεγξε τη θερμοκρασία και εμφάνισε την

Go to arduino server

Loop(2) // επανάληψη (2)

If (button 1 on) // αν το button 1 είναι ενεργοποιημένο

armrelay // ενεργοποίησε το relay 1

Else if (button 2 on) // αλλιώς αν button 2 είναι ενεργοποιημένο

arm relay 2 // όπλισερelay 2

Else if (button 3 on) // αλλιώς αν button 3 είναι ενεργοποιημένο

armrelay 3 // όπλισερelay 3

Else go to Loop(2) // επανάληψη της διαδικασίας

Go to arduino server

Loop(3)

If (light<800lum) // αν ο φωτισμός είναι<800lum

Light flash 100% //άναψε το φως

Else if (800<light<1200) // αλλιώς αν ο φωτισμός είναι μεταξύ 800-1200lum

Light flash 50% // ενεργοποίησε το φως στο 50%

Else if light off // αλλιώς σβήσε το φως

Else go to Loop(3)

Παραπάνω είναι σε μορφή ψευδοκώδικα και σε συντομία η δομή του προγράμματος μας.

Σημαντική παρατήρηση που πρέπει να σημειωθεί είναι ότι ο επεξεργαστής του arduino είναι σχεδιασμένος για πειραματικούς σκοπούς και γι αυτό το λόγω δεν διαθέτει μεγάλη μνήμη. Όπως παρατηρήσατε παραπάνω καλείτε να τρέχει διαρκώς 24 ώρες το 24ωρο κάποιες επαναλαμβανόμενες εκτελέσεις που υπερκαλύπτουν τις δυνατότητες του.

Για να εξοικονομήσουμε χώρο στη μνήμη του συστήματος κάναμε χρήση του webserver και στο πρακτικό κομμάτι αφαιρέθηκε η διαδικασία του αυτομάτου φωτισμού.

Επίσης να σημειώσουμε ότι αυτό συνέβη λόγω του ότι χρησιμοποιήσαμε την πλατφόρμα uno. Στην αγορά διατίθεται πλέον η πλατφόρμα yagi και mega που μπορούν πλέον να υποστηρίξουν πολλαπλούς αισθητήρες ταυτόχρονα και με αρκετά μεγαλύτερη μνήμη ram και rom

21.1Αρχικοποιήσειςπεριφερειακών:

21.1.1Ethernetshield

```
#include<SPI.h>
#include<Ethernet.h>

bytemac[] = {
0xDE, 0xDE, 0xDE, 0xDE, 0xDE, 0xDE};
//ορίζεταιηεπιθυμητήmac address
IPAddressip(192,168,1,25); //ορίζεταιηεπιθυμητήIP

EthernetServer server(80);

void setup() {
Serial.begin(9600);
while (!Serial) {
    ;
  }

Ethernet.begin(mac, ip);
server.begin();
Serial.print("server is at ");
Serial.println(Ethernet.localIP());
}
```

Ο παραπάνω κώδικας παρέχεται δωρεάν από την κατασκευάστρια εταιρεία και διανέμεται είτε με την αγορά του shield, είτε στο επίσημο sitewww.arduino.cc

21.1.2 Μέτρηση θερμοκρασίας μέσω lm35 και εμφάνιση σε serialmonitor

```
int analogPin = 0;
int readvalue = 0;
float temp = 0;


     voidsetup() { // Η συνάρτηση voidsetup εκτελείται
μία φορά κατά την εκκίνηση

     Serial.begin(9600); // Το Serial.begin(9600) ορίζει
ότι θα χρησιμοποιηθεί το SerialMonitor (κυρίως χρειάζεται
για Debugging, με ταχύτητα 9600bps

}

voidloop() { // Η voidloop εκτελείται κατ' επανάληψη
readvalue = analogRead(analogPin); //διαβάζεται η
ένδειξηαπότοanalogPin0
temp = (readvalue * 0.49); //μετατροπή A/D
(5*temp*100)/1024
Serial.print("Temprature of sensor 1: ");
Serial.print(temp); //εμφάνιση στο serial output
Serial.println("C ");
delay(1000);
}
```

21.1.3 Ενεργοποίηση buzzer

```
void buzz(int targetPin, long frequency, long length)
{
long delayValue = 1000000/frequency/2;
long numCycles = frequency * length/ 1000;
for (long i=0; i < numCycles; i++)
{
digitalWrite(targetPin,HIGH);
delayMicroseconds(delayValue);
digitalWrite(targetPin,LOW);
delayMicroseconds(delayValue);
}
```

21.1.4Κώδικας για φωτοαντίσταση

```
intlight_sensor_1 = A1; //αναλογικό pin 1
int led = 13; //ψηφιακό pin 13

void setup(){
Serial.begin(9600); //serial για debug
pinMode(red_led, OUTPUT);
}

void loop(){
int light_read = analogRead(light_sensor_1);
if (light_read<600) {
digitalWrite(led, HIGH); //ανάβει το led
  }
else {
digitalWrite(led,LOW); //σβήνει το  led
}
Serial.println(light_read);
delay(300); //ορίζεται μια μικρή χρονο-καθυστέρηση για να
επιτευχθεί καλύτερο output
}
```

21.1.5Βιβλιοθήκη για WebServer

Είναι απαραίτητη για να στηθεί ο httpserver μας και παρέχεται επίσης δωρεάν από το επίσημο site.

