ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΜΕΛΕΤΗ, ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΟΙΚΙΑΚΟΥ ΑΥΤΟΜΑΤΙΣΜΟΥ ΜΕ ΧΡΗΣΗ ΜΙΚΡΟΕΛΕΓΚΤΗ ARDUINO ΣΕ ΤΟΠΙΚΟ ΚΑΙ ΑΠΟΜΑΚΡΥΣΜΕΝΟ ΠΕΡΙΒΑΛΛΟΝ

ΣΠΟΥΔΑΣΤΗΣ: ΛΑΓΟΔΗΜΟΣ ΙΩΑΝΝΗΣ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: ΤΟΠΑΛΗΣ ΕΥΑΓΓΕΛΟΣ

ΠΑΤΡΑ 2015

Ευχαριστίες

Πρώτα απ' όλα, θέλω να ευχαριστήσω τον επιβλέποντα της πτυχιακής εργασίας μου, Καθηγητή κ. Ευάγγελο Τοπάλη, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της δουλειάς μου. Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής της πτυχιακής εργασίας μου, Καθηγητές κ. Λουκά Χαδέλλη και κ. Βασίλειο Καψάλη για την προσεκτική ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους. Οφείλω ευχαριστίες στον Καθηγητή κ. Καλαντζοπουλο Αθανάσιο για τη βοήθειά του στον προγραμματισμό του λογισμικού LabVIEW. Ευχαριστώ την αδερφή μου Χρυσάνθη για την πολύτιμη βοήθεια και τις συμβουλές της στον προγραμματισμό του διαδικτύου και του Arduino και για την ψυχολογική υποστήριξή της στην πορεία της εργασίας μου. Ευχαριστώ το συνάδελφο μου Αρντένη Φεϊζάι για την πολύτιμη βοήθειά του στον προγραμματισμό με LabVIEW και Arduino. Ευχαριστώ τους φίλους(ες) μου Χρήστο Σώκκα, Θεοδώρα Λαμπροπούλου, Έλενα Διαμαντοπούλου, Δήμητρα Γιώτα και Κώστα Μουλά για την ηθική υποστήριξή τους. Επίσης, ευχαριστώ την Βιβιάννα Κοσκινά για την κατανόηση, την υπομονή και την ψυχολογική υποστήριξή της, ιδιαίτερα κατά τη διάρκεια των τελευταίων μηνών της προσπάθειάς μου. Πάνω απ' όλα, είμαι ευγνώμων στους γονείς μου, Γεώργιο και Σοφία για την ολόψυχη αγάπη και υποστήριξή τους όλα αυτά τα χρόνια. Αφιερώνω αυτή την εργασία στην μητέρα μου και στον πατέρα μου.

Γιάννης Λαγοδήμος

Περίληψη

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η υλοποίηση ενός συστήματος οικιακού αυτοματισμού με τοπικό και απομακρυσμένο έλεγχο συσκευών χρησιμοποιώντας μικροελεγκτή Arduino Mega, με τη βοήθεια αισθητήρων, lcd οθόνης, φωνητικών εντολών μέσω Bluetooth, ιστοσελίδας και της πλατφόρμας σχεδιασμού συστημάτων σε περιβάλλον ανάπτυξης οπτικού προγραμματισμού LabVIEW. Αναλυτικότερα πραγματεύεται το πώς με ένα σχετικά χαμηλό κόστος μια κατοικία μπορεί να αποκτήσει νοημοσύνη και να γίνει λειτουργική με τη βοήθεια αυτοματισμών τους οποίους μπορούμε να επαναπρογραμματίζουμε ανάλογα με τις ανάγκες μας, αποκτώντας τη δυνατότητα να ελέγχει οικιακές συσκευές, το φωτισμό και την κατανάλωση ενέργειας, μέσω ενός Smartphone ή ενός υπολογιστή, χωρίς αυτό να απαιτεί προϋπάρχουσα υποδομή.

Καταρχήν θα μελετηθεί και θα κατανοηθεί η έννοια του έξυπνου σπιτιού όσον αφορά τη χρησιμότητα και την εφαρμογή του στην καθημερινότητα. Στη συνέχεια θα γίνει εκτεταμένη αναφορά στους μικροελεγκτές Arduino, στα προγράμματα που χρησιμοποιήθηκαν όπως το LabVIEW, το Xampp, στις βάσεις δεδομένων mySQL, στις γλώσσες και τα εργαλεία προγραμματισμού διαδικτύου HTML, PHP, JavaScript, CSS. Επιπροσθέτως θα αναλυθεί ο τρόπος παρουσίασης γραφικών παραστάσεων που πραγματοποιήθηκαν με τη βοήθεια της βιβλιοθήκης amCharts.

Στο βασικό μέρος της πτυχιακής εργασίας θα επεξηγηθεί διεξοδικά το σύστημα που υλοποιήθηκε. Συγκεκριμένα θα γίνει ανάλυση όσον άφορα τους χρησιμοποιούμενους αισθητήρες, τους αυτοματισμούς και τις λογικές που εφαρμόστηκαν αφενός στα πλαίσια του προγραμματισμού και αφετέρου στο κομμάτι των τεχνικών δικτύωσης και των απαιτούμενων συνδέσεων.

Τέλος , θα γίνει παράθεση συμπερασμάτων, όσον αφορά το τελικό αποτέλεσμα και θα αναφερθούμε σε ιδέες και σκέψεις για μελλοντικές επεκτάσεις και βελτιώσεις του συστήματος που υλοποιήθηκε.

Πίνακας περιεχομένων

Περίληψη iv
Κεφάλαιο 1: Εισαγωγή 9
1.1 Γενικά 9
1.2 Έξυπνο σπίτι 9
1.2.1 Ορίζοντας το "Έξυπνο σπίτι" 9
1.2.2Δυνατότητες έξυπνου σπιτιού 10
1.2.3 Πλεονεκτήματα και χαρακτηριστικά του έξυπνου σπιτιού 10
1.3 Ιδέα - Σύστημα προς υλοποίηση 11
ΚΕΦΑΛΑΙΟ 2: Τεχνολογίες, Πλατφόρμες και Εργαλεία Ανάπτυξης που χρησιμοποιήθηκαν 13
2.1 Τι είναι η HTML 13
2.1.1 Σήμανση HTML 13
2.1.2 Παραδείγματα στοιχείων HTML 14
2.1.3 Ιδιότητες HTML 14
2.1.4 Τύποι δεδομένων HTML 16
2.1.5 Δήλωση τύπου εγγράφου 16
2.2 CSS 16
2.3 JavaScript 17
2.4 PHP 17
2.4.1 Πώς λειτουργεί η PHP 18
2.5 Τι είναι η MySQL; 19
2.6 Τεχνολογίες επικοινωνίας που χρησιμοποιήθηκαν 20
2.6.1 Bluetooth 20
2.6.2 IEEE 802.11(Wi-Fi) 21
2.7 Μικροελεγκτές 21
2.7.1 Arduino 22
2.7.2 Εκδόσεις Arduino 23
2.8 Πακέτο προγραμμάτων XAMPP 25
2.8.1 XAMPP 25
2.8.2 Apache HTTP εξυπηρετητής 26
2.8.3 Δυνατότητες και απαιτήσεις XAMPP 26
2.8.4 Χρήση XAMPP 26
2.9 Γραφήματα με JavaScript AmCharts 27

2.10Fritzing 27
2.11 Λογισμικό LabVIEW.....28
2.11.1 National Instruments VISA 28
2.11.2 Προγραμματισμός σε LabVIEW.....28
2.11.3 Έλεγχος LED με LabVIEW 32
2.11.4 Σύνδεση LabVIEW με Arduino 33
2.11.5 Απαιτήσεις Arduino για σύνδεση και έλεγχο από LabVIEW 33
2.12 Λογισμικό προγραμματισμού ARDUINO IDE 1.6.4 35
2.12.1 Ελάχιστα συστατικά του προγράμματος και αρχικές ρυθμίσεις.....35
2.12.2 Προγραμματισμός εξόδου για έλεγχο LED 36
2.12.3 Βήματα υλοποίησης ελέγχου LED 37
2.12.5 Αναφορά στη γλώσσα του Arduino.....41
2.13 Διαδικτυακή πλατφόρμα προγραμματισμού codebender.cc.....45
ΚΕΦΑΛΑΙΟ 3: Υλικό Μέρος Συστήματος.....48
3.1 Arduino MEGA 2560.....48
3.1.1Προγραμματισμός Arduino Mega 49
3.1.2 Προσοχή/Προειδοποιήσεις για Arduino/Genuino 2560 Mega 49
3.1.3 Τροφοδοσία Arduino/Genuino 2560 Mega 49
3.1.4 Τα Pin τροφοδοσίας του Arduino/Genuino 2560 Mega.....49
3.1.5 Μνήμη Arduino/Genuino 2560 Mega.....50
3.1.6 Είσοδοι και Έξοδοι Arduino/Genuino 2560 Mega.....50
3.2 Επικοινωνία Arduino με το διαδίκτυο (Ethernet Shield).....52
3.3 Αισθητήρες που χρησιμοποιήθηκαν 54
3.3.1 Microbot DHT11: Ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας.....54
3.3.2 PIR Ανιχνευτής Κίνησης.....57
3.3.2.1 Πώς λειτουργεί ένας PIR; 58
3.3.3 Photo Resistor LDR 63
3.3.4 Non-Invasive Current Sensor - 30A.....66
3.3.5 Ανιχνευτής βροχής.....70
3.3.6 Bluetooth HC-06.....71
3.3.7 Οθόνη LCD IIC/TWI 20x4.....73
3.4 Κύκλωμα 74
ΚΕΦΑΛΑΙΟ 4: Λογισμικό Μέρος Συστήματος.....77
4.1 Προγραμματισμός Arduino 77
4.1.1 Καταχώρηση βιβλιοθηκών.....77
4.1.2 Δηλώσεις Ethernet, αρχικοποιήσεις μεταβλητών και ονοματοδοσία Pin ...77

4.1.3 Προσδιορισμός I/O 78
4.1.4 Μετρήσεις αισθητήρων 79
4.1.5 Σενάρια αυτοματισμών 79
4.1.6 Έλεγχος από οθόνη LCD 81
4.1.7 Έλεγχος σειριακών θηρών 82
4.1.8 Έλεγχος με φωνητικές εντολές 83
4.1.9 Ιστοσελίδα στην κάρτα SD του Arduino 84
4.1.10 Αποστολή μετρήσεων στη βάση δεδομένων 85
4.1.11 Δημιουργία XML Ιστοσελίδας 86
4.2 Ιστοσελίδα που υλοποιήθηκε και λειτουργίες της 88
4.2.1 Πως λειτουργεί ο server της ιστοσελίδας 91
4.3 Βάση δεδομένων 95
4.4 Γραφήματα 104
4.5 Προγραμματισμός LabVIEW 109
Κεφάλαιο 5: Συμπεράσματα και Μελλοντικές Επεκτάσεις 117
Βιβλιογραφία 119
Παραρτημα 120
Κώδικες 120

Κεφάλαιο 1: Εισαγωγή

1.1 Γενικά

Στις μέρες μας τα πάντα τείνουν να αυτοματοποιηθούν, να αποκτήσουν νοημοσύνη και να γίνουν έξυπνα. Περιτριγυριζόμαστε από έξυπνες οικιακές συσκευές, έξυπνα αυτοκίνητα, έξυπνα κινητά τηλέφωνα. Δε θα μπορούσε λοιπόν να λείψει από αυτό το πλαίσιο που κατακλύζει την καθημερινότητά μας και το ίδιο μας το σπίτι.

Κάποτε για να μπορέσουμε να έχουμε φως τη νύχτα έπρεπε να χτυπάμε πέτρες και με τη σπίθα να ανάψουμε φωτιά σε ξερά χόρτα και κλαδιά. Αργότερα, κάποιος έπρεπε να ρυθμίσει μια λάμπα πετρελαίου. Στη συνέχεια τα πράγματα έγιναν πιο εύκολα, καθώς το πάτημα και μόνο ενός διακόπτη αρκούσε για να ανάψει ένα ηλεκτρικό λαμπτήρα. Σήμερα όμως έχουμε φτάσει σε ένα σημείο που ακόμα και το πάτημα αυτού του διακόπτη δεν χρειάζεται να το κάνουμε εμείς. Πολύ απλά ένας υπολογιστής μπορεί να αναλαμβάνει αυτή τη δουλεία για εμάς. Κάποιος μπορεί αυτό μπορεί άνετα να το θεωρήσει οκνηρία. Η επιστήμη και η τεχνολογία όμως το αποκαλούν πρόοδο και εξέλιξη.

Από το αυτόματο πότισμα του κήπου και το αυτόματο κλείσιμο των παραθύρων σε περίπτωση βροχής μέχρι την εναλλαγή διάφορων αυτοματοποιημένων σεναρίων ανάλογα με το αν για παράδειγμα ο ιδιοκτήτης βρίσκεται ή όχι στο χώρο του σπιτιού, αυτό που ονομάζουμε έξυπνο σπίτι μπορεί να κάνει τη ζωή μας πιο εύκολη, πιο πρακτική, πιο άνετη, πιο ποιοτική και να μας κερδίσει χρόνο. Με μια σχετική δαπάνη ανάλογα με τις ανάγκες μας, μπορούμε να αυτοματοποιήσουμε τα πάντα σε ένα σπίτι.

Όπως αναφέρθηκε και προηγουμένως, ένα έξυπνο και αυτοματοποιημένο σπίτι μπορεί να είναι όσο σύνθετο ή απλό επιθυμούμε. Η βασική ιδέα του έξυπνου σπιτιού είναι να εξοπλίσουμε την κατοικία με αισθητήρες και συστήματα ελέγχου και με τη βοήθεια τους να ελέγχουμε τη θερμοκρασία, τον κλιματισμό, τον φωτισμό ,τις καταναλώσεις και πολλές συσκευές. Προσαρμόζοντας περαιτέρω τους μηχανισμούς του σπιτιού στις ανάγκες του κατόχου του, το έξυπνο σπίτι μπορεί να αποτελέσει ένα πιο ασφαλές, πιο άνετο και πιο οικονομικό κατάλυμα. Παραδείγματος χάριν, ο ηλεκτρονικός ελεγκτής σε ένα αυτοματοποιημένο σπίτι μπορεί να αποφασίσει πότε οι ένοικοι έχουν αποσυρθεί από ένα δωμάτιο και να κλείσει τα φώτα ή να χαμηλώσει τον θερμοστάτη. Μπορεί ακόμη να ενεργοποιήσει συναγερμούς φωτιάς ή κλοπής, καθώς και να βελτιστοποιήσει τη διαχείριση πόρων, όπως το νερό και το ρεύμα.