Τα πνευματικά δικαιώματα του συγκεκριμένου κώδικα ανήκουν στους BenCombee, RanTalbott, ChristopherLee, MartinLormes.

```
/* -*- Mode: C++; tab-width: 2; indent-tabs-mode: nil;
c-file-style: "k&r"; c-basic-offset: 2; -*-



   Webduino, a simple Arduino web server

   Copyright 2009-2012 Ben Combee, Ran Talbott,
Christopher Lee, Martin Lormes

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR

   IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY,

   FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE

   AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER

   LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM,

   OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN

   THE SOFTWARE.

*/



#ifndef WEBDUINO_H_

#define WEBDUINO_H_
```

```
#include <string.h>

#include <stdlib.h>


#include <EthernetClient.h>

#include <EthernetServer.h>


/************************************************************
************

 * CONFIGURATION


************************************************************
***********/


#define WEBDUINO_VERSION 1007

#define WEBDUINO_VERSION_STRING "1.7"


#if WEBDUINO_SUPRESS_SERVER_HEADER

#define WEBDUINO_SERVER_HEADER ""

#else

#define WEBDUINO_SERVER_HEADER "Server: Webduino/"
WEBDUINO_VERSION_STRING CRLF

#endif


// standard END-OF-LINE marker in HTTP

#define CRLF "\r\n"


// If processConnection is called without a buffer, it
allocates one
```

```
// of 32 bytes

#define WEBDUINO_DEFAULT_REQUEST_LENGTH 32


// How long to wait before considering a connection as
dead when

// reading the HTTP request.  Used to avoid DOS attacks.

#ifndef WEBDUINO_READ_TIMEOUT_IN_MS

#define WEBDUINO_READ_TIMEOUT_IN_MS 1000

#endif


#ifndef WEBDUINO_FAIL_MESSAGE

#define WEBDUINO_FAIL_MESSAGE "<h1>EPIC FAIL</h1>"

#endif


#ifndef WEBDUINO_AUTH_REALM

#define WEBDUINO_AUTH_REALM "Webduino"

#endif // #ifndef WEBDUINO_AUTH_REALM


#ifndef WEBDUINO_AUTH_MESSAGE

#define WEBDUINO_AUTH_MESSAGE "<h1>401 Unauthorized</h1>"

#endif // #ifndef WEBDUINO_AUTH_MESSAGE


#ifndef WEBDUINO_SERVER_ERROR_MESSAGE

#define WEBDUINO_SERVER_ERROR_MESSAGE "<h1>500 Internal
Server Error</h1>"

#endif // WEBDUINO_SERVER_ERROR_MESSAGE
```

```
// add '#define WEBDUINO_FAVICON_DATA ""' to your
application

// before including WebServer.h to send a null file as
the favicon.ico file

// otherwise this defaults to a 16x16 px black diode on
blue ground

// (or include your own icon if you like)

#ifndef WEBDUINO_FAVICON_DATA

#define WEBDUINO_FAVICON_DATA { 0x00, 0x00, 0x01, 0x00,
0x01, 0x00, 0x10, \

                                0x10, 0x02, 0x00, 0x01,
0x00, 0x01, 0x00, \

                                0xb0, 0x00, 0x00, 0x00,
0x16, 0x00, 0x00, \

                                0x00, 0x28, 0x00, 0x00,
0x00, 0x10, 0x00, \

                                0x00, 0x00, 0x20, 0x00,
0x00, 0x00, 0x01, \

                                0x00, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, \

                                0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, \

                                0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, \

                                0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, \

                                0x00, 0x00, 0x00, 0xff,
0x00, 0x00, 0x00, \

                                0xff, 0xff, 0x00, 0x00,
0xff, 0xff, 0x00, \

                                0x00, 0xff, 0xff, 0x00,
0x00, 0xcf, 0xbf, \
```

```
0x00, 0x00, 0xc3, \
0xbf, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0xc1, 0xbf, \
0x00, 0x00, 0xc7, \
0xbf, 0x00, 0x00, \
0xff, 0xff, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
0x00, 0x00, 0x00, \
```

```
0x00, 0x00, 0xc7, 0xbf,
0xbf, 0x00, 0x00, 0xc1,
0xc0, 0xbf, 0x00, 0x00,
0x00, 0xc0, 0xbf, 0x00,
0x00, 0x00, 0xc3, 0xbf,
0xbf, 0x00, 0x00, 0xcf,
0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00 }
```

```
#endif // #ifndef WEBDUINO_FAVICON_DATA


// add "#define WEBDUINO_SERIAL_DEBUGGING 1" to your
application

// before including WebServer.h to have incoming requests
logged to

// the serial port.

#ifndef WEBDUINO_SERIAL_DEBUGGING

#define WEBDUINO_SERIAL_DEBUGGING 0

#endif

#if WEBDUINO_SERIAL_DEBUGGING

#include <HardwareSerial.h>

#endif


// declared in wiring.h

extern "C" unsigned long millis(void);


// declare a static string

#define P(name)   static const prog_uchar name[] PROGMEM


// returns the number of elements in the array

#define SIZE(array) (sizeof(array) / sizeof(*array))


/************************************************************
************

 * DECLARATIONS
```

```
**********************************************************
***********/


/* Return codes from nextURLparam.  NOTE: URLPARAM_EOS is
returned

 * when you call nextURLparam AFTER the last parameter is
read.  The

 * last actual parameter gets an "OK" return code. */


typedef enum URLPARAM_RESULT { URLPARAM_OK,

                               URLPARAM_NAME_OFLO,

                               URLPARAM_VALUE_OFLO,

                               URLPARAM_BOTH_OFLO,

                               URLPARAM_EOS         // No
params left

};


class WebServer: public Print

{

public:

  // passed to a command to indicate what kind of request
was received

enum ConnectionType { INVALID, GET, HEAD, POST, PUT,
DELETE, PATCH };


  // any commands registered with the web server have to
follow

  // this prototype.
```

```
  // url_tail contains the part of the URL that wasn't
matched against

  //          the registered command table.

  // tail_complete is true if the complete URL fit in
url_tail,  false if

  //          part of it was lost because the buffer was
too small.

typedef void Command(WebServer &server, ConnectionType
type,

char *url_tail, bool tail_complete);



  // constructor for webserver object

WebServer(const char *urlPrefix = "", int port = 80);



  // start listening for connections

void begin();



  // check for an incoming connection, and if it exists,
process it

  // by reading its request and calling the appropriate
command

  // handler.  This version is for compatibility with
apps written for

  // version 1.1,  and allocates the URL "tail" buffer
internally.

void processConnection();



  // check for an incoming connection, and if it exists,
process it
```

```
  // by reading its request and calling the appropriate
command

  // handler.  This version saves the "tail" of the URL
in buff.

void processConnection(char *buff, int *bufflen);



  // set command that's run when you access the root of
the server

void setDefaultCommand(Command *cmd);



  // set command run for undefined pages

void setFailureCommand(Command *cmd);



  // add a new command to be run at the URL specified by
verb

void addCommand(const char *verb, Command *cmd);



  // utility function to output CRLF pair

void printCRLF();



  // output a string stored in program memory, usually
one defined

  // with the P macro

void printP(const prog_uchar *str);



  // inline overload for printP to handle signed char
strings

void printP(const prog_char *str) {
printP((prog_uchar*)str); }
```

```
  // output raw data stored in program memory
void writeP(const prog_uchar *data, size_t length);

  // output HTML for a radio button
void radioButton(const char *name, const char *val,
const char *label, bool selected);

  // output HTML for a checkbox
void checkBox(const char *name, const char *val,
const char *label, bool selected);

  // returns next character or -1 if we're at end-of-
stream
int read();

  // put a character that's been read back into the input
pool
void push(int ch);

  // returns true if the string is next in the stream.
Doesn't
  // consume any character if false, so can be used to
try out
  // different expected values.
bool expect(const char *expectedStr);
```

```
  // returns true if a number, with possible whitespace
in front, was

  // read from the server stream.  number will be set
with the new

  // value or 0 if nothing was read.

bool readInt(int &number);



  // reads a header value, stripped of possible
whitespace in front,

  // from the server stream

void readHeader(char *value, int valueLen);



  // Read the next keyword parameter from the socket.
Assumes that other

  // code has already skipped over the headers,  and the
next thing to

  // be read will be the start of a keyword.

  //

  // returns true if we're not at end-of-stream

bool readPOSTparam(char *name, int nameLen, char *value,
int valueLen);



  // Read the next keyword parameter from the buffer
filled by getRequest.

  //

  // returns 0 if everything weent okay,  non-zero if not

  // (see the typedef for codes)

  URLPARAM_RESULT nextURLparam(char **tail, char *name,
int nameLen,
```

```
char *value, int valueLen);


  // compare string against credentials in current
request

  //

  // authCredentials must be Base64 encoded outside of
Webduino

  // (I wanted to be easy on the resources)

  //

  // returns true if strings match, false otherwise

bool checkCredentials(const char authCredentials[45]);


  // output headers and a message indicating a server
error

void httpFail();


  // output headers and a message indicating "401
Unauthorized"

void httpUnauthorized();


  // output headers and a message indicating "500
Internal Server Error"

void httpServerError();


  // output standard headers indicating "200 Success".
You can change the

  // type of the data you're outputting or also add extra
headers like
```

```
  // "Refresh: 1".  Extra headers should each be
terminated with CRLF.

void httpSuccess(const char *contentType = "text/html;
charset=utf-8",

const char *extraHeaders = NULL);



  // used with POST to output a redirect to another URL.
This is

  // preferable to outputting HTML from a post because
you can then

  // refresh the page without getting a "resubmit form"
dialog.

void httpSeeOther(const char *otherURL);



  // implementation of write used to implement Print
interface

virtual size_t write(uint8_t);

virtual size_t write(const char *str);

virtual size_t write(const uint8_t *buffer, size_t size);

size_t write(const char *data, size_t length);



private:

  EthernetServer m_server;

  EthernetClient m_client;

const char *m_urlPrefix;



unsigned char m_pushback[32];

char m_pushbackDepth;
```

```
int m_contentLength;

char m_authCredentials[51];

bool m_readingContent;


  Command *m_failureCmd;

  Command *m_defaultCmd;

struct CommandMap

  {

const char *verb;

    Command *cmd;

  } m_commands[8];

char m_cmdCount;


void reset();

void getRequest(WebServer::ConnectionType &type, char
*request, int *length);

bool dispatchCommand(ConnectionType requestType, char
*verb,

bool tail_complete);

void processHeaders();

void outputCheckboxOrRadio(const char *element, const
char *name,

const char *val, const char *label,

bool selected);
```

```
static void defaultFailCmd(WebServer &server,
ConnectionType type,

char *url_tail, bool tail_complete);

void noRobots(ConnectionType type);

void favicon(ConnectionType type);

};


/* define this macro if you want to include the header in
a sketch source

file but not define any of the implementation. This is
useful if

multiple source files are using the Webduino class. */

#ifndef WEBDUINO_NO_IMPLEMENTATION


/************************************************************
************

 * IMPLEMENTATION


************************************************************
***********/


WebServer::WebServer(const char *urlPrefix, int port) :

  m_server(port),

  m_client(255),

  m_urlPrefix(urlPrefix),

  m_pushbackDepth(0),

  m_cmdCount(0),

  m_contentLength(0),
```

```
  m_failureCmd(&defaultFailCmd),
  m_defaultCmd(&defaultFailCmd)
{
}

void WebServer::begin()
{
  m_server.begin();
}

void WebServer::setDefaultCommand(Command *cmd)
{
  m_defaultCmd = cmd;
}

void WebServer::setFailureCommand(Command *cmd)
{
  m_failureCmd = cmd;
}

void WebServer::addCommand(const char *verb, Command
*cmd)
{
if (m_cmdCount < SIZE(m_commands))
  {