1.2 Έξυπνο σπίτι

1.2.1 Ορίζοντας το "Έξυπνο σπίτι"

Επισήμως η λέξη «έξυπνο» χρησιμοποιήθηκε πρώτη φορά κατά τη δεκαετία του '70 ως προσδιορισμός τεχνολογικών επιτευγμάτων. Συγκεκριμένα, αναφερόταν σε στρατιωτικά προϊόντα, όπως βόμβες ή πυραύλους που καθοδηγούσαν τον εαυτό τους προς το στόχο («έξυπνες βόμβες»). Κατά την τεχνολογική άνθηση της δεκαετίας του '80 η λέξη «έξυπνο» απέκτησε άλλες προεκτάσεις: αναφερόταν σε συσκευές που εμπεριείχαν ολοκληρωμένα κυκλώματα, όπως οι υπολογιστές και οι προηγμένες

οικιακές συσκευές. Ο όρος «έξυπνο σπίτι» καθιερώθηκε από την αμερικανική ομοσπονδία κατασκευαστών σπιτιών το 1984. Ένας απλός ορισμός για την έννοια του έξυπνου σπιτιού είναι ο εξής:

«Έξυπνο σπίτι είναι μια κατοικία που ενσωματώνει ένα δίκτυο επικοινωνίας, το οποίο συνδέει ηλεκτρικές συσκευές και υπηρεσίες και επιτρέπει απομακρυσμένο έλεγχο, παρακολούθηση και πρόσβαση σε αυτές.»

Όταν μιλάμε για απομακρυσμένο έλεγχο εννοούμε ότι οι συσκευές και οι υπηρεσίες θα μπορούν να ελεγχθούν μέσα αλλά και έξω από την κατοικία. Ο ορισμός αυτός συμφωνεί με τις περισσότερες περιπτώσεις υλοποίησης έξυπνων σπιτιών, αφού σχεδόν πάντα έχουμε να κάνουμε με δικτύωση, αλληλεπίδραση συσκευών και έλεγχό τους με διάφορα σενάρια αυτοματισμού. Άρα συμπεραίνουμε πως με τον όρο έξυπνο υποδηλώνεται το τελικό αποτέλεσμα, δηλαδή οι υπηρεσίες που παρέχει ένα έξυπνο σπίτι και όχι τα επιμέρους στοιχεία του όπως οι αισθητήρες και οι διακόπτες.

1.2.2Δυνατότητες έξυπνου σπιτιού

Όπως αναφέρθηκε παραπάνω, ένα έξυπνο σπίτι αποτελείται από ποικίλους αισθητήρες, όπου οι συσκευές ελέγχονται ηλεκτρονικά και μπορούν να λειτουργούν αυτόματα με βάση κάποια προγραμματισμένα σενάρια.

Τέτοια σενάρια μπορούν να είναι για παράδειγμα:

- Τα παράθυρα και οι πόρτες να διαθέτουν διακόπτες που να δείχνουν πότε είναι ανοικτά.
- Οι γλάστρες να περιέχουν αισθητήρες υγρασίας που να προσδιορίζουν το νερό στο χώμα.
- Κάθε δωμάτιο να περιέχει έναν αισθητήρα θερμοκρασίας που να μετρά τη θερμοκρασία του χώρου.
- Ένας ανιχνευτής σε κάθε δωμάτιο να αισθάνεται πότε τα φώτα είναι ανοικτά.
- Κάθε ένας από αυτούς τους αισθητήρες συνδέεται με καλώδιο ή ασύρματα σε έναν υπολογιστή, ο οποίος μπορεί να ελέγξει τις τιμές και να εκτελέσει διάφορες ενέργειες.

Οι πιθανές αυτές ενέργειες περιλάβουν:

- Ήχος προειδοποίησης εάν κάποιος ξεχάσει στο σπίτι μια πόρτα ή ένα παράθυρο ανοικτό.
- Προειδοποίηση στον χρήστη όταν τα διάφορα φυτά στις γλάστρες απαιτούν νερό ή αυτόματο πότισμα εάν είναι υπάρχει διαθέσιμο σύστημα.
- Σβήσιμο της θέρμανσης όταν η θερμοκρασία στο δωμάτιο ανεβεί πιο πάνω από το προκαθορισμένο σημείο.
- Υποβολή έκθεσης σχετικά με πόση ενέργεια έχει χρησιμοποιηθεί για το φωτισμό κατά τη διάρκεια του προηγούμενου μήνα.

Αυτές είναι μερικές πιθανές ενέργειες που μπορεί να πραγματοποιήσει ένα "έξυπνο σπίτι", που ελέγχεται από έναν κεντρικό υπολογιστή.

1.2.3 Πλεονεκτήματα και χαρακτηριστικά του έξυπνου σπιτιού

Η ευκολία στην χρήση και οι ανέσεις που παρέχουν τα ολοκληρωμένα συστήματα αυτοματισμού είναι παράγοντες που αυξάνουν την αξία μεταπώλησης μιας κατοικίας.

Τα Συστήματα Αυτοματισμού βασίζονται σε δίκτυα δομημένης καλωδίωσης που έχουν σχεδιασθεί για να ανταποκριθούν στις ανάγκες δικτύωσης και επικοινωνιών του σύγχρονου σπιτιού. Θα είναι ικανά να υποστηρίζουν την νέα γενιά διαδραστικών υπηρεσιών φωνής, δεδομένων, video, πολυμέσων και τηλεπικοινωνιών, που συνήθως όλες μαζί αποκαλούνται η Παγκόσμια Λεωφόρος της Πληροφορίας.

Για να εξασφαλιστεί ότι η καλωδίωση του συστήματος αυτοματισμού του σπιτιού ξεπερνά τις προσδοκίες των σημερινών αγοραστών, εταιρείες έχουν αναπτύξει και υποστηρίξει την συνολική λύση αυτοματισμού "TOTAL HOME AUTOMATION SOLUTION" που περιλαμβάνει ένα ολοκληρωμένο σύστημα δομημένης καλωδίωσης και όλα τα απαιτούμενα ενεργά εξαρτήματα και συσκευές που συμμορφώνονται σε τυπικούς κανόνες σχεδιασμού τέτοιων συστημάτων.

Η καλωδιακή υποδομή μπορεί να προσαρμοσθεί σε οποιοδήποτε σπίτι, σύμφωνα με τις ιδιαίτερες ανάγκες ή τον τρόπο ζωής των ενοίκων του. Τα ολοκληρωμένα Συστήματα Αυτοματισμών της Κατοικίας σύντομα θα αποτελούν ένα τυπικό χαρακτηριστικό των νέων κατασκευών καθώς όσοι γνωρίζουν τις δυνατότητες από την σύγκλιση πληροφορικής και τηλεπικοινωνιών, θα τα απαιτούν όλο και περισσότερο.

Ωστόσο δεν είναι απαραίτητο ένα σπίτι να έχει από την αρχή υποδομή για να γίνει έξυπνο. Με τη χρήση μικροελεγκτών και με τη βοήθεια ασύρματων τεχνολογιών όπως Wi-Fi, Bluetooth, X10, ZigBee, μπορούμε να έχουμε το ίδιο αποτέλεσμα χωρίς προβλήματα καλωδιώσεων.

1.3 Ιδέα - Σύστημα προς υλοποίηση

Η αρχική ιδέα ήταν ο απλός έλεγχος μίας συσκευής από απόσταση ή πιο συγκεκριμένα η υλοποίηση ενός οικιακού αυτοματισμού από την αρχή μέχρι το τέλος, χωρίς προϋπάρχουσα υποδομή. Η ιδέα αυτή εξελίχθηκε με την προσθήκη αισθητήρων για τη δημιουργία περεταίρω ελέγχου και αυτοματοποίησης. Το σύστημα περιλαμβάνει αισθητήρα κίνησης για τη δημιουργία συστήματος συναγερμού, αισθητήρα θερμοκρασίας και υγρασίας για τον έλεγχο της εσωτερικής θερμοκρασίας του σπιτιού, αισθητήρα ανίχνευσης βροχής για τον έλεγχο τον παραθύρων και των τεντών, αισθητήρα ρεύματος και μέτρηση τάσης με τη βοήθεια του Arduino για τον έλεγχο συσκευών αλλά και των καταναλώσεων τους και αισθητήρα φωτεινότητας για τον έλεγχο του εσωτερικού ή του εξωτερικού φωτισμού. Επίσης διαθέτει τέσσερα relays για τη δυνατότητα σύνδεσης και ελέγχου πραγματικών συσκευών. Για κάθε ένα λοιπόν από αυτούς τους αισθητήρες έχουν υλοποιηθεί ένα ή περισσότερα σενάρια αυτοματισμών ανάλογα με τις μετρήσεις τους. Επίσης γίνεται καταγραφή των περισσότερων μετρήσεων σε μια βάση δεδομένων και παρουσιάζονται σε γραφικές παραστάσεις ως προς τον χρόνο έτσι ώστε να μπορεί ο χρήστης να έχει τη δυνατότητα καλύτερης και πιο εύκολης απεικόνισης και κατανόησης των μετρήσεων.

Ο τελικός χρήστης χρησιμοποιώντας τερματικές συσκευές - όπως για παράδειγμα έναν υπολογιστή ή ένα κινητό τηλέφωνο - και έχοντας πρόσβαση σε μια ιστοσελίδα στο internet αλλά και μέσω της εφαρμογής LabVIEW καθώς και σε τοπικό περιβάλλον με τη χρήση buttons, θα έχει τη δυνατότητα να ελέγχει κάποιες οικιακές συσκευές. Επιπλέον θα μπορεί να δει την κατάσταση του σπιτιού του σύμφωνα με τις αντίστοιχες μετρήσεις των αισθητήρων που έχουν καταγραφεί στη βάση δεδομένων, για τα διαστήματα όπου μπορεί να απουσίαζε από το σπίτι. Το σύστημα διαθέτει οθόνη η οποία μπορεί να απεικονίσει σε πραγματικό χρόνο τις μετρήσεις των αισθητήρων και ο χρήστης μπορεί να πραγματοποιήσει χειρισμούς στο σπίτι μέσω αυτής με τη βοήθεια μερικών buttons. Επίσης ο χρήστης του συστήματος,

χρησιμοποιώντας μια απλή εφαρμογή στο android κινητό του, μπορεί μέσω Bluetooth να δίνει φωνητικές εντολές για να ελέγξει φωτιστικά συσκευές.

ΚΕΦΑΛΑΙΟ 2: Τεχνολογίες, Πλατφόρμες και Εργαλεία Ανάπτυξης που χρησιμοποιήθηκαν

2.1 Τι είναι η HTML

Η HTML (HyperText Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα "<" και ">" (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Ανάμεσα στις ετικέτες, μπορεί να τοποθετηθεί κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα. Μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού.

2.1.1 Σήμανση HTML

Η σήμανση HTML αποτελείται από μερικά βασικά συστατικά, συμπεριλαμβανομένων των στοιχείων και των ιδιοτήτων τους, τους βασισμένους σε χαρακτήρες τύπους δεδομένων, τις αναφορές χαρακτήρων και τις αναφορές οντοτήτων. Ένα ξεχωριστό σημαντικό συστατικό είναι η δήλωση τύπου εγγράφου (document type declaration), η οποία ορίζει στον browser τον τρόπο εμφάνισης της σελίδας.

Στην HTML, το πρόγραμμα Hello world, φτιάχνεται με 9 γραμμές κώδικα στην HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>Hello HTML</title>
</head>
<body>
<p>Hello world</p>
</body>
</html>
```

Το κείμενο ανάμεσα στο <html>και το </html>περιγράφει την ιστοσελίδα, και το κείμενο μεταξύ του <body>και του </body>είναι το ορατό μέρος της. Το σημασμένο κείμενο <title>**Hello HTML**</title> καθορίζει τον τίτλο που θα εμφανίζεται στην μπάρα τίτλου του browser.)
Το Document Type Declaration στον πιο πάνω κώδικα είναι για την HTML5. Αν δεν συμπεριλαμβάνεται η δήλωση <!DOCTYPE html> τότε μερικοί browser θα καταφύγουν στην λειτουργία quirks για την εμφάνιση της σελίδας.

2.1.2 Παραδείγματα στοιχείων HTML

Κεφαλίδα του εγγράφου HTML: <head>...</head>.Συνήθως περιέχει τον τίτλο, π.χ.:

```
<head>
<title>ο τίτλος/span></title>
</head>
```

Επικεφαλίδες: οι επικεφαλίδες στην HTML ορίζονται με τις ετικέτες <h1> έως </h1>:

```
<h1>Επικεφαλίδα 1</h1>
<h2>Επικεφαλίδα 2</h2>
<h3>Επικεφαλίδα 3</h3>
<h4>Επικεφαλίδα 4</h4>
<h5>Επικεφαλίδα 5</h5>
<h6>Επικεφαλίδα 6</h6>
```

Παράγραφοι:

```
<p>Παράγραφος 1</p>  <p>Παράγραφος 2</p>
```

Αλλαγή γραμμής: **
**. Η διαφορά ανάμεσα στο **
**και το <p>είναι ότι το **
** αλλάζει γραμμή χωρίς να αλλάζει την σημαντική δομή της σελίδας, ενώ το <p>τεμαχίζει τη σελίδα σε παραγράφους. Το **
**είναι ένα άδειο στοιχείο, δηλαδή δεν έχει περιεχόμενο, ούτε και χρειάζεται ετικέτα τερματισμού.

```
<p>Αυτή <br>είναι μία παράγραφος <br> με <br> αλλαγή γραμμής</p>
```

Αυτός είναι ένας σύνδεσμος σε HTML: Για τη δημιουργία του χρησιμοποιείται η ετικέτα <a>. Η ιδιότητα href περιέχει τη σελίδα προορισμού του συνδέσμου.