    m_commands[m_cmdCount].verb = verb;
```

```
    m_commands[m_cmdCount++].cmd = cmd;
  }
}

size_t WebServer::write(uint8_t ch)
{
return m_client.write(ch);
}

size_t WebServer::write(const char *str)
{
return m_client.write(str);
}

size_t WebServer::write(const uint8_t *buffer, size_t
size)
{
return m_client.write(buffer, size);
}

size_t WebServer::write(const char *buffer, size_t
length)
{
return m_client.write((const uint8_t *)buffer, length);
}
```

```
void WebServer::writeP(const prog_uchar *data, size_t
length)

{

  // copy data out of program memory into local storage,
write out in

  // chunks of 32 bytes to avoid extra short TCP/IP
packets

uint8_t buffer[32];

size_t bufferEnd = 0;



while (length--)

  {

if (bufferEnd == 32)

    {

      m_client.write(buffer, 32);

bufferEnd = 0;

    }



buffer[bufferEnd++] = pgm_read_byte(data++);

  }



if (bufferEnd > 0)

    m_client.write(buffer, bufferEnd);

}



void WebServer::printP(const prog_uchar *str)

{
```

```
  // copy data out of program memory into local storage,
write out in

  // chunks of 32 bytes to avoid extra short TCP/IP
packets

uint8_t buffer[32];

size_t bufferEnd = 0;


while (buffer[bufferEnd++] = pgm_read_byte(str++))

  {

if (bufferEnd == 32)

    {

      m_client.write(buffer, 32);

bufferEnd = 0;

    }

  }


  // write out everything left but trailing NUL

if (bufferEnd > 1)

    m_client.write(buffer, bufferEnd - 1);

}


void WebServer::printCRLF()

{

  m_client.write((const uint8_t *)"\r\n", 2);

}
```

```
bool WebServer::dispatchCommand(ConnectionType
requestType, char *verb,

bool tail_complete)

{

  // if there is no URL, i.e. we have a prefix and it's
requested without a

  // trailing slash or if the URL is just the slash

if ((verb[0] == 0) || ((verb[0] == '/') && (verb[1] ==
0)))

  {

    m_defaultCmd(*this, requestType, "", tail_complete);

return true;

  }

  // if the URL is just a slash followed by a question
mark

  // we're looking at the default command with GET
parameters passed

if ((verb[0] == '/') && (verb[1] == '?'))

  {

verb+=2; // skip over the "/?" part of the url

    m_defaultCmd(*this, requestType, verb,
tail_complete);

return true;

  }

  // We now know that the URL contains at least one
character.  And,

  // if the first character is a slash,  there's more
after it.

if (verb[0] == '/')
```

```
  {
char i;
char *qm_loc;
int verb_len;
int qm_offset;
    // Skip over the leading "/",  because it makes the
code more
    // efficient and easier to understand.
verb++;
    // Look for a "?" separating the filename part of the
URL from the
    // parameters.  If it's not there, compare to the
whole URL.
    qm_loc = strchr(verb, '?');
    verb_len = (qm_loc == NULL) ?strlen(verb) : (qm_loc -
verb);
    qm_offset = (qm_loc == NULL) ?0 : 1;
for (i = 0; i < m_cmdCount; ++i)
    {
if ((verb_len == strlen(m_commands[i].verb))
&& (strncmp(verb, m_commands[i].verb, verb_len) == 0))
      {
        // Skip over the "verb" part of the URL (and the
question
        // mark, if present) when passing it to the
"action" routine
        m_commands[i].cmd(*this, requestType,
verb + verb_len + qm_offset,
```

```
        tail_complete);
return true;
      }
    }
  }
return false;
}

// processConnection with a default buffer
void WebServer::processConnection()
{
char request[WEBDUINO_DEFAULT_REQUEST_LENGTH];
int  request_len = WEBDUINO_DEFAULT_REQUEST_LENGTH;
processConnection(request, &request_len);
}

void WebServer::processConnection(char *buff, int
*bufflen)
{
int urlPrefixLen = strlen(m_urlPrefix);

  m_client = m_server.available();

if (m_client) {
    m_readingContent = false;
buff[0] = 0;
```

```
    ConnectionType requestType = INVALID;
#if WEBDUINO_SERIAL_DEBUGGING > 1
Serial.println("*** checking request ***");
#endif
getRequest(requestType, buff, bufflen);
#if WEBDUINO_SERIAL_DEBUGGING > 1
Serial.print("*** requestType = ");
Serial.print((int)requestType);
Serial.print(", request = \"");
Serial.print(buff);
Serial.println("\" ***");
#endif


    // don't even look further at invalid requests.
    // this is done to prevent Webduino from hanging
    // - when there are illegal requests,
    // - when someone contacts it through telnet rather
than proper HTTP,
    // - etc.
if (requestType != INVALID)
    {
processHeaders();
#if WEBDUINO_SERIAL_DEBUGGING > 1
Serial.println("*** headers complete ***");
#endif
```

```
if (strcmp(buff, "/robots.txt") == 0)
      {
noRobots(requestType);
      }
else if (strcmp(buff, "/favicon.ico") == 0)
      {
favicon(requestType);
      }
    }
if      (requestType == INVALID ||
strncmp(buff, m_urlPrefix, urlPrefixLen) != 0 ||
             !dispatchCommand(requestType, buff +
urlPrefixLen,
                              (*bufflen) >= 0))
    {
      m_failureCmd(*this, requestType, buff, (*bufflen)
>= 0);
    }

#if WEBDUINO_SERIAL_DEBUGGING > 1
Serial.println("*** stopping connection ***");
#endif
reset();
  }
}
```

```
bool WebServer::checkCredentials(const char
authCredentials[45])

{

char basic[7] = "Basic ";

if((0 == strncmp(m_authCredentials,basic,6)) &&

     (0 == strcmp(authCredentials, m_authCredentials +
6))) return true;

return false;

}


void WebServer::httpFail()

{

P(failMsg) =

    "HTTP/1.0 400 Bad Request" CRLF

    WEBDUINO_SERVER_HEADER

    "Content-Type: text/html" CRLF

    CRLF

    WEBDUINO_FAIL_MESSAGE;


printP(failMsg);

}


void WebServer::defaultFailCmd(WebServer &server,

                               WebServer::ConnectionType
type,

char *url_tail,

bool tail_complete)
```

```
{
server.httpFail();
}

void WebServer::noRobots(ConnectionType type)
{
httpSuccess("text/plain");
if (type != HEAD)
  {
P(allowNoneMsg) = "User-agent: *" CRLF "Disallow: /"
CRLF;
printP(allowNoneMsg);
  }
}

void WebServer::favicon(ConnectionType type)
{
httpSuccess("image/x-icon","Cache-Control: max-
age=31536000\r\n");
if (type != HEAD)
  {
P(faviconIco) = WEBDUINO_FAVICON_DATA;
writeP(faviconIco, sizeof(faviconIco));
  }
}
```

```
void WebServer::httpUnauthorized()