```
<a href="http://el.wikipedia.org/">Ενας σύνδεσμος στη Βικιπαίδια</a>
```

2.1.3 Ιδιότητες HTML

Οι περισσότερες ιδιότητες των στοιχείων είναι ζεύγη ονομάτων και τιμών, τα οποία διαχωρίζονται με ένα «=» και γράφονται μέσα στην ετικέτα εκκίνησης ενός στοιχείου, μετά το όνομα του στοιχείου. Η τιμή μπορεί να περικλείεται σε μονά ή διπλά εισαγωγικά, παρότι τιμές που αποτελούνται από συγκεκριμένους χαρακτήρες μπορούν να γράφονται χωρίς εισαγωγικά στην HTML, αλλά όχι στην XHTML. Το να μένουν οι τιμές των ιδιοτήτων χωρίς εισαγωγικά θεωρείται ανασφαλές. Εκτός από τις

ιδιότητες που γράφονται ως ζεύγη ονομάτων και τιμών, υπάρχουν και μερικές οι οποίες επηρεάζουν το στοιχείο απλά με την παρουσία τους μέσα στην ετικέτα εκκίνησης.

Υπάρχουν και μερικές κοινές ιδιότητες οι οποίες εμφανίζονται σε πολλά στοιχεία:

Η ιδιότητα id παρέχει ένα αναγνωριστικό για ένα στοιχείο το οποίο είναι μοναδικό σε ολόκληρο το έγγραφο. Χρησιμοποιείται για να ταχτοποιεί το στοιχείο ώστε τα CSS να μπορούν να αλλάξουν τον τρόπο που αυτό εμφανίζεται, καθώς και τα σενάρια μπορούν να αλλάξουν, να μετακινήσουν ή να διαγράψουν τα περιεχόμενα ή την εμφάνισή του. Ακόμα, αν ένα id προστεθεί στο URL μιας σελίδας, παρέχει ένα μοναδικό αναγνωριστικό για ένα τμήμα της σελίδας για παράδειγμα: http://el.wikipedia.org/wiki/HTML#Ιδιότητες

- Η ιδιότητα class παρέχει τη δυνατότητα ταξινόμησης παρόμοιων αντικειμένων στην ίδια κλάση. Μπορεί να χρησιμοποιηθεί για να αποδώσει κάποια σημασία στο στοιχείο, ή για σκοπούς εμφάνισης. Για παράδειγμα, ένα έγγραφο HTML μπορεί να χρησιμοποιεί την επισήμανση class="notation" σε μερικά στοιχεία για να ξεχωρίσει από το υπόλοιπο κείμενο του εγγράφου. Κατά την εμφάνιση του εγγράφου, αυτά τα στοιχεία μπορεί -για παράδειγμα- να εμφανίζονται όλα μαζί στο τέλος της σελίδας ως υποσημειώσεις, άσχετα με την θέση που εμφανίζονται μέσα στον κώδικα. Επίσης οι ιδιότητες class χρησιμοποιούνται σημασιολογικά στα microformat . Ένα στοιχείο μπορεί να έχει πολλαπλές κλάσεις, για παράδειγμα το class="notation important" βάζει το στοιχείο τόσο στην κλάση «notation» όσο και στην «important».
- Η ιδιότητα style εφαρμόζει στυλ εμφάνισης σε συγκεκριμένα στοιχεία. Θεωρείται καλύτερη τακτική να χρησιμοποιούνται οι ιδιότητες id ή class ώστε να επιλέγεται το στοιχείο μέσα σε ένα CSS, αλλά μερικές φορές μπορεί να είναι πιο απλό να ανατεθούν style κατευθείαν στο στοιχείο.
- Η ιδιότητα title προσθέτει μια εξήγηση στο στοιχείο στο οποίο εφαρμόζεται. Στους περισσότερους φυλλομετρητές αυτή η ιδιότητα εμφανίζεται ως αναδυόμενο παράθυρο βοήθειας.
- Η ιδιότητα lang ταυτοποιεί την φυσική γλώσσα των περιεχομένων του στοιχείου, η οποία μπορεί να είναι διαφορετική από το υπόλοιπο έγγραφο.

Για παράδειγμα, μέσα σε ένα έγγραφο μπορεί να υπάρχει το κείμενο:

```
<p>Τι να κάνουμε;<span Lang="fr">c'est la vie</span>, όπως λένε και οι γάλλοι</p>
```

Το στοιχείο abbr μπορεί να χρησιμοποιηθεί για να επιδείξει μερικές από τις πιο πάνω ιδιότητες:

```
<abbr id="anId" class="jargon" style="color:purple;" title="Hypertext Markup
Language">HTML</abbr>
```

Αυτό το παράδειγμα εμφανίζεται ως: HTML. Στους περισσότερους φυλλομετρητές, όταν σταθεί ο δείκτης του ποντικιού πάνω από αυτό το στοιχείο, τότε θα εμφανιστεί ένα αναδυόμενο παράθυρο με το περιεχόμενο «Hypertext Markup Language».

Τα περισσότερα στοιχεία μπορούν επίσης να δεχτούν την ιδιότητα dir η οποία καθορίζει την κατεύθυνση του κειμένου. Η τιμή «rtl» της ιδιότητας, δείχνει κείμενο που διαβάζεται από δεξιά προς τα αριστερά, όπως για παράδειγμα στα Αραβικά, στα Περσικά ή στα Εβραϊκά.

2.1.4 Τύποι δεδομένων HTML

Στην HTML ορίζονται μερικοί τύποι δεδομένων για το περιεχόμενο των στοιχείων, όπως σενάρια εντολών ή stylesheet, και μια πληθώρα τύπων για τις τιμές των ιδιοτήτων, συμπεριλαμβανομένων των ID, των name, των URI, διαφόρων αριθμών και μονάδων μήκους, γλωσσών, τύπων αρχείων πολυμέσων, χρωμάτων, κωδικοποιήσεων χαρακτήρων, ημερομηνιών κλπ.

2.1.5 Δήλωση τύπου εγγράφου

Τα έγγραφα HTML πρέπει να αρχίζουν με μια Δήλωση τύπου εγγράφου (Document Type Declaration, ανεπίσημα λέγεται και «doctype»). Αυτή η δήλωση βοηθά τους browser να καταλάβουν πώς πρέπει να διαβάσουν το περιεχόμενο του εγγράφου και πώς να το παρουσιάσουν μετά, και ιδιαίτερα όταν χρησιμοποιείται το quirks mode.

Ο αρχικός σκοπός του doctype ήταν να επιτρέπει την ανάλυση και επιβεβαίωση των εγγράφων HTML από εργαλεία SGML τα οποία ήταν βασισμένα στο Document Type Definition (DTD). Το DTD στο οποίο αναφέρεται το DOCTYPE περιέχει μια γραμματική σε γλώσσα μηχανής, η οποία καθορίζει τι επιτρέπεται και τι απαγορεύεται να υπάρχει μέσα στο έγγραφο. Οι browser, από την άλλη, δεν υλοποιούν την HTML ως εφαρμογή της SGML και συνεπώς δεν διαβάζουν το DTD. Στην HTML5 δεν ορίζεται κανένα DTD, λόγω εγγενών τεχνολογικών περιορισμών. Έτσι το doctype <!doctype html> δεν αναφέρεται σε κανένα.

2.2 CSS

CSS σημαίνει Cascading Style Sheets και είναι στυλ που μπορούμε να ορίσουμε για τις HTML σελίδες. Γράφοντας τις σελίδες μας μόνο με HTML κώδικα, μπορούμε να ορίσουμε το χρώμα και το μέγεθος του κειμένου αλλά και άλλων στοιχείων της σελίδας (όπως πίνακες, links, λίστες κτλ). Για να αλλάξουμε το χρώμα κάποιου κειμένου ή το χρώμα ενός πίνακα, θα πρέπει να βρούμε το χρώμα αυτό μέσα στον κώδικα και να το αλλάξουμε. Η διαδικασία αυτή μπορεί να φαντάζει εύκολη όταν έχουμε να διαχειριστούμε μια μόνο σελίδα, αλλά ένα site αποτελείται από δεκάδες σελίδες τις οποίες χρειάζεται να διαχειριζόμαστε εύκολα και γρήγορα.

Φανταστείτε, για παράδειγμα, πόσο χρονοβόρο θα είναι αν θελήσουμε κάποια στιγμή να αλλάξουμε τα χρώματα στο κύριο μενού του site μας, το οποίο επαναλαμβάνεται σε όλες τις σελίδες. Σε μια τέτοια περίπτωση θα χρειαζόταν να ανοίγουμε κάθε σελίδα του site και να αλλάζουμε τα χρώματα του φόντου και των links του μενού, διαδικασία που εκτός από χρονοβόρα είναι και κουραστική.

Με την χρήση CSS μπορούμε να ορίζουμε χρώματα και μεγέθη οργανωμένα σε στυλ και έπειτα να εφαρμόζουμε τα στυλ αυτά στα στοιχεία των σελίδων του site μας. Με αυτόν τον τρόπο, κάθε φορά που αλλάζουμε το χρώμα ενός στυλ, αλλάζει το χρώμα όλων των στοιχείων που έχουν αναφορά στο στυλ αυτό. Έτσι αν έχουμε ορίσει ένα στυλ για το κύριο μενού του site, τότε θα χρειάζεται να αλλάξουμε το χρώμα του στυλ αυτού και αυτόματα θα εφαρμοστεί σε όλες τις σελίδες.

Εκτός από την ευκολία στην διαχείριση ενός site, ένα άλλο σημαντικό πλεονέκτημα της χρήσης CSS στις σελίδες είναι ο "καθαρότερος" κώδικας, χωρίς πολλές ιδιότητες στις ετικέτες οι οποίες τον κάνουν δυσανάγνωστο. Επιπλέον κάνει γρηγορότερη την πλοήγηση καθώς το αρχείο, μέσα στο οποίο ορίζονται τα στυλ, "διαβάζεται" από τον browser μόνο μια φορά και έπειτα αποθηκεύεται στην cache memory, μειώνοντας έτσι το μέγεθος της πληροφορίας που γίνεται download από τους browsers.

2.3 JavaScript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστραφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript.

Η Javascript δεν θα πρέπει να συγχέεται με τη Java, που είναι διαφορετική γλώσσα προγραμματισμού και με διαφορετικές εφαρμογές. Η χρήση της λέξης "Java" στο όνομα της γλώσσας έχει περισσότερη σχέση με το προφίλ του προϊόντος που έπρεπε να έχει και λιγότερο με κάποια πιθανή συμβατότητα ή άλλη στενή σχέση με τη Java. Ρόλο σε αυτήν τη σύγχυση έπαιξε και ότι η Java και η Javascript έχουν δεχτεί σημαντικές επιρροές από τη γλώσσα C, ειδικά στο συντακτικό, ενώ είναι και οι δύο αντικειμενοστραφείς γλώσσες. Τονίζεται ότι ο σωστός τρόπος γραφής της είναι "Javascript" και όχι 'Java script' σαν δύο λέξεις, όπως λανθασμένα γράφεται ορισμένες φορές.

2.4 PHP

Το ακρωνύμιο PHP αρχικά σήμαινε "Personal Home Page” (προσωπική αρχική σελίδα), επειδή το δημιούργησε το 1994 ο Rasmus Lerdorf για να παρακολουθεί τις

επισκέψεις στο δημοσιευμένο στο Διαδίκτυο βιογραφικό του. Καθώς η χρησιμότητα και οι δυνατότητες της PHP αυξάνονταν (και χρησιμοποιούνταν σε πιο επαγγελματικές περιστάσεις), κατέληξε να σημαίνει "PHP: Hypertext Preprocessor" (Προεπεξεργαστής Υπερ-κειμένου).

Σύμφωνα με την επίσημη τοποθεσία Ιστού της PHP, στη διεύθυνση www. php . net, η PHP είναι μια ευρέως διαδεδομένη γλώσσα σεναρίων γενικού σκοπού, ιδιαίτερα κατάλληλη για ανάπτυξη εφαρμογών Ιστού, και μπορεί να ενσωματωθεί σε κώδικα HTML. (Πρόκειται για ένα μακροσκελή αλλά περιγραφικό ορισμό, που θα εξηγήσω στη συνέχεια).

Ξεκινώντας από το τέλος, η δυνατότητα ενσωμάτωσης σε κώδικα HTML σημαίνει ότι μπορούμε να πάρουμε μια τυπική σελίδα HTML, να προσθέσουμε κώδικα PHP στη θέση που θέλουμε, και να καταλήξουμε σε ένα δυναμικό αποτέλεσμα. Αυτό το χαρακτηριστικό διευκολύνει τη χρήση της PHP από οποιονδήποτε έχει έστω και μια μικρή ιδέα από κώδικα HTML.

Επίσης, η PHP είναι μια γλώσσα σεναρίων που, αντίθετα από τις γλώσσες προγραμματισμού, έχει σχεδιαστεί για τη γραφή σεναρίων Ιστού και όχι αυτόνομων εφαρμογών (αν και με κάποια επιπλέον προσπάθεια μπορούμε τώρα πια να δημιουργήσουμε εφαρμογές σε PHP). Τα σενάρια της εκτελούνται μόνο μετά από ένα συμβάν. Για παράδειγμα, όταν ένας χρήστης υποβάλει μια φόρμα ή μόλις επισκεφθεί μια διεύθυνση URL.

Σε αυτόν τον ορισμό, θα πρέπει να προστεθεί ότι η PHP είναι μια τεχνολογία στην πλευρά του διακομιστή (sever-side). Αυτό σημαίνει ότι οτιδήποτε κάνει ο κώδικας PHP συμβαίνει στο διακομιστή. Για την εκτέλεση κώδικα PHP απαιτείται μια εφαρμογή διακομιστή Ιστού, όπως ο Apache ή ο IIS (Internet Information Services) της Microsoft, και όλα τα σενάρια PHP πρέπει να προσπελάζονται μέσω διευθύνσεων URL (http : // -κάτι). Επίσης, η PHP είναι μια διασυστημική τεχνολογία, που σημαίνει ότι εκτελείται στα περισσότερα λειτουργικά συστήματα, μεταξύ των οποίων τα Windows, Unix (και οι διάφορες παραλλαγές του), και Macintosh. Πιο σημαντικό ακόμη, τα σενάρια PHP που είναι γραμμένα για ένα διακομιστή λειτουργούν σε έναν άλλο διακομιστή με λίγες ή καθόλου τροποποιήσεις.

2.4.1 Πώς λειτουργεί η PHP

Όπως είπαμε προηγουμένως, η PHP είναι γλώσσα διακομιστή. Αυτό σημαίνει ότι ο κώδικας που γράφετε βρίσκεται σε έναν υπολογιστή υπηρεσίας (host computer) που ονομάζεται διακομιστής (server). Ο διακομιστής στέλνει ιστοσελίδες στους φυλλομετρητές των επισκεπτών (πελατών -clients) που τις ζητούν.

Όταν ένας χρήστης επισκέπτεται μια σελίδα γραμμένη σε PHP, ο διακομιστής διαβάζει τον κώδικα PHP (σενάριο) και εκτελεί τις εντολές του.

Η διαφορά από μια στατική ιστοσελίδα είναι ότι, στη δεύτερη περίπτωση, ο διακομιστής στέλνει τα δεδομένα HTML στο φυλλομετρητή χωρίς να ερμηνεύει τίποτε. Επειδή δεν απαιτείται κάποια άλλη ενέργεια του διακομιστή, μπορείτε να ανοίξετε HTML στο φυλλομετρητή σας χωρίς τη μεσολάβηση διακομιστή.

Στον τελικό χρήστη και στο φυλλομετρητή του, η διαφορά στην εμφάνιση μεταξύ των σελίδων home. html και home.php δε γίνεται αντιληπτή, αλλά το περιεχόμενο κάθε σελίδας δημιουργήθηκε με διαφορετικό τρόπο, και αυτή είναι η σημαντική διαφορά.

2.5 Τι είναι η MySQL;

Η MySQL (www.mysql.com) είναι η πιο δημοφιλής βάση δεδομένων ανοιχτού κώδικα στον κόσμο. Μάλιστα, σήμερα η MySQL συναγωνίζεται τους πανάκριβους Γολιάθ όπως οι Oracle και Microsoft SQL Server. Όπως και η PHP, η MySQL παρέχει εξαιρετική απόδοση, φορητότητα, και αξιοπιστία, με λίγο κόπο και μικρό έως μηδενικό κόστος.

Η MySQL είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (Relational Database Management System — RDBMS). Μια βάση δεδομένων, με απλά λόγια, είναι ένα σύνολο σχετιζόμενων μεταξύ τους δεδομένων (κείμενο, αριθμοί, ή δυαδικά αρχεία) αποθηκευμένων και οργανωμένων στο σύστημα DBMS.

Υπάρχουν πολλά είδη βάσεων δεδομένων, από βάσεις επιπέδων αρχείων (flat file) μέχρι τις σχεσιακές και τις αντικειμενοστραφείς. Μια σχεσιακή βάση δεδομένων χρησιμοποιεί πολλούς πίνακες για την αποθήκευση πληροφοριών διαχωρισμένων σε όσο το δυνατόν πιο απλά μέρη. Αν και οι φάσεις σχεδιασμού και προγραμματισμού των σχεσιακών βάσεων δεδομένων απαιτούν περισσότερη σκέψη, παρέχουν

βελτιωμένη αξιοπιστία και ακεραιότητα δεδομένων που αξίζουν τον επιπλέον κόπο. Επίσης, οι σχεσιακές βάσεις δεδομένων επιτρέπουν την ταυτόχρονη χρήση από πολλούς χρήστες, και η αναζήτηση σε αυτές γίνεται πιο εύκολα και γρήγορα.

Με την ενσωμάτωση μιας βάσης δεδομένων σε μια εφαρμογή Ιστού, κάποια από τα δεδομένα που παράγει η PHP μπορούν να ανακτηθούν από τη MySQL. Με αυτόν τον τρόπο, η τοποθεσία γίνεται ακόμη πιο ευέλικτη (η ευελιξία είναι το κλειδί για δυναμικές τοποθεσίες Ιστού).

Η MySQL είναι μια εφαρμογή ανοιχτού κώδικα, όπως και η PHP, πράγμα που σημαίνει ότι μπορείτε να τη χρησιμοποιήσετε και να την τροποποιήσετε ελεύθερα (μπορείτε να κατεβάσετε τον πηγαίο κώδικα της MySQL από το Διαδίκτυο). Σε κάποιες περιπτώσεις ίσως χρειαστεί να πληρώσετε για την άδεια χρήσης της MySQL, ειδικά αν κερδίζετε χρήματα από τις πωλήσεις ή την ενσωμάτωση του προϊόντος MySQL. Ελέγξτε ποια είναι η πολιτική αδειοδότησης όσο αφορά τη MySQL.

Το λογισμικό της MySQL αποτελείται από διάφορα τμήματα, που περιλαμβάνουν το διακομιστή MySQL (mysqld, ο οποίος εκτελεί και διαχειρίζεται τις βάσεις δεδομένων), τον πελάτη MySQL (mysql, που παρέχει μια διασύνδεση με το διακομιστή), και διάφορα βοηθητικά προγράμματα για συντήρηση και άλλες λειτουργίες. Η PHP υποστήριζε πάντα ικανοποιητικά τη MySQL, και αυτό ισχύει ακόμη περισσότερο στις πρόσφατες εκδόσεις της γλώσσας.

Η MySQL μπορεί να υποστηρίξει βάσεις δεδομένων με μέχρι 60000 πίνακες και περισσότερες από πέντε δισεκατομμύρια γραμμές. Η MySQL μπορεί να επεξεργαστεί πίνακες μεγέθους μέχρι οκτώ εκατομμύρια terabyte σε μερικά

λειτουργικά συστήματα, και γενικά μεγέθους μέχρι 4 GB. Μεταξύ άλλων, η MySQL χρησιμοποιείται από τη NASA και τη Στατιστική Υπηρεσία των Η.Π.Α.

2.6 Τεχνολογίες επικοινωνίας που χρησιμοποιήθηκαν

2.6.1 Bluetooth

Το Bluetooth είναι μια ασύρματη τεχνολογία που αρχικά σχεδιάστηκε για να αντικαταστήσει τα καλώδια κατά τις συνδέσεις κινητών τηλεφώνων ή laptop. Η τεχνολογία αυτή πήρε το όνομα της από έναν μεσαιωνικό βασιλιά της Νορβηγίας τον *Harald Blatand* (=Bluetooth) Gormson, που είχε προσπαθήσει να ενοποιήσει τα τότε βόρεια σκανδιναβικά βασίλεια. Έτσι και το Bluetooth σαν τεχνολογία προσπάθησε να ενοποιήσει τα ήδη υπάρχοντα ασύρματα πρότυπα, για να καθιερωθεί τελικά ως μια βασική τεχνολογία για την ασύρματη σύνδεση σε μικρές αποστάσεις.
Από τεχνικής πλευράς, το Bluetooth δουλεύει στα 2.4 GHz, που είναι το μη αδειοδοτημένο φάσμα ISM (Industrial Scientific Medical). Πρόκειται για φάσμα ανοιχτό στον οποιοδήποτε, με αποτέλεσμα όλα τα συστήματα που το χρησιμοποιούν να πρέπει να είναι έτοιμα για «διαμάχες» με άλλες συσκευές. Το Bluetooth για να αντιμετωπίσει τέτοιου είδους προβλήματα χρησιμοποιεί την τεχνολογία FHSS(Frequency Spectrum Hopping Technology), κάτι που σημαίνει ότι δύο συνδεδεμένες συσκευές Bluetooth μετά την αποστολή κάθε πακέτου, αλλάζουν περιοδικά κανάλι επικοινωνίας. Στην πράξη αυτό σημαίνει ότι εάν ένα κανάλι είναι δεσμευμένο από μια άλλη συσκευή, οι συσκευές Bluetooth θα αλλάξουν αυτόματα κανάλι με αποτέλεσμα η επικοινωνία τους να επηρεαστεί ελάχιστα. Η τεχνολογία FHSS διασφαλίζει ακόμη ότι πολλαπλά δίκτυα Bluetooth μπορούν να συνυπάρχουν ταυτόχρονα χωρίς να διαταράσσονται οι επιμέρους συνδέσεις.
Οι συσκευές που είναι συνδεδεμένες με Bluetooth μπορούν να δημιουργήσουν δίκτυα γνωστά ως piconets, που αποτελούνται από δύο έως οκτώ συσκευές. Από αυτές τις συσκευές μία δρα ως κύρια συσκευή (master) και είναι υπεύθυνη για τη διευθέτηση της κυκλοφορίας δεδομένων σε όλο το piconet. Οι υπόλοιπες συσκευές του piconet λειτουργούν ως «σκλάβοι» (slaves). Μόνο μια συσκευή μπορεί να είναι master κάθε στιγμή, αλλά αυτή η συσκευή μπορεί να αλλάξει αν μια συσκευή slave επιθυμεί να γίνει master. Η συσκευή master όχι μόνο αποτελεί το κέντρο του δικτύου, αλλά επίσης παίρνει όλες τις αποφάσεις με αποτέλεσμα να ορίζει και το συνολικό εύρος ζώνης του piconet.
Όταν μία ή περισσότερες συσκευές του piconet συνδεθούν σε ένα άλλο piconet, τότε έχουμε ένα scatternet. Οι συσκευές Bluetooth μπορεί να συμπεριφέρονται σαν σκλάβοι σε διάφορα piconet, αλλά master μπορούν να είναι αυστηρά μόνο σε ένα. Οι συσκευές αυτές που ανήκουν σε πολλά piconet μπορούν να δράσουν σαν συνδετικός κρίκος μεταξύ των υποδικτύων για τη μεταφορά δεδομένων.
Η μέση εμβέλεια για τις συσκευές Bluetooth είναι περίπου δέκα μέτρα, αλλά πιο ισχυρές συσκευές μπορούν να φτάσουν ακόμη και τα εκατό μέτρα. Σε γενικές γραμμές μπορούμε να πούμε ότι το Bluetooth είναι μια τεχνολογία με μικρή εμβέλεια, ειδικά σε σύγκριση με ασύρματες τεχνολογίες της κατηγορίας IEEE 802.11. Ωστόσο, η μικρή εμβέλεια των δικτύων Bluetooth κάποιες φορές αποτελεί πλεονέκτημα. Για παράδειγμα μία κινητή συσκευή που χρησιμοποιεί το δίκτυο Bluetooth είναι σίγουρα πολύ κοντά στο δέκτη, γεγονός που μπορεί να θεωρηθεί θετικό όσον αφορά την ασφάλεια. Η τεχνολογία Bluetooth μπορεί να χρησιμοποιηθεί επιτυχώς στο σύγχρονο έξυπνο σπίτι. Κάτι τέτοιο επιτυγχάνεται με την τοποθέτηση δεκτών διάσπαρτα μέσα στο σπίτι, οι οποίες εντοπίζουν το χρήστη μέσω του κινητού του τηλεφώνου και κατόπιν παρέχουν υπηρεσίες σε αυτόν ανάλογα με το που βρίσκεται (follow me content).

2.6.2 IEEE 802.11(Wi-Fi)

Η οικογένεια ασύρματων προτύπων IEEE 802.11 έχει γνωρίσει χαρακτηριστική αποδοχή και ανάπτυξη τα τελευταία χρόνια. Συνήθως αποκαλείται Wi-Fi ή πιο απλά WLAN. Τα πλεονεκτήματα της είναι αρκετά. Βασικότερο αυτών είναι η ικανοποίηση της ανάγκης που υπήρχε και υπάρχει για αντικατάσταση της παραδοσιακής καλωδιακής δικτύωσης των σπιτιών, των γραφείων και άλλων χώρων όπως οι καφετέριες κ.τ.λ. Ένα άλλο πλεονέκτημα είναι το χαμηλό κόστος των προϊόντων που είναι συμβατά με τα πρότυπα 802.11, όπως τα σημεία πρόσβασης (Access Points) και οι επαναλήπτες σήματος (Repeaters). Επίσης, τα πιο πρόσφατα πρότυπα της οικογένειας IEEE 802.11 παρέχουν αρκετά μεγάλες ταχύτητες μετάδοσης δεδομένων. Για παράδειγμα το πρότυπο 802.11η μπορεί να επιτύχει ρυθμό μετάδοσης δεδομένων πάνω από 500 Mbps. Η διάδοση αυτής της τεχνολογίας έφερε στο χρήστη εύκολη και πολλές φορές δωρεάν πρόσβαση στο διαδίκτυο σε διάφορους δημόσιους χώρους. Τέλος, τα πρότυπα IEEE 802.11 λειτουργούν και αυτά στο μη αδειοδοτημένο φάσμα ISM (Industrial Scientific Medical) στη συχνότητα των 2.4 GHz.

Η τοπολογία δικτύου που χρησιμοποιείται είναι «κεντροποιημένη». Αυτό πρακτικά σημαίνει ότι όλοι οι πελάτες (clients) ενός δικτύου συνδέονται σε μια κεντρική οντότητα, που συχνά αναφέρεται ως σημείο πρόσβασης. Αυτή η οντότητα λαμβάνει δεδομένα από τους πελάτες και κατόπιν τα δρομολογεί. Σήμερα γίνεται προσπάθεια παροχής δικτύων «mesh». Με ένα τέτοιο δίκτυο οι συσκευές δε θα χρειάζονταν μια κεντρική οντότητα. Αντιθέτως, θα δρουν αυτόνομα προωθώντας τα αντίστοιχα δεδομένα μεταξύ τους. Πρέπει να σημειωθεί εδώ ότι οι ανάγκες σε ενέργεια για τις διάφορες συσκευές με κάρτες που υποστηρίζουν τα πρότυπα IEEE 802.11 κυμαίνονται μεταξύ 2.06W και 0.14W όταν αυτές είναι σε αναμονή.

Το τελευταίο αυτό χαρακτηριστικό αποτελεί και το βασικότερο μειονέκτημα της συγκεκριμένης τεχνολογίας όσον αφορά την ευρεία χρησιμοποίησή της στα έξυπνα σπίτια. Και αυτό γιατί οι αισθητήρες που αποτελούν βασικό συστατικό ενός έξυπνου σπιτιού δεν πρέπει να καταναλώνουν μεγάλα ποσά ενέργειας. Τη λύση σε αυτό το πρόβλημα τη δίνει η τεχνολογία δικτύωσης ZigBee.

2.7 Μικροελεγκτές

Η σχεδίαση ενός μικροϋπολογιστικού συστήματος, ενός συστήματος δηλαδή βασισμένο σε μικροεπεξεργαστή, ο οποίος ελέγχει μια πληθώρα από περιφερειακές συσκευές, περιλαμβάνει, το ολοκληρωμένο κύκλωμα του μικροεπεξεργαστή και τις περιφερειακές συσκευές όπως μνήμες, συσκευές εισόδου/εξόδου, αισθητήρες κ.τ.λ. Η σύνδεση και ο συγχρονισμός των διαφόρων συσκευών με τον μικροεπεξεργαστή καθιστά πολύπλοκη τη σχεδίαση του συστήματος. Μια λύση στο πρόβλημα θα ήταν να υπήρχε ένα ολοκληρωμένο κύκλωμα, το οποίο να ενσωματώνει κάποιες απαραίτητες περιφερειακές συσκευές. Ένα τέτοιο κύκλωμα είναι ο μικροελεγκτής.

Ένας μικροελεγκτής έχει ενσωματωμένη μνήμη και θύρες Εισόδου/Εξόδου (I/O), ενώ αντίθετα σε ένα μικροεπεξεργαστή χρειάζεται να συνδέσουμε τα παραπάνω σαν πρόσθετα περιφερειακά κυκλώματα.

Οι μικροελεγκτές είναι το κύριο στοιχείο σε πολλά είδη ηλεκτρονικού εξοπλισμού. Ένα τυπικό σπίτι στο δυτικό κόσμο είναι πιθανό να περιλαμβάνει μόνο έναν ή δύο γενικού σκοπού μικροεπεξεργαστές, αλλά περισσότερους από είκοσι μικροελεγκτές. Μικροελεγκτές απαντώνται σε οποιονδήποτε τύπο ηλεκτρικής συσκευής, πλυντήρια ρούχων, φούρνους μικροκυμάτων, τηλέφωνα κ.λπ.

Οι περισσότεροι μικροελεγκτές βασίζονται στην αρχιτεκτονική Von Neumann, η οποία καθόρισε σαφώς τα τέσσερα βασικά συστατικά που απαιτούνται για ένα ψηφιακό σύστημα. Αυτά περιλαμβάνουν έναν επεξεργαστικό πυρήνα (CPU), τη μνήμη για το πρόγραμμα και τα δεδομένα (RAM), χώρο μόνιμης αποθήκευσης (FLASH σε έναν Μικροελεγκτή), καθώς επίσης και τις θύρες I/O για επικοινωνία με εξωτερικές περιφερειακές μονάδες - όλο αυτά σε ένα ενιαίο ολοκληρωμένο κύκλωμα.

Οι μικροελεγκτές συνήθως περιλαμβάνουν θύρες I/O για σύνδεση με ποικίλες συσκευές εισόδου/εξόδου, όπως αναλογικές (με ενσωματωμένους μετατροπείς από αναλογικό σε ψηφιακό), UARTs ή ειδικευμένα σειριακά interface επικοινωνιών όπως το PC, το SPI και το Controller Area Network (CAN). Μερικοί σύγχρονοι μικροελεγκτές έχουν ενσωματωμένη και μία υψηλού επιπέδου γλώσσα προγραμματισμού όπως η BASIC.

2.7.1 Arduino

Το 2003 στην πόλη Ιβρέα της Ιταλίας (η Ιβρέα βρίσκεται έξω από το Τορίνο) ένας φοιτητής στα πλαίσια μια εργασίας ανέπτυξε το Wiring project. Το project αυτό ήταν η ανάπτυξη ενός απλού συστήματος βασισμένου σε μικροελεγκτή που να προσφέρεται για εύκολη και γρήγορη ανάπτυξη εφαρμογών. Το Arduino ήταν μια «επέκταση» του Wiring project με σκοπό να χρησιμοποιηθεί από ανθρώπους μη σχετικούς με την τεχνολογία. Με αυτό το στόχο το Arduino αποτέλεσε ταυτόχρονα και το καλύτερο ξεκίνημα για φοιτητές και μηχανικούς. Το Arduino πήρε το όνομά του από τον Arduin που ήταν στρατιωτικός διοικητής της Ιβρέα και αργότερα βασιλιάς της Ιταλίας. Το Arduino είναι μία πλατφόρμα ανάπτυξης έργων (φυσικού υπολογισμού όπως λέγεται), ανοιχτού κώδικα. Η πλατφόρμα περιλαμβάνει μία απλή πλακέτα μικροελεγκτή (τα αντίστοιχα σχέδια είναι δωρεάν) μαζί με ένα περιβάλλον ανάπτυξης προγραμμάτων (επίσης δωρεάν) για τον προγραμματισμό του μικροελεγκτή. Το Arduino χρησιμοποιεί μικροελεγκτές Atmel AVR ενώ υπάρχουν παραλλαγές με διαφορετική έκδοση του μικροελεγκτή που διαθέτουν περισσότερες ή λιγότερες δυνατότητες. Η γλώσσα προγραμματισμού που χρησιμοποιείται για τον προγραμματισμό του μικροελεγκτή είναι η Wiring C, η οποία αποτελεί παραλλαγή της γλώσσας C++ ενώ μέσα από την προσθήκη βιβλιοθηκών υποστηρίζονται περισσότερες λειτουργίες και δυνατότητες. Το περιβάλλον ανάπτυξης που χρησιμοποιεί το Arduino είναι δανεισμένο από τη γλώσσα Processing. Τα βασικότερα πλεονεκτήματα του Arduino (http://arduino.cc) είναι:

- Χαμηλό κόστος. Μπορεί κάποιος να αγοράσει το Arduino Uno R3 με κόστος μικρότερο από 12 € (στο ebay) και δωρεάν μεταφορικά. Επίσης, υπάρχουν starter kit που περιλαμβάνουν εκτός από την πλακέτα και μία πλειάδα από αισθητήρες, καλώδια, led, διακόπτες, οθόνες lcd, κλπ. που κοστίζουν λιγότερο από 40 €. Όσο για το περιβάλλον ανάπτυξης, αυτό είναι ανοιχτού κώδικα και μπορεί να το κατεβάσει κάποιος δωρεάν από το επίσημο site του Arduino en/MaiNSoftware).

- Συμβατότητα. Το λογισμικό του Arduino εκτελείται σε περιβάλλοντα Windows, Macintosh και Linux.
- Απλότητα. Το περιβάλλον ανάπτυξης είναι αρκετά απλό ώστε να χρησιμοποιείται εύκολα και από αρχάριους χρήστες.

- Επεκτάσιμο λογισμικό ανοιχτού κώδικα. Τα προγράμματα του Arduino είναι ανοιχτού κώδικα που σημαίνει ότι οποιοσδήποτε μπορεί να βρει τμήματα πηγαίου κώδικα, να τα μελετήσει και να τα τροποποιήσει σύμφωνα με τις ανάγκες του. Επίσης, μέσα από την προσθήκη βιβλιοθηκών C++ μπορούν να προστεθούν δυνατότητες στο περιβάλλον ανάπτυξης. Κάποιος πεπειραμένος χρήστης μπορεί να γράψει δικό του κώδικα χαμηλού επιπέδου και να τον καταστήσει διαθέσιμο μέσω βιβλιοθηκών, ενώ οι αρχάριοι χρήστες μπορούν να προσθέσουν υποστήριξη για νέες διατάξεις (αισθητήρες, κ.τ.λ.) χωρίς να χρειάζεται να γνωρίζουν τις χαμηλού επιπέδου λεπτομέρειες του προγραμματισμού των διατάξεων αυτών.

2.7.2 **Εκδόσεις** Arduino

Υπάρχουν πολλές εκδόσεις και παραλλαγές του Arduino (http://arduino.cc). Κάθε board υποστηρίζει συγκεκριμένο τύπο μικροελεγκτή και διαθέτει ένα σύνολο ακροδεκτών για είσοδο και έξοδο. οι ακροδέκτες μπορούν να χωριστούν σε ακροδέκτες εισόδου/εξόδου (ανταλλαγή σημάτων με εξωτερικά συστήματα, συσκευές και κυκλώματα) και ακροδέκτες που σχετίζονται με την τροφοδοσία. Το Arduino μπορεί να διαβάσει αναλογικά σήματα (όλες τις στάθμες από 0 έως 5V) καθώς και ψηφιακά σήματα (Ο ή 5 V). Για παράδειγμα, κάποιος αισθητήρας (π.χ. θερμοκρασίας, επιτάχυνσης) μπορεί να δίνει αναλογική ή ψηφιακή έξοδο η οποία θα πρέπει στη συνέχεια να συνδεθεί στο Arduino για να είναι δυνατή η αξιοποίηση των σημάτων. Οι έξοδοι υποστηρίζουν μόνο ψηφιακά σήματα.

Επιπλέον, το Arduino μπορεί να προσφέρει τροφοδοσία σε εξωτερικές συσκευές (π.χ. αισθητήρας) στα 3.3 ή τα 5V ενώ δεν μπορεί να δώσει στις εξόδους μεγαλύτερο ρεύμα από 30-40mA. Αυτός είναι ένας ειδικός περιορισμός όταν θέλουμε να ελέγξουμε μια εξωτερική συσκευή όπως ένα μοτέρ που συνήθως απαιτεί περισσότερο ρεύμα.

Τέλος, η σύνδεση USB επιτρέπει την επικοινωνία του με τον υπολογιστή (π.χ. upload κώδικα του χρήστη, εμφάνιση βοηθητικών μηνυμάτων από το Arduino στον υπολογιστή) ενώ παρέχει ταυτόχρονα και τροφοδοσία 5 V για τη λειτουργία του. Μετά τον προγραμματισμό του, το Arduino μπορεί να λειτουργήσει αυτόνομα χωρίς υπολογιστή αρκεί να τροφοδοτείται από την ειδική είσοδο για να μπορεί να λειτουργεί (συνήθως με μπαταρία).

Η Flash Memory φιλοξενεί τα προγράμματα του χρήστη. Όπως ίσως να γνωρίζετε από τον προγραμματισμό, ο πηγαίος κώδικας(πχ σε γλώσσα c) δεν μπορεί να εκτελεστεί διότι δεν είναι κατανοητός από τον υπολογιστή. Αυτό που τρέχει τελικά είναι ο εκτελέσιμος κώδικας που έχει παραχθεί από τη μεταγλώττιση του πηγαίου κώδικα. Θα πρέπει σε αυτό το σημείο να αναφερθεί ότι ο εκτελέσιμος κώδικας καταλαμβάνει περισσότερο χώρο (byte) από τον πηγαίο. Έτσι και στην περίπτωση του Arduino, ο κώδικας που τελικά παράγεται από το περιβάλλον ανάπτυξης είναι αυτός μεταφορτώνεται από τον υπολογιστή στον μικροελεγκτή.

Ο τελικός αυτός κώδικας αποθηκεύεται στη Flash Memory τόσο για να είναι δυνατή η εκτέλεση του όσο και για να μένει αποθηκευμένος όσο απουσιάζει η τροφοδοσία από τον μικροελεγκτή.

Στη συνέχεια μπορείτε να δείτε τα γενικά χαρακτηριστικά από τις πιο διαδεδομένες εκδόσεις Arduino που χρησιμοποιούνται σήμερα.

Το πρωτότυπο υλισμικό του Arduino κατασκευάζεται από την Ιταλική εταιρία Smart Projects. Κάποιες πλακέτες με την μάρκα του Arduino έχουν σχεδιαστεί από την Αμερικάνικη εταιρία SparkFun Electronics. Δεκαέξι εκδοχές του Arduino Hardware έχουν χρησιμοποιηθεί εμπορικά μέχρι τώρα:

1. Το Serial Arduino, προγραμματισμένο με μία σειριακή DE-9 σύνδεση χρησιμοποιώντας τεχνολογία ATmega8

2. Το Arduino Extreme, με ένα USB interface για προγραμματισμό χρησιμοποιώντας τεχνολογία ATmega8

3. Το Arduino Mini, μία έκδοση μινιατούρας του Arduino χρησιμοποιώντας τεχνολογία surface-mounted ATmega168

4. Το Arduino Nano, ένα ακόμα πιο μικρό, USB τροφοδοτούμενη εκδοχή του Arduino χρησιμοποιώντας τεχνολογία surface-mounted ATmega168 (ATmega328 για την νεότερη έκδοση)

5. Το LilyPad Arduino, ένα μινιμαλιστικό σχέδιο για εφαρμογές ένδυσης και E-textiles χρησιμοποιώντας τεχνολογία surface-mounted AT-mega328

6.Το Arduino NG, με ένα USB interface για προγραμματισμό και χρησιμοποιώντας τεχνολογία ATmega8

7. Το Arduino NG plus, με ένα USB interface για προγραμματισμό και χρησιμοποιώντας τεχνολογία atmega168

8. Το Arduino Bluetooth, με Bluetooth interface για προγραμματισμό χρησιμοποιώντας τεχνολογία ATmega168

9. Το Arduino Diecimila, με ένα USB interface και χρησιμοποιεί τεχνολογία ATmega168 σε ένα DIP28 πακέτο

10. Το Arduino Duemilanove (“2009”), χρησιμοποιεί τεχνολογία ATmega168 (ATmega328 για την καινούργια έκδοση) και τροφοδοτείτε μέσω ενέργειας USB/DC, αυτόματα εναλλασσόμενης

11. Το Arduino Mega, χρησιμοποιώντας τεχνολογία surface-mounted ATmega1280 για περαιτέρω I/O και μνήμη[11]

12. Το Arduino Uno, χρησιμοποιώντας την ίδια τεχνολογία ATmega328 όπως το τελευταίο μοντέλο Duemilanove, αλλά ενώ το Duemilanove χρησιμοποιεί ένα FTDI chipset για το USB, το Uno χρησιμοποιεί τεχνολογία ATmega8U2 προγραμματισμένο ως σειριακός μετατροπέας

13. Το Arduino Mega2560, χρησιμοποιεί τεχνολογία surface-mounted ATmega2560 φέρνοντας την ολική μνήμη στα 256kB. Επίσης ενσωματώνει την νέα τεχνολογία ATmega8U2 (ATmega16U2 σε αναθεώρηση τύπου 3) USB chipset.

14. Το Arduino Leonardo, με ένα ATmega32U4 chip που εξαλείφει την ανάγκη για συνδεσιμότητα μέσω USB και μπορεί να χρησιμοποιηθεί ως ψηφιακό πληκτρολόγιο ή ποντίκι. Κυκλοφόρησε στο Maker Faire Bay Area το 2012

15. Το Arduino Esplora, με εμφάνιση που παραπέμπει σε χειριστήριο κονσόλας βιντεοπαιχνιδιών με joystick και ενσωματωμένους αισθητήρες για ήχο, φώς, θερμοκρασία και επιτάχυνση

16. Το Arduino Due είναι ένα μικρό χειριστήριο board βασισμένο στην τεχνολογία Atmel SAM3X8E ARM Cortex-M3 CPU. Είναι το πρώτο board της Arduino βασισμένη σε επεξεργαστή 32-bit ARM microcontroller

Η έκδοση του arduino που εφαρμόζεται στην παρούσα εργασία είναι αυτή του Arduino Mega 2560. Σε αρχικό στάδιο οι εφαρμογή χτιζόταν πάνω σε Arduino UNO αλλά λόγω του μικρού αριθμού pin που παρέχει για αναλογικές και ψηφιακές εξόδους αλλά και λόγο της περιορισμένης μνήμης, έγινε τελικά επιλογή του Arduino Mega. Παρακάτω θα αναφερθούμε σε κάποια γενικά χαρακτηριστικά και τις δυνατότητες του Arduino Mega, τα υπόλοιπα στο κεφάλαιο 3.1.

Όπως είπαμε στο Arduino Mega χρησιμοποιείται διαφορετικός τύπος και φυσική μορφή μικροελεγκτή η οποία διαθέτει μεγαλύτερο πλήθος ακροδεκτών σε σχέση με την έκδοση UNO.

Από τις 14 ψηφιακές εισόδους/εξόδους του UNO πάμε στις 54 και από τις 6 αναλογικές στις 16. Το πλεονέκτημα των πολλών ακροδεκτών προσφέρει τη δυνατότητα απευθείας σύνδεσης με πολλές εξωτερικές συσκευές και διατάξεις. Το περιορισμένο πλήθος ακροδεκτών αντιθέτως θα μας ανάγκαζε να χρησιμοποιήσουμε και ενδιάμεσα κυκλώματα(πχ αποκωδικοποιητές) προκειμένου να ελέγχουμε μεγαλύτερο πλήθος εξωτερικών κυκλωμάτων. Φυσικά η επιλογή εξαρτάται από το είδος της εφαρμογής που έχουμε προς ανάπτυξη.

Ένα άλλο σημαντικό χαρακτηριστικό του Mega 2560 είναι η αυξημένη χωρητικότητα της flash memory(256KB)

2.8 Πακέτο προγραμμάτων XAMPP

2.8.1 XAMPP

Το **XAMPP** είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού, λογισμικού ανοικτού κώδικα και ανεξαρτήτου πλατφόρμας το οποίο περιέχει τον εξυπηρετητή ιστοσελίδων http Apache, την βάση δεδομένων MySQL και ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP και Perl.

Ετυμολογία

Το XAMPP είναι ακρωνύμιο και αναφέρεται στα παρακάτω αρχικά:

- X (αναφέρεται στο «cross-platform» που σημαίνει λογισμικό ανεξάρτητο πλατφόρμας)
- Apache HTTP εξυπηρετητής
- MySQL
- PHP
- Perl

Το XAMPP είναι ένα ελεύθερο λογισμικό το οποίο περιέχει ένα εξυπηρετητή ιστοσελίδων το οποίο μπορεί να εξυπηρετεί και δυναμικές ιστοσελίδες τεχνολογίας PHP/MySQL. Είναι ανεξάρτητο πλατφόρμας και τρέχει σε Microsoft Windows, Linux, Solaris, and Mac OS X και χρησιμοποιείται ως πλατφόρμα για την σχεδίαση και ανάπτυξη ιστοσελίδων με την τεχνολογίες όπως PHP, JSP και Servlets.

2.8.2 Apache HTTP εξυπηρετητής