{

P(failMsg) =

    "HTTP/1.0 401 Authorization Required" CRLF

    WEBDUINO_SERVER_HEADER

    "Content-Type: text/html" CRLF

    "WWW-Authenticate: Basic realm=\""
WEBDUINO_AUTH_REALM "\"" CRLF

    CRLF

    WEBDUINO_AUTH_MESSAGE;


printP(failMsg);

}


void WebServer::httpServerError()

{

P(failMsg) =

    "HTTP/1.0 500 Internal Server Error" CRLF

    WEBDUINO_SERVER_HEADER

    "Content-Type: text/html" CRLF

    CRLF

    WEBDUINO_SERVER_ERROR_MESSAGE;


printP(failMsg);

}
```

```
void WebServer::httpSuccess(const char *contentType,
const char *extraHeaders)
{
P(successMsg1) =
    "HTTP/1.0 200 OK" CRLF
    WEBDUINO_SERVER_HEADER
    "Access-Control-Allow-Origin: *" CRLF
    "Content-Type: ";

printP(successMsg1);
print(contentType);
printCRLF();
if (extraHeaders)
print(extraHeaders);
printCRLF();
}

void WebServer::httpSeeOther(const char *otherURL)
{
P(seeOtherMsg) =
    "HTTP/1.0 303 See Other" CRLF
    WEBDUINO_SERVER_HEADER
    "Location: ";

printP(seeOtherMsg);
```

```
print(otherURL);

printCRLF();

printCRLF();

}


int WebServer::read()

{

if (m_client == NULL)

return -1;


if (m_pushbackDepth == 0)

  {

unsigned long timeoutTime = millis() +
WEBDUINO_READ_TIMEOUT_IN_MS;


while (m_client.connected())

    {

      // stop reading the socket early if we get to
content-length

      // characters in the POST.  This is because some
clients leave

      // the socket open because they assume HTTP keep-
alive.

if (m_readingContent)

      {

if (m_contentLength == 0)

        {
```

```
#if WEBDUINO_SERIAL_DEBUGGING > 1

Serial.println("\n*** End of content, terminating
connection");

#endif

return -1;

        }

        --m_contentLength;

      }



int ch = m_client.read();



      // if we get a character, return it, otherwise
continue in while

      // loop, checking connection status

if (ch != -1)

      {

#if WEBDUINO_SERIAL_DEBUGGING

if (ch == '\r')

Serial.print("<CR>");

else if (ch == '\n')

Serial.println("<LF>");

else

Serial.print((char)ch);

#endif

return ch;

      }
```

```
else
      {
unsigned long now = millis();
if (now > timeoutTime)
        {
          // connection timed out, destroy client, return
EOF
#if WEBDUINO_SERIAL_DEBUGGING
Serial.println("*** Connection timed out");
#endif
reset();
return -1;
        }
      }
    }

    // connection lost, return EOF
#if WEBDUINO_SERIAL_DEBUGGING
Serial.println("*** Connection lost");
#endif
return -1;
  }
else
return m_pushback[--m_pushbackDepth];
}
```

```
void WebServer::push(int ch)

{

  // don't allow pushing EOF

if (ch == -1)

return;


  m_pushback[m_pushbackDepth++] = ch;

  // can't raise error here, so just replace last char
over and over

if (m_pushbackDepth == SIZE(m_pushback))

    m_pushbackDepth = SIZE(m_pushback) - 1;

}


void WebServer::reset()

{

  m_pushbackDepth = 0;

  m_client.flush();

  m_client.stop();

}


bool WebServer::expect(const char *str)

{

const char *curr = str;

while (*curr != 0)

  {

int ch = read();
```

```
if (ch != *curr++)
    {
      // push back ch and the characters we accepted
push(ch);
while (--curr != str)
push(curr[-1]);
return false;
    }
  }
return true;
}

bool WebServer::readInt(int &number)
{
bool negate = false;
bool gotNumber = false;
int ch;
number = 0;

  // absorb whitespace
do
  {
ch = read();
  } while (ch == ' ' || ch == '\t');
```

```
  // check for leading minus sign
if (ch == '-')
  {
negate = true;
ch = read();
  }

  // read digits to update number, exit when we find non-
digit
while (ch >= '0' && ch <= '9')
  {
gotNumber = true;
number = number * 10 + ch - '0';
ch = read();
  }

push(ch);
if (negate)
number = -number;
return gotNumber;
}

void WebServer::readHeader(char *value, int valueLen)
{
int ch;
memset(value, 0, valueLen);
```

```
  --valueLen;

  // absorb whitespace
do
  {
ch = read();
  } while (ch == ' ' || ch == '\t');

  // read rest of line
do
  {
if (valueLen > 1)
    {
      *value++=ch;
      --valueLen;
ch = read();
    }
  } while (ch != '\r');
push(ch);
}

bool WebServer::readPOSTparam(char *name, int nameLen,
char *value, int valueLen)
{
  // assume name is at current place in stream
```

```
int ch;


  // clear out name and value so they'll be NUL
terminated

memset(name, 0, nameLen);

memset(value, 0, valueLen);


  // decrement length so we don't write into NUL
terminator

  --nameLen;

  --valueLen;


while ((ch = read()) != -1)

  {

if (ch == '+')

    {

ch = ' ';

    }

else if (ch == '=')

    {

      /* that's end of name, so switch to storing in
value */

nameLen = 0;

continue;

    }

else if (ch == '&')

    {
```

```
      /* that's end of pair, go away */
return true;
    }
else if (ch == '%')
    {
      /* handle URL encoded characters by converting back
to original form */
int ch1 = read();
int ch2 = read();
if (ch1 == -1 || ch2 == -1)
return false;
char hex[3] = { ch1, ch2, 0 };
ch = strtoul(hex, NULL, 16);
    }


    // output the new character into the appropriate
buffer or drop it if
    // there's no room in either one.  This code will
malfunction in the
    // case where the parameter name is too long to fit
into the name buffer,
    // but in that case, it will just overflow into the
value buffer so
    // there's no harm.
if (nameLen > 0)
    {
      *name++ = ch;
      --nameLen;
```

```
    }
else if (valueLen > 0)
    {
      *value++ = ch;
      --valueLen;
    }
  }

  // if we get here, we hit the end-of-file, so POST is
over and there
  // are no more parameters
return false;
}

/* Retrieve a parameter that was encoded as part of the
URL, stored in
 * the buffer pointed to by *tail.  tail is updated to
point just past
 * the last character read from the buffer. */
URLPARAM_RESULT WebServer::nextURLparam(char **tail, char
*name, int nameLen,
char *value, int valueLen)
{
  // assume name is at current place in stream
char ch, hex[3];
  URLPARAM_RESULT result = URLPARAM_OK;
char *s = *tail;
```

```
bool keep_scanning = true;

bool need_value = true;


  // clear out name and value so they'll be NUL