Ο Apache HTTP γνωστός και απλά σαν Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού (web). Όποτε ένας χρήστης επισκέπτεται ένα ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί με έναν διακομιστή (server) μέσω του πρωτοκόλλου HTTP, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης. Ο Apache είναι ένας από τους δημοφιλέστερους εξυπηρετητές ιστού, εν μέρει γιατί λειτουργεί σε διάφορες πλατφόρμες όπως τα Windows, το Linux, το Unix και το Mac OS X. Κυκλοφόρησε υπό την άδεια λογισμικού Apache και είναι λογισμικό ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοικτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation).

Ο Apache χρησιμοποιείται και σε τοπικά δίκτυα σαν διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων π.χ. Oracle, MySQL.

Η πρώτη του έκδοση, γνωστή ως NCSA HTTPd, δημιουργήθηκε από τον Robert McCool και κυκλοφόρησε το 1993. Θεωρείται ότι έπαιξε σημαντικό ρόλο στην αρχική επέκταση του παγκόσμιου ιστού. Ήταν η πρώτη βιώσιμη εναλλακτική επιλογή που παρουσιάστηκε απέναντι στον εξυπηρετητή http της εταιρείας Netscape και από τότε έχει εξελιχθεί στο σημείο να ανταγωνίζεται άλλους εξυπηρετητές βασισμένους στο Unix σε λειτουργικότητα και απόδοση. Από το 1996 ήταν από τους πιο δημοφιλείς όμως από τον Μάρτιο του 2006 έχει μειωθεί το ποσοστό της εγκατάστασής του κυρίως από τον Microsoft Internet Information Services και την πλατφόρμα .NET. Τον Οκτώβριο του 2007 το μερίδιο του ήταν 47.73% από όλους τους

2.8.3 Δυνατότητες και απαιτήσεις XAMPP

Το XAMPP προϋποθέτει μόνο τα λογισμικά συμπίεσης αρχείων zip, tar, 7z ή exe κατά την διάρκεια της εγκατάστασης. Το XAMPP έχει δυνατότητα αναβάθμισης σε νέες εκδόσεις του εξυπηρετητή ιστοσελίδων http Apache, της βάσης δεδομένων MySQL, της γλώσσας PHP και Perl. Το XAMPP συμπεριλαμβάνει επίσης τα πακέτα OpenSSL και το phpMyAdmin.

2.8.4 Χρήση XAMPP

Επίσημα οι σχεδιαστές του XAMPP προόριζαν το λογισμικό ως εργαλείο ανάπτυξης και δοκιμής ιστοσελίδων τοπικά στον υπολογιστή χωρίς να είναι απαραίτητη η

σύνδεση στο διαδίκτυο. Για να είναι δυνατή η χρήση του, πολλές σημαντικές λειτουργίες ασφάλειας έχουν απενεργοποιηθεί. Στην πράξη το XAMPP ορισμένες φορές χρησιμοποιείται και για την φιλοξενία ιστοσελίδων. Υπάρχει ειδικό εργαλείο το οποίο περιέχεται στο XAMPP για την προστασία με κωδικό των σημαντικών μερών. Το XAMPP υποστηρίζει την δημιουργία και διαχείριση βάσεων δεδομένων τύπου MySQL και SQLite.

Όταν το XAMPP εγκατασταθεί στον τοπικό υπολογιστή διαχειρίζεται τον localhost ως ένα απομακρυσμένο κόμβο, ο οποίος συνδέεται με το πρωτόκολλο μεταφοράς αρχείων FTP. Η σύνδεση στον localhost μέσω του FTP μπορεί να γίνει με το όνομα χρήστη «newuser» και το κωδικό «wampp». Για την βάση δεδομένων MySQL υπάρχει ο χρήστης «root» χωρίς κωδικό πρόσβασης.

2.9 Γραφήματα με JavaScript AmCharts

Πρόκειται για μια αυτόνομη και ανεξάρτητη προηγμένη βιβλιοθήκη JavaScript/HTML5 που παρέχει δωρεάν γραφήματα και διαγράμματα σε μορφές όπως στήλη, μπάρες, "πίτα", γραμμική, πυραμίδα κλπ και δίνει επίσης και τη δυνατότητα μετρήσεων. Τα γραφήματα αυτά υποστηρίζονται και από τα σύγχρονα κινητά τηλέφωνα. Πρόκειται για μία ελεύθερη βιβλιοθήκη η οποία διαθέτει ακόμα και τα επί πληρωμή προϊόντα της ελεύθερα με τον περιορισμό πως αναγράφεται ένα μικρό link στα γραφήματα.

Επιπροσθέτως υπάρχει live editor στον browser ο οποίος επιτρέπει στον χρήστη να δημιουργεί γραφήματα διάφορων ειδών απλά εισάγοντας δεδομένα, αποτελώντας μια ιδανική λύση για όσους δεν διαθέτουν γνώσεις προγραμματισμού.

Επιπλέον διαθέτει χρηματιστηριακά γραφήματα και οικονομικά διαγράμματα τα οποία μπορούν να χρησιμοποιηθούν για την οπτικοποίηση οποιουδήποτε δεδομένου με βάση τον χρόνο. Έχει ενσωματωμένο επιλογέα ημερομηνίας και δεδομένων πράγμα που επιτρέπει να προσθέσουμε νέες γραφικές χωρίς καινούριο κώδικα.

Τέλος χρησιμοποιεί και βιβλιοθήκες χαρτών με σκοπό τη δημιουργία διαδραστικών κινούμενων χαρτών.

Σε όλα τα γραφήματα μπορούν να προστεθούν χαρακτηριστικά όπως κύλιση(scroll) και μεγέθυνση, διάφορα χρώματα και θέματα εμφάνισης. Επίσης υπάρχει η δυνατότητα εξαγωγής σε pdf.

2.10Fritzing

Το Fritzing είναι ένα open source σχεδιαστικό πρόγραμμα. Έχει δημιουργηθεί στο πνεύμα της γλώσσας Processing και του Arduino και επιτρέπει στον σχεδιαστή να μετατρέψει το πρωτότυπο Arduino σχέδιό του σε PCB.

Πρόκειται ίσως για τον πιο εύκολο τρόπο σχεδίασης κυκλωμάτων πλακετών. Διάφορες εταιρίες ηλεκτρονικών διαθέτουν ελεύθερες βιβλιοθήκες με αισθητήρες και ηλεκτρονικά εξαρτήματα και εργαλεία που μπορούμε να προσθέτουμε στα σχέδιά μας. Επίσης ο κάθε χρήστης έχει τη δυνατότητα να σχεδιάσει μόνος του κάποιο εξάρτημα ή κάποιο αισθητήρα.

Το πρόγραμμα μπορούμε να το κατεβάσουμε ελεύθερα από την επίσημη ιστοσελίδα http://fritzing.org/home/. Όλα τα σχέδια που αφορούν το Arduino και παρουσιάζονται στο υπόλοιπο μέρος της πτυχιακής εργασίας σχεδιαστεί στο Fritzing.

2.11 Λογισμικό LabVIEW

Το LabVIEW ® (Laboratory Virtual Instrument Engineering Workbench) είναι μία πανίσχυρη γλώσσα προγραμματισμού instrumentation και ανάλυσης για υπολογιστές. Τρέχει στα λειτουργικά συστήματα : Win(3.1, 95, NT, 2K, XP, Vista), Solaris, Macintosh, HP-UX. Το LabVIEW ξεφεύγει από την παραδοσιακή φύση γλωσσών προγραμματισμού, εισάγοντας τον χρήστη σε ένα γραφικό περιβάλλον για instrumentation με όλα τα εργαλεία για συλλογή μετρήσεων, έλεγχο αυτόνομων οργάνων, ανάλυση και παρουσίαση. Σε αυτή τη γραφική γλώσσα προγραμματισμού που λέγεται “G”, μπορείτε να φτιάξετε το πρόγραμμα σας σε διαγραμματικά μπλοκ, κάτι πολύ φυσικό για μηχανικούς και επιστήμονες. Αφού φτιάξετε τα διαγράμματα σας, το LabVIEW τα μεταφράζει σε κώδικα μηχανής. Το LabVIEW αποτελεί ένα ενιαίο σύστημα συλλογής μετρήσεων, ανάλυσης, ελέγχου και παρουσίασης. Για την συλλογή μετρήσεων και έλεγχο οργάνων, το LabVIEW υποστηρίζει RS-232/422/485, IEEE488(GPIB), VISA, VXI, καθώς επίσης και κάρτες δειγματοληψίας. Μία ολοκληρωμένη βιβλιοθήκη από drivers για όργανα, διευκολύνει τον έλεγχο αυτόνομων οργάνων. Για την ανάλυση δεδομένων, υπάρχουν ρουτίνες Επεξεργασίας Σήματος, Φίλτρων, Στατιστικής, Παρεμβολής, Γραμμικής Άλγεβρας κ.ο.κ. Τέλος, όντας γραφικό στην φύση του, το LabVIEW παρουσιάζει πολλά πλεονεκτήματα σαν σύστημα παρουσίασης αποτελεσμάτων.

2.11.1 National Instruments VISA

Το VISA (Virtual Software Architecture) είναι ένα πρότυπο για τη διαμόρφωση, τον προγραμματισμό, και την επίλυση προβλημάτων οργανωμένων συστημάτων η οποία περιλαμβάνει διεπαφές GPIB, VXI, PXI, Serial, Ethernet, and/or USB. Το VISA παρέχει τον προγραμματιστικό περιβάλλον μεταξύ υλικού και αναπτυξιακού περιβάλλοντος όπως το LabVIEW στη δικιά μας περίπτωση, το LabWindows/CVI και το Measurement Studio για Microsoft Visual Studio. Το NI-VISA είναι μία εφαρμογή της National Instruments του προτύπου VISA I/O. Περιλαμβάνει βιβλιοθήκες λογισμικού, διαδραστικές υπηρεσίες όπως NI I/O Trace και τη VISA Interactive Control, και τα προγράμματα διαμόρφωσης μετρήσεων και Αυτοματισμών για όλες τις αναπτυξιακές ανάγκες. Το NI-VISA είναι πρότυπο σε όλη τη σειρά προϊόντων της National Instruments.

2.11.2 Προγραμματισμός σε LabVIEW

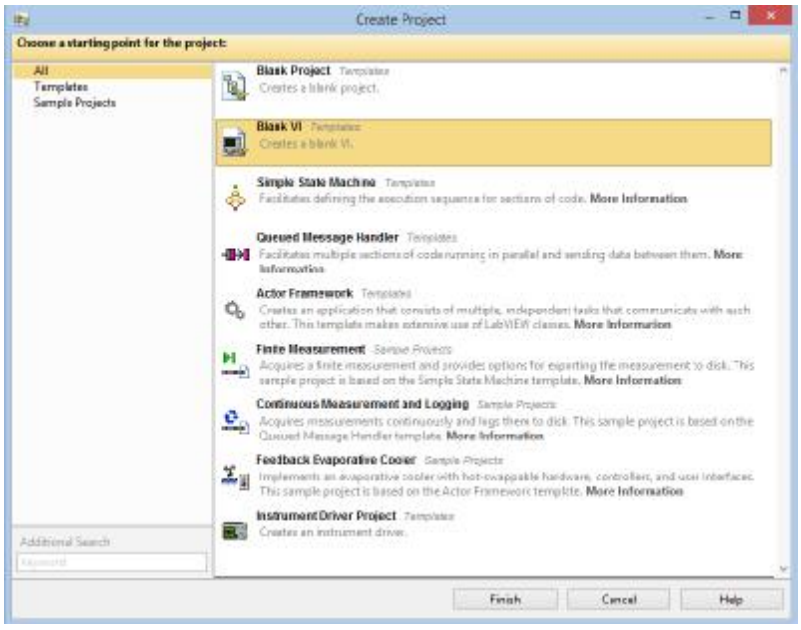
Ένα πρόγραμμα στο LabVIEW αποτελείται από δύο βασικά μέρη. Αυτά είναι το block diagram και το front panel. Το πρώτο είναι αυτό στο οποίο γράφουμε τον κώδικά μας με τη μορφή blocks και διάφορων εικονιδίων και συμβόλων συνδεδεμένων μεταξύ τους. Τ δεύτερο, αποτελεί το αποτέλεσμα του κώδικα του block diagram στο οποίο ο χρήστης μπορεί να δει γραφικά, μετρήσεις, κυματομορφές κλπ ανάλογα με τον κώδικά του.

Ας δούμε τα βασικότερα σημεία του προγραμματισμού σε LabVIEW με βάση το project μας.

Κατά την εκκίνηση του LabVIEW μας ανοίγει ένα menu (Εικόνα 2.1) στο οποίο καλούμαστε να φτιάξουμε ένα νέο project ή να ανοίξουμε ένα ήδη υπάρχον:

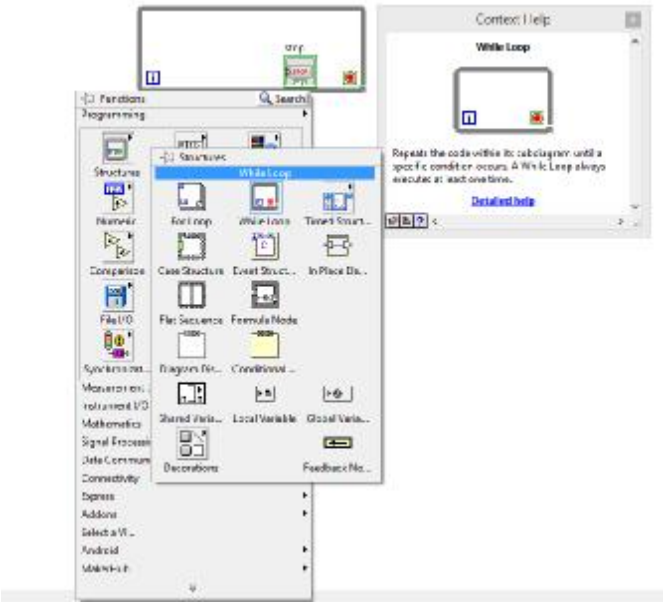
Εικόνα 2.1-Αρχική οθόνη LabVIEW

Στην περίπτωση που έχουμε ήδη κάποιο έτοιμο project επιλέγουμε Open Existing και διαλέγουμε το project μας. Σε διαφορετική περίπτωση επιλέγουμε Create Project και στο επόμενο παράθυρο (Εικόνα 2.2) επιλέγουμε να δημιουργήσει Blank VI. Το αποτέλεσμα θα είναι να μας εμφανιστούν τα δύο μέρη του προγράμματος που αναφέραμε παραπάνω.



Εικόνα 2.2- Δημιουργία νέου Project

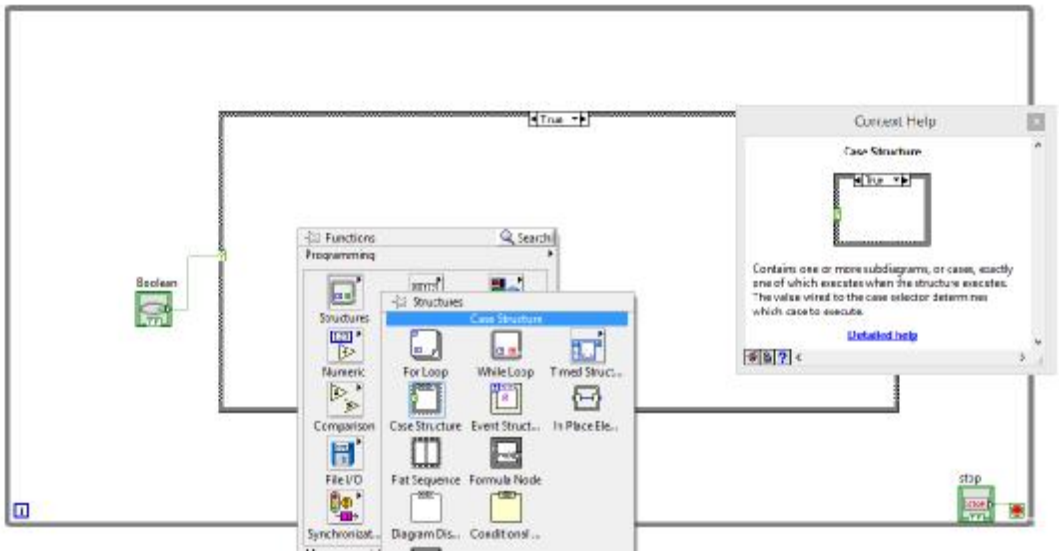
Κάθε πρόγραμμα το LabVIEW συνήθως εκτελείται μέσα σε ένα loop (Εικόνα 2.3). Το loop αυτό είναι μια while loop ή οποία περιέχει όλο τον κώδικα ο οποίος εκτελείται από την αρχή μέχρι το τέλος για τουλάχιστον μία φορά. Η σε διαφορετική περίπτωση θα σταματήσει μόνο εάν υπάρξει κάποια συνθήκη που θα έχουμε ορίσει εμείς αλλιώς με το button STOP που μπορούμε εύκολα να δημιουργήσουμε.



Εικόνα 2.3-Εισαγωγή While Loop

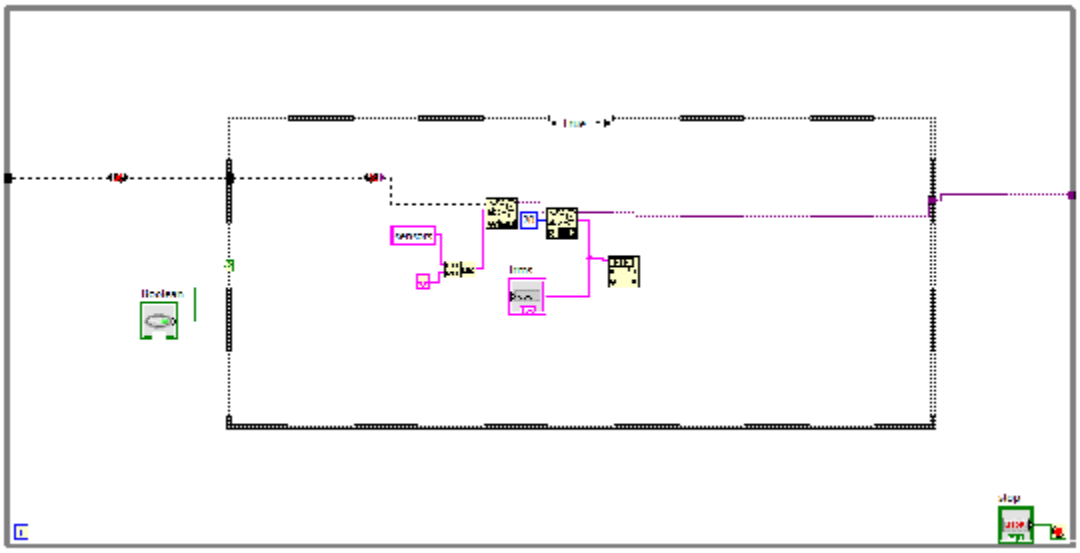
Για το STOP επιλεγούμε create indicator στο loop condition (κόκκινη τελεία).

Έστω τώρα πως θέλουμε να διαβάζουμε την τιμή ενός αισθητήρα από το Arduino και να την εμφανίζουμε στο LabVIEW. Για να το κάνουμε αυτό θα χρειαστούμε μια συνθήκη true/false (Εικόνα 2.4) που θα ενεργοποιείται με ένα button. Στην ουσία θα ρωτάει το LabVIEW το Arduino για αυτή την τιμή κάθε φορά που θα πατάμε το button και η συνθήκη γίνεται true.



Εικόνα 2.4-Εισαγωγή συνθήκης True/False

Πραγματοποιώντας το παρακάτω κύκλωμα (Εικόνα 2.5) θα μπορούσαμε να ζητήσουμε από το Arduino μια τιμή όπως για παράδειγμα την τιμή του ρεύματος:



Εικόνα 2.5-Έλεγχος τιμής αισθητήρα

Για να γίνει βέβαια αυτό λείπουν ακόμα αρκετά πράγματα από τον κώδικα και χρειάζεται και προγραμματισμός και στο Arduino. Αρχικά ας εξηγήσουμε τα νέα σύμβολα και τη λογική του κώδικα. Όπως είπαμε η while εκτελείται συνέχεια. Εάν τώρα δεν έχουμε πατήσει το button Boolean δεν θα μπορούμε να πάρουμε κάποια απάντηση από το Arduino αφού η συνθήκη είναι false. Εφόσον πατηθεί το button, η συνθήκη γίνεται True και εκτελείται ο κώδικας που περιλαμβάνει.

Στην παραπάνω Εικόνα βλέπουμε κάποια καινούρια σύμβολα.

 Αυτό το σύμβολο(Concatenate Strings) είναι ένα μέσο που δέχεται σαν είσοδο μεταβλητές string και της βγάζει σε μία έξοδο. Βλέπουμε λοιπόν πως έχουμε βάλει σαν είσοδο το string sensors και ένα ακόμα σύμβολο  το οποίο μια σταθερά string ASCII χαρακτήρα για αλλαγή γραμμής.

 Το VISA Write χρησιμοποιείται για να γράψει δεδομένα στη συσκευή που είναι συνδεδεμένη στο VISA, στην περίπτωση μας στο Arduino. Άρα χρησιμοποιώντας το όπως βλέπουμε στον κώδικα μπορούμε να γράψουμε στη σειριακή θήρα του Arduino τη λέξη sensors.

 Το VISA Read χρησιμοποιείται για να διαβάσει τα byte δεδομένων που στέλνει η συνδεδεμένη συσκευή. Μπορούμε να ορίσουμε πόσα byte δεδομένων περιμένουμε όπως πχ 20byte επιλέγοντας στο byte count την επιλογή create constant.

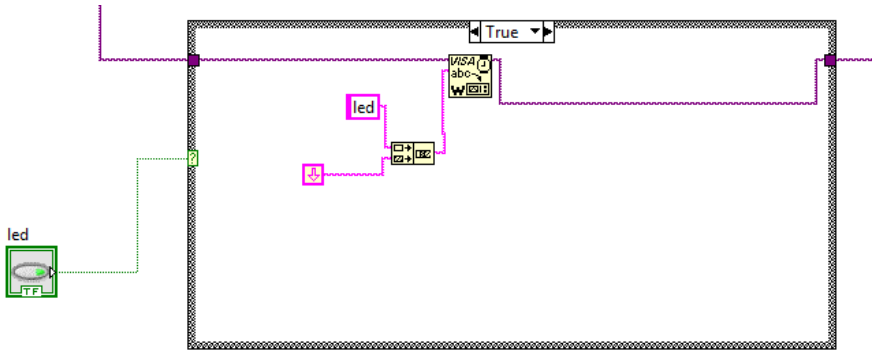
Σαν έξοδο τώρα του VISA Read μπορούμε να βάλουμε κάποιο γραφικό ή απλά ένα indicator.

 Το σύμβολο αυτό(Decimal String to Number)αποτελεί μετατροπέα από string σε number έτσι ώστε να μπορέσουμε να απεικονίσουμε τις τιμές που μετράμε σε

κάποιο γραφικό σε διαφορετική περίπτωση με έναν απλό indicator  μπορούμε να βλέπουμε στο Front Panel τις μετρήσεις μας.

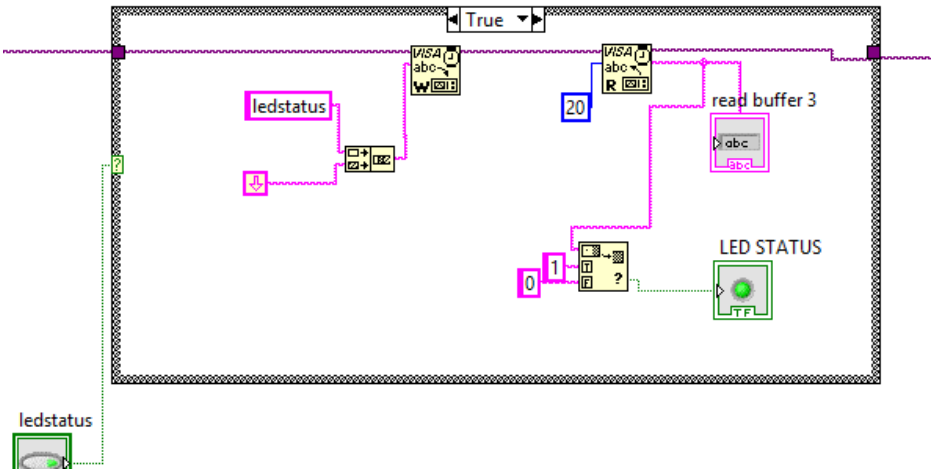
2.11.3 Έλεγχος LED με LabVIEW

Έστω τώρα πως θέλουμε να ελέγξουμε ένα LED. Η λογική που ακολουθήσαμε μέχρι τώρα είναι σχεδόν ίδια, δηλαδή θα χρειαστούμε πάλι συνθήκη μέσα σε while, εντολή VISA write για να γράψουμε στο serial του Arduino την εντολή μας και VISA Read εάν θέλουμε να γνωρίζουμε την κατάσταση του LED.



Εικόνα 2.6-Έλεγχος Led

Ο παραπάνω κώδικας (Εικόνα 2.2) μπορεί να χρησιμοποιηθεί για να ανοίξει ή να σβήσει ένα led. Τα σύμβολα και οι εντολές είναι πλέον γνωστές. Τι γίνεται όμως με τον έλεγχο της κατάστασης του LED; Ο κώδικας φαίνεται στην επόμενη εικόνα(Εικόνα 2.3).



Εικόνα 2.7-Έλεγχος κατάστασης Led

Τα νέα σύμβολα-εντολές είναι το Match True/False String  Εξετάζει την αρχή του string για να δείτε αν ταιριάζει true string ή false string. Αυτή η συνάρτηση επιστρέφει μια Boolean τιμή true ή false, ανάλογα με το αν το string ταιριάζει με true string ή false string . Με λίγα λόγια στην περίπτωσή μας εξετάζει την τιμή του LED.

Εάν είναι ανοιχτό το Arduino θα στείλει string 1 το οποίο θα συγκριθεί με το true το οποίο επίσης 1 άρα στην έξοδο θα έχουμε 1 και αντίστροφα.

Το δεύτερο σύμβολο είναι το LED το οποίο παίρνει σαν είσοδο την έξοδο της προηγούμενης συνθήκης και ανάβει ή σβήνει αντίστοιχα.

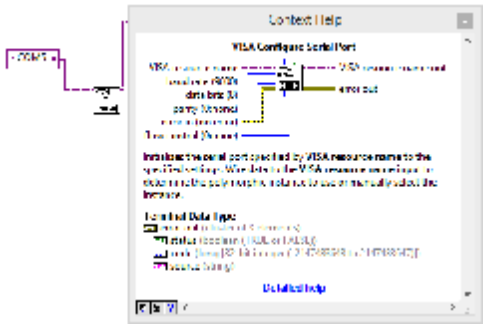
Παρατήρηση

Σχεδόν πάντα εφαρμόζουμε μια καθυστέρηση με την εντολή wait  *. Χρησιμοποιείται για να εξαλείψει την ανάγκη της συνεχούς αίτησης μετάδοσης στο front panel σε περίπτωση που κάποια τιμή αλλάξει λόγω του διαφορετικού χρονισμού στα ρολόγια των επεξεργαστών του υπολογιστή μας και του Arduino.*

2.11.4 Σύνδεση LabVIEW με Arduino

Επιπλέον και πολύ σημαντικό, πρέπει να συνδέσουμε τον κώδικα του LabVIEW με το Arduino. Ας δούμε πρώτα τι χρειάζεται από πλευράς LabVIEW. Αρχικά θα πρέπει να δηλώσουμε τη θύρα USB ίδια με αυτή που έχουμε δηλώσει και στο Arduino IDE.

Για να το κάνουμε αυτό εισάγουμε την εντολή VISA Configure Serial Port (Εικόνα 2.8).



Εικόνα 2.8-VISA Configuration Port

Με την εντολή αυτή μπορούμε να ορίσουμε τη θήρα USB που έχουμε συνδέσει το Arduino, στη δικιά μας περίπτωση COM 5, καθώς και το bound rate 9600 συνήθως όπως και στο Arduino. Οι εντολές αυτές μπαίνουν έξω από το while loop.

Τέλος, μένει να γράψουμε και την εντολή VISA close  η οποία κλείνει τη σύνδεση με τη συνδεδεμένη συσκευή.

2.11.5 Απαιτήσεις Arduino για σύνδεση και έλεγχο από LabVIEW

Όπως είπαμε και παραπάνω, η λογική που χρησιμοποιούμε για να επικοινωνήσει το LabVIEW με το Arduino είναι το πρώτο να "ρωτάει" στέλνοντας ένα string και το δεύτερο να "απαντάει" στέλνοντας τα κατάλληλα δεδομένα. Για να γίνει αυτό θα πρέπει το Arduino να δέχεται με κάποιο τρόπο τα string που του στέλνει το LabVIEW. Για το λόγο αυτό θα ορίσουμε μια μεταβλητή string που θα την ονομάσουμε buffer και θα ελέγχουμε για string χαρακτήρες στη σειριακή θύρα. Στη συνέχεια εφόσον το Arduino λάβει κάποιο string τότε θα βάλουμε συνθήκες if...else

και ανάλογα με τις τιμές του buffer θα κάνουμε και τις αντίστοιχες επιθυμητές ενέργειες.

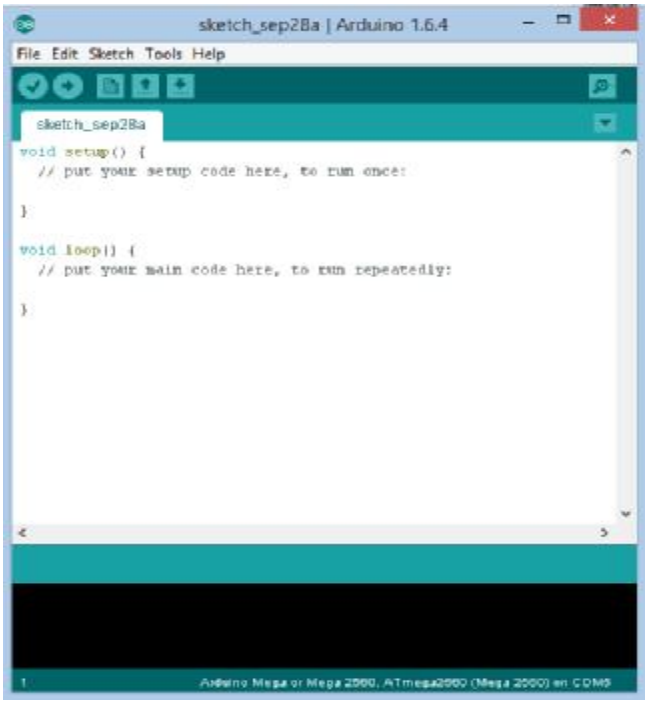
Έτσι αρχικά ορίζουμε μία μεταβλητή τύπου χαρακτήρα g στην οποία θα σώζουμε τα δεδομένα του serial. Άρα πάνω από το void setup() γράφουμε

```
char g;
String buffer;
```

Στη συνέχεια επειδή θέλουμε να ελέγχουμε τι υπάρχει στη σειριακή θήρα ορίζουμε κάποιες συνθήκες οι οποίες είναι: εάν υπάρχει σειριακή επικοινωνία και για όσο υπάρχει να καταγράφονται οι string χαρακτήρες του serial στον g και να σώζονται στον buffer με τον οποίο θα κάνουμε ελέγχους με if.

```
if (Serial.available() > 0)
  {
    delay(100);
    while (Serial.available() > 0)
    {

      g = Serial.read();
      buffer += g;
    }
    if (buffer == "sensors\n") // Ean o buffer exei tin timi sensors, to Arduino
    {                          // tha tipwsei tin metabliti t tin opoia tha diabasei
      Serial.println(t);        // to LabVIEW me to VISA Read
      buffer = "";
    }
    else if (buffer == "led\n") //Ean o buffer einai "led" tha allaxei i katastasi tou
    {                               // led
      // Serial.println("1");
      ledstate = digitalRead(led);       // kathe fora pou o buffer einai led kaleitai
      ledstate = change_status(ledstate); //i iporoutina change status i opoia
      digitalWrite(led, ledstate);        // antistrefei tin teleutaia katastasi tou
      buffer = "";                    // led
    }
    else if (buffer == "ledstatus\n") //Ean o Buffer einai ledstatus
    {                                 // tha tipwthei i katastasi tou led
      ledstate = digitalRead(led);    // ws 0 h 1
      Serial.println(ledstate);
      buffer = "";
    }