terminated

memset(name, 0, nameLen);

memset(value, 0, valueLen);


if (*s == 0)

return URLPARAM_EOS;

  // Read the keyword name

while (keep_scanning)

  {

ch = *s++;

switch (ch)

    {

case 0:

s--;  // Back up to point to terminating NUL

      // Fall through to "stop the scan" code

case '&':

      /* that's end of pair, go away */

      keep_scanning = false;

      need_value = false;

break;

case '+':

ch = ' ';
```

```
break;
case '%':
      /* handle URL encoded characters by converting back
       * to original form */
if ((hex[0] = *s++) == 0)
      {
s--;        // Back up to NUL
        keep_scanning = false;
        need_value = false;
      }
else
      {
if ((hex[1] = *s++) == 0)
        {
s--;  // Back up to NUL
          keep_scanning = false;
          need_value = false;
        }
else
        {
hex[2] = 0;
ch = strtoul(hex, NULL, 16);
        }
      }
break;
```

```
case '=':

      /* that's end of name, so switch to storing in
value */

      keep_scanning = false;

break;

    }



    // check against 1 so we don't overwrite the final
NUL

if (keep_scanning && (nameLen > 1))

    {

      *name++ = ch;

      --nameLen;

    }

else

result = URLPARAM_NAME_OFLO;

  }


if (need_value && (*s != 0))

  {

    keep_scanning = true;

while (keep_scanning)

    {

ch = *s++;

switch (ch)
```

```
      {
case 0:
s--;  // Back up to point to terminating NUL
              // Fall through to "stop the scan" code
case '&':
        /* that's end of pair, go away */
        keep_scanning = false;
        need_value = false;
break;
case '+':
ch = ' ';
break;
case '%':
        /* handle URL encoded characters by converting
back to original form */
if ((hex[0] = *s++) == 0)
        {
s--;  // Back up to NUL
          keep_scanning = false;
          need_value = false;
        }
else
        {
if ((hex[1] = *s++) == 0)
          {
s--;  // Back up to NUL
```

```
            keep_scanning = false;
            need_value = false;
          }
else
          {
hex[2] = 0;
ch = strtoul(hex, NULL, 16);
          }

        }
break;
      }


      // check against 1 so we don't overwrite the final
NUL
if (keep_scanning && (valueLen > 1))
      {
        *value++ = ch;
        --valueLen;
      }
else
result = (result == URLPARAM_OK) ?
          URLPARAM_VALUE_OFLO :
          URLPARAM_BOTH_OFLO;
    }
```

```
  }

  *tail = s;

return result;

}
```

```
// Read and parse the first line of the request header.

// The "command" (GET/HEAD/POST) is translated into a
numeric value in type.

// The URL is stored in request,  up to the length passed
in length

// NOTE 1: length must include one byte for the
terminating NUL.

// NOTE 2: request is NOT checked for NULL,  nor length
for a value < 1.

// Reading stops when the code encounters a space, CR, or
LF.  If the HTTP

// version was supplied by the client,  it will still be
waiting in the input

// stream when we exit.

//

// On return, length contains the amount of space left in
request.  If it's

// less than 0,  the URL was longer than the buffer,  and
part of it had to

// be discarded.


void WebServer::getRequest(WebServer::ConnectionType
&type,
```

```
char *request, int *length)

{

  --*length; // save room for NUL


type = INVALID;


  // store the HTTP method line of the request

if (expect("GET "))

type = GET;

else if (expect("HEAD "))

type = HEAD;

else if (expect("POST "))

type = POST;

else if (expect("PUT "))

type = PUT;

else if (expect("DELETE "))

type = DELETE;

else if (expect("PATCH "))

type = PATCH;


  // if it doesn't start with any of those, we have an
unknown method

  // so just get out of here

else

return;
```

```
int ch;
while ((ch = read()) != -1)
  {
    // stop storing at first space or end of line
if (ch == ' ' || ch == '\n' || ch == '\r')
    {
break;
    }
if (*length > 0)
    {
      *request = ch;
      ++request;
    }
    --*length;
  }
  // NUL terminate
  *request = 0;
}

void WebServer::processHeaders()
{
  // look for three things: the Content-Length header,
the Authorization
  // header, and the double-CRLF that ends the headers.
```

```
  // empty the m_authCredentials before every run of this
function.

  // otherwise users who don't send an Authorization
header would be treated

  // like the last user who tried to authenticate
(possibly successful)

  m_authCredentials[0]=0;



while (1)

  {

if (expect("Content-Length:"))

    {

readInt(m_contentLength);

#if WEBDUINO_SERIAL_DEBUGGING > 1

Serial.print("\n*** got Content-Length of ");

Serial.print(m_contentLength);

Serial.print(" ***");

#endif

continue;

    }



if (expect("Authorization:"))

    {

readHeader(m_authCredentials,51);

#if WEBDUINO_SERIAL_DEBUGGING > 1

Serial.print("\n*** got Authorization: of ");

Serial.print(m_authCredentials);
```

```
Serial.print(" ***");

#endif

continue;

    }


if (expect(CRLF CRLF))

    {

      m_readingContent = true;

return;

    }


    // no expect checks hit, so just absorb a character
and try again

if (read() == -1)

    {

return;

    }

  }

}


void WebServer::outputCheckboxOrRadio(const char
*element, const char *name,

const char *val, const char *label,

bool selected)

{

P(cbPart1a) = "<label><input type='";
```

```
P(cbPart1b) = "' name='";

P(cbPart2) = "' value='";

P(cbPart3) = "' ";

P(cbChecked) = "checked ";

P(cbPart4) = "/> ";

P(cbPart5) = "</label>";


printP(cbPart1a);

print(element);

printP(cbPart1b);

print(name);

printP(cbPart2);

print(val);

printP(cbPart3);

if (selected)

printP(cbChecked);

printP(cbPart4);

print(label);

printP(cbPart5);

}


void WebServer::checkBox(const char *name, const char
*val,

const char *label, bool selected)

{
```

```
outputCheckboxOrRadio("checkbox", name, val, label,
selected);

}


void WebServer::radioButton(const char *name, const char
*val,

const char *label, bool selected)

{

outputCheckboxOrRadio("radio", name, val, label,
selected);

}


#endif // WEBDUINO_NO_IMPLEMENTATION


#endif // WEBDUINO_H_
```

21.2Πρότυπος κώδικας για κατασκευή WebServer

Ο κάτωθι κώδικας αποτέλεσε τη βάση ώστε να δημιουργήσουμε το δικό μας WebServer και επίσης παρέχεται δωρεάν από το επίσημο site.

```
/*
  Web Server

 A simple web server that shows the value of the analog
input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5
(optional)

 created 18 Dec 2009
 by David A. Mellis
 modified 9 Apr 2012
 by Tom Igoe

 */

#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address and IP address for your controller
below.
// The IP address will be dependent on your local
network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for
Leonardo only
  }
```

```
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line
(received a newline
        // character) and the line is blank, the http
request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the
connection will be closed after completion of the
response
      client.println("Refresh: 5");  // refresh the page
automatically every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin

  for (int analogChannel = 0; analogChannel < 6; analogCh
annel++) {

  int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
```

```
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current
line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```

21.2.1Web Server

```
#include "SPI.h"

#include "Ethernet.h"

#include "WebServer.h"


static uint8_t mac[6] = { 0xAA, 0xAA, 0xAA, 0xAA, 0xAA,
0xAA };


static uint8_t ip[4] = { 192, 168, 1, 220 }; // Η ip
πουδίνουμεστο arduino


#define PREFIX "/remote"

WebServer webserver(PREFIX, 80);


#define RED_PIN 5

#define GREEN_PIN 3

#define BLUE_PIN 6


int red = 0;  //pin 5,καλώδιοπουσυνδέει arduino με buzzer

int blue = 0;   //pin 6,καλώδιοπουσυνδέει arduino με led

int green = 0;  //pin 3, καλώδιοπουσυνδέει arduino με
relay

void rgbCmd(WebServer &server, WebServer::ConnectionType
type, char *, bool)

{

if (type == WebServer::POST)
```

```
  {
bool repeat;
char name[16], value[16];
do
    {
repeat = server.readPOSTparam(name, 16, value, 16);

if (strcmp(name, "red") == 0)
      {
red = strtoul(value, NULL, 10);
      }
if (strcmp(name, "green") == 0)
      {
green = strtoul(value, NULL, 10);
      }
if (strcmp(name, "blue") == 0)
      {
blue = strtoul(value, NULL, 10);
      }
    } while (repeat);

server.httpSeeOther(PREFIX);

return;
  }
```

```
server.httpSuccess();


if (type == WebServer::GET)

  {

P(message) =

  "<!DOCTYPE html><html><head>"

  "<meta http-equiv=\"Content-Type\" content=\"text/html;
charset=utf-8\">"

  "<title>Smart Home</title>"

  "<link rel=\"stylesheet\"
href=\"http://code.jquery.com/mobile/1.3.1/jquery.mobile-
1.3.1.min.css\" />"

  "<script src=\"http://code.jquery.com/jquery-
1.9.1.min.js\"></script>"

  "<script
src=\"http://code.jquery.com/mobile/1.3.1/jquery.mobile-
1.3.1.min.js\"></script>"

  "<script>"



    "$(document).ready(function(){ $('#red, #green,
#blue').slider; $('#red, #green, #blue').bind( 'change',
function(event, ui) { jQuery.ajaxSetup({timeout: 1000});
/*not to DDoS the Arduino, you might have to change this
to some threshold value that fits your setup*/ var id =
$(this).attr('id'); var strength = $(this).val(); if (id
== 'red') $.post('/remote', { red: strength } ); if (id
== 'green') $.post('/remote', { green: strength } ); if
(id == 'blue') $.post('/remote', { blue: strength } );
});});"

  "</script>"

"</head>"

"<body>"
```

```
  "<div data-role=\"header\" data-
position=\"inline\"><h1>Smart Home by Crhistopher-Bill
Koutsikos</h1></div>"

    "<div class=\"ui-body ui-body-a\">"

      "<div data-role=\"fieldcontain\">"

        "<h2>Φώς</h2>"

     "<label for=\"flip-3\">Διακόπτης 1</label>"

     "<select name=\"flip-3\" id=\"blue\" data-
role=\"slider\" data-theme=\"b\">"

          "<option value=\"0\">OFF</option>"

          "<option value=\"255\">ON</option>"

     "</select> "

        "<h2>Γκαράζ</h2>"

        "<label for=\"flip-3\">Διακόπτης 2</label>"

     "<select name=\"flip-3\" id=\"red\" data-
role=\"slider\" data-theme=\"b\">"

          "<option value=\"0\">OFF</option>"

          "<option value=\"255\">ON</option>"

     "</select> "

        "<h2>Θερμοσίφωνο:</h2>"

        "<label for=\"flip-3\">Διακόπτης 3</label>"

     "<select name=\"flip-3\" id=\"green\" data-
role=\"slider\" data-theme=\"b\">"

          "<option value=\"0\">OFF</option>"

          "<option value=\"255\">ON</option>"

     "</select> "

      "</div>"
```

```
     "<center><img src=\"http://images.intomobile.com/wp-
content/uploads/2011/05/Android-Home.jpg\"></center>"

    "</div>"

  "</body>"

"</html>";


server.printP(message);

  }

}


void setup()

{

pinMode(RED_PIN, OUTPUT);

pinMode(GREEN_PIN, OUTPUT);

pinMode(BLUE_PIN, OUTPUT);


Ethernet.begin(mac, ip);


webserver.setDefaultCommand(&rgbCmd);


webserver.begin();

}


void loop()

{

webserver.processConnection();
```

```
analogWrite(RED_PIN, red);
analogWrite(GREEN_PIN, green);
analogWrite(BLUE_PIN, blue);
}
```

22 Βιβλιογραφία

- ✓ **www.arduino.cc**
- ✓ **www.wikipedia.org**
- ✓ **www.arduino.gr.com**
- ✓ **www.deltahacker.gr**
- ✓ **www.sparkfun.com**
- ✓ **www.smarthome.com**
- ✓ **www.smart-home.gr**
- ✓ **www.sma.de**
- ✓ **www.all-experts.gr**
- ✓ **www.cprogramming.com**
- ✓ **www.physics.drexel.edu**
- ✓ **www.cs.ucsb.edu**
- ✓ **www.learn-c.org**
- ✓ **www.cforbeginners.com**
- ✓ **www.wiring.org.co**
- ✓ **www.wiringthebrain.com**
- ✓ **www.android.com**
- ✓ **www.doctorandroid.gr**
- ✓ **www.androidcommunity.com**
- ✓ **www.developer.android.com**
- ✓ **www.youtube.com**
- ✓ **www.creativebloq.com**
- ✓ **www.tutorialspoint.com**
- ✓ **www.httpd.apache.org**
- ✓ **www.npmjs.org**
- ✓ **www.rejetto.com**
- ✓ **www.ibm.com**
- ✓ **www.docs.python.org**
- ✓ **www.quackit.com**
- ✓ **www.serverwatch.com**
- ✓ **www.yolinux.com**
- ✓ **www.ee.surrey.ac.uk**
- ✓ **www.tldp.org**
- ✓ **www.linux-tutorial.info**
- ✓ **www.linux.org**
- ✓ **www.linuxsurvival.com**
- ✓ **www.tutorial.webcrawler.com**

20 Πηγές εικόνων

Εικόνα 1: http://www.epaggelmaties.com/writer/2001-2003/teyxos212.html

Εικόνα 2: https://www.arduino.cc/en/uploads/Main/ArduinoDiecimilaComponents.jpg

Εικόνα3α: http://www.grobot.gr/images/stories/2010/ArduinoWithEthernetShield.jpg

Εικόνα3β: http://www.grobot.gr/images/stories/2010/arduino-pro-and-pro-mini.jpg

Εικόνα 4: http://www.robotpark.com.tr/image/cache/data/PRO/91077/91077-Arduino-GSM-Shield-Pic01-700x700.png

Εικόνα 5: https://www.arduino.cc/en/Main/ArduinoEthernetShield

Εικόνα 6: https://www.arduino.cc/en/Main/ArduinoWiFiShield

Εικόνα 7: https://www.arduino.cc/en/Main/ArduinoWirelessShield

Εικόνα 8: https://www.arduino.cc/en/Main/ArduinoUSBHostShield

Εικόνα 9: https://www.arduino.cc/en/Main/ArduinoMotorShieldR3

Εικόνα 10: https://jkdevices.com/image/cache/data/JKD-ULTRASONIC-01a-800x600.jpg

Εικόνα 11: http://learn.parallax.com/sites/default/files/content/shield/CH4-gas-sensor-demo/605-00008a-250H.png

Εικόνα 12:http://www.mhobbies.com/media/catalog/product/cache/1/image/500x500/1ed5d654e5bcf35b6a089ddd1e89d647/6/4/6490-arduino-dht11-humidity-temperature-sensor-module_1_.jpg

Εικόνα 13: http://www.mhobbies.com/media/wysiwyg/Arduino_photoresistance_light_sensor_module_light_sensor_Use_for_light_detection.jpg

Εικόνα 14: http://img.dxcdn.com/productimages/sku_150778_1.jpg

Εικόνα 15: http://www.grobot.gr/images/stories/2010/Arduino_IDE_-_v0011_Alpha.png

Εικόνα 16: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 17: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 18: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 19: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 20: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 21: Φωτογράφηση Χ. Β. Κούτσικος

Εικόνα 22: Φωτογράφηση Χ. Β. Κούτσικος