    buffer = "";  //
  }
int change_status(int i)
{
  int result;
  result = !i;
  return result;
}
```

2.12 Λογισμικό προγραμματισμού ARDUINO IDE 1.6.4



Εικόνα 2.9-Arduino IDE

Πρόκειται για ένα πρόγραμμα ανοιχτού κώδικα στο οποίο μπορούμε να γράψουμε κώδικα και να τον ανεβάσουμε στην πλακέτα μας. Τρέχει σε Windows, Mac OS X, και Linux. Το περιβάλλον είναι γραμμένο σε java και βασίζεται σε Processing όπου είναι ένα άλλο πρόγραμμα ανοιχτού κώδικα και στο και το σχέδιο Wiring. Μπορεί να χρησιμοποιηθεί με κάθε πακέτα Arduino.

2.12.1 Ελάχιστα συστατικά του προγράμματος και αρχικές ρυθμίσεις

Κάθε εφαρμογή στο Arduino απαιτεί ένα κύκλωμα και ένα κώδικα. Ο κώδικας(πρόγραμμα) εκτελείται από τον μικροελεγκτή και το αποτέλεσμα εμφανίζεται στο κύκλωμα. Η δυνατότητα ελέγχου εφαρμογών μέσω προγραμματισμού προσφέρει μεγάλη ευκολία για το σχεδιαστή. Σε αντίθετη περίπτωση θα έπρεπε να αναπτυχθεί ένα πολύπλοκο ηλεκτρονικό κύκλωμα εφόσον δεν υπάρχει στο Arduino.

ο κώδικας που αναπτύσσεται στο Arduino βασίζεται στη γλώσσα προγραμματισμού Wiring C η οποία υποστηρίζει τους κλασικούς κανόνες δομημένου προγραμματισμού

κάθε πρόγραμμα αποτελείται από δύο τουλάχιστον συναρτήσεις(ή διαδικασίες στην πραγματικότητα εφόσον δεν επιστρέφουμε κάποιο αποτέλεσμα).

η πρώτη συνάρτηση ονομάζεται **setup** και ο κώδικας που περιλαμβάνει εκτελείται μόνο μία φορά κατά την εφαρμογή της τροφοδοσίας στο Arduino. Συνήθως χρησιμοποιείται για αρχικοποίηση.

Η δεύτερη συνάρτηση ονομάζεται **loop** και εκτελείται συνέχεια σαν βρόγχος όσο υπάρχει τροφοδοσία στο Arduino.

Ο κώδικας που ακολουθεί δείχνει τα ελάχιστα συστατικά ενός τέτοιου προγράμματος.

```
void setup()
```

```
{
//εντολές που εκτελούνται μόνο μία φορά
}
void loop()
{
//εντολές που εκτελούνται συνέχεια
}
```

Όπου υπάρχουν τα σύμβολα (//) πρόκειται για σχόλια.

2.12.2 Προγραμματισμός εξόδου για έλεγχο LED

Όπως φαίνεται και στην πλακέτα του Arduino, υπάρχουν αριθμημένοι ακροδέκτες οι οποίοι χρησιμοποιούνται για την ανταλλαγή σημάτων με εξωτερικές συσκευές και κυκλώματα.

Πιο συγκεκριμένα, οι ακροδέκτες 0 έως 13 υποστηρίζουν την ανταλλαγή ψηφιακών σημάτων στάθμης 0V ή 5 V. Εμείς διαλέξαμε στην τύχη τον ακροδέκτη 5. Σύμφωνα με τις στάθμες σήματος που υποστηρίζονται, θα έχουμε αναμμένο το LED δίνοντας 5V στην έξοδο στην οποία είναι συνδεδεμένο και σβηστό θέτοντας την έξοδο στα 0V.

Ο έλεγχος του LED προϋποθέτει δύο βήματα: (α) Καθορισμός του ακροδέκτη 5 ως εξόδου και (β) ρύθμιση της στάθμης σήματος (π.χ. 5V→0V→ 5V, κλπ ώστε να ανάβει και να σβήνει).

Ο καθορισμός του ακροδέκτη 5 μπορεί να γίνει εντός της συνάρτησης **setup**, ενώ η ρύθμιση της στάθμης για το LED εντός της **loop**. Έτσι, εξασφαλίζουμε τον συνεχή έλεγχο του LED όσο το Arduino είναι συνδεδεμένο στην τροφοδοσία.

Ο καθορισμός συγκεκριμένου ακροδέκτη για είσοδο ή έξοδο γίνεται με την εντολή `PinMode (αριθμός ακροδέκτη, τύπος);`

Για τους ψηφιακούς ακροδέκτες, ο αριθμός ακροδέκτη μπορεί να βρίσκεται στο διάστημα 0 έως 13 (σε περίπτωση Arduino MEGA, κλπ, έχουμε περισσότερες επιλογές αφού οι ακροδέκτες είναι περισσότεροι).

Οι βασικές τιμές της παραμέτρου τύπος μπορεί να είναι INPUT, OUTPUT (σε ειδικές περιπτώσεις υπάρχει και ο τύπος INPUT PULLUP). Στη δική μας περίπτωση ο τύπος θα είναι OUTPUT.

Έτσι, για να καθοριστεί ο ψηφιακός ακροδέκτης 5 ως εξόδου θα δώσουμε την εντολή `pinMode (5, OUTPUT) ;`

Το κυρίως πρόγραμμα (εντός της συνάρτησης **loop**) θα περιλαμβάνει εντολές που καθορίζουν τη στάθμη του σήματος στον ακροδέκτη 5. Η εντολή που καθορίζει τη στάθμη σήματος σε ένα ψηφιακό ακροδέκτη εξόδου ονομάζεται digitalWrite και συντάσσεται ως εξής:

```
digitalWrite (ακροδέκτης,στάθμη);
```

Η στάθμη μπορεί να παίρνει τις τιμές **HIGH** (5 V) ή **LOW** (0 V). Στην πραγματικότητα βέβαια η στάθμη LOW δεν αντιστοιχεί ακριβώς στα 0V. Η χρήση της εντολής digitalWrite για να ανάβουμε και να σβήνουμε το LED θα έχει την ακόλουθη μορφή:

```
digitalWrite (5, HIGH);
```

για να ανάψει το LED, και

```
digitalWrite (5, LOW);
```

για να σβήσει το LED.

Αν τοποθετηθούν οι δύο αυτές εντολές η μία δίπλα στην άλλη, δεν θα προλαβαίνουμε να δούμε το led να ανάβει και να σβήνει. Αυτό σημαίνει πως θα πρέπει να υπάρχει μια χρονική καθυστέρηση πριν το LED ανάψει ή σβήσει. Αυτό επιτυγχάνεται με τη χρήση της εντολής **delay** η οποία δημιουργεί χρονική καθυστέρηση. Η εντολή συντάσσεται

```
delay(πλήθος  ms);
```

για παράδειγμα, η χρονική καθυστέρηση ενός δευτερολέπτου επιτυγχάνεται αν για παράμετρο βάλουμε τον αριθμό 1000(100ms=1sec).

Έτσι ο κώδικας μπορεί να εμπλουτιστεί ως εξής:

```
digitalWrite (5, HIGH) ;
delay(1000);
digitalWrite(5,LOW);
```

Με το δεδομένο ότι αυτός ο κώδικας μπαίνει στη συνάρτηση **loop**, το αποτέλεσμα είναι το LED να ανάβει και να σβήνει για όσο υπάρχει τροφοδοσία στο Arduino.

Ολοκληρωμένο το πρόγραμμα έχει ως εξής:

```
void setup()
{
   pinMode (5, OUTPUT) ;
}
void loop()
{
   digitalWrite (5, HIGH) ;
   delay(1000);
   digitalWrite(5,LOW);
}
```

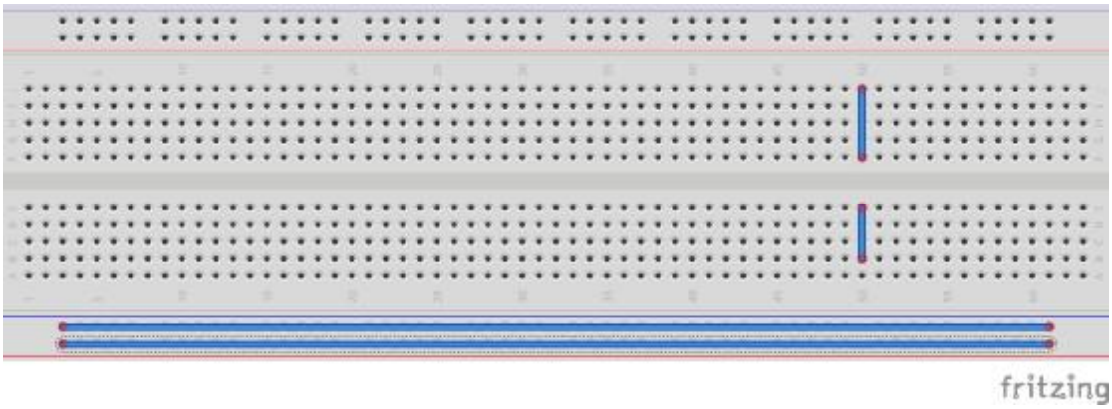
2.12.3 Βήματα υλοποίησης ελέγχου LED

Τα βήματα υλοποίησης για την εφαρμογή ελέγχου του LED αλλά και όλων των εφαρμογών είναι τα ακόλουθα:

Βήμα 1. Υλοποίηση κυκλώματος(χωρίς σύνδεση εξωτερικής τροφοδοσίας ή USB)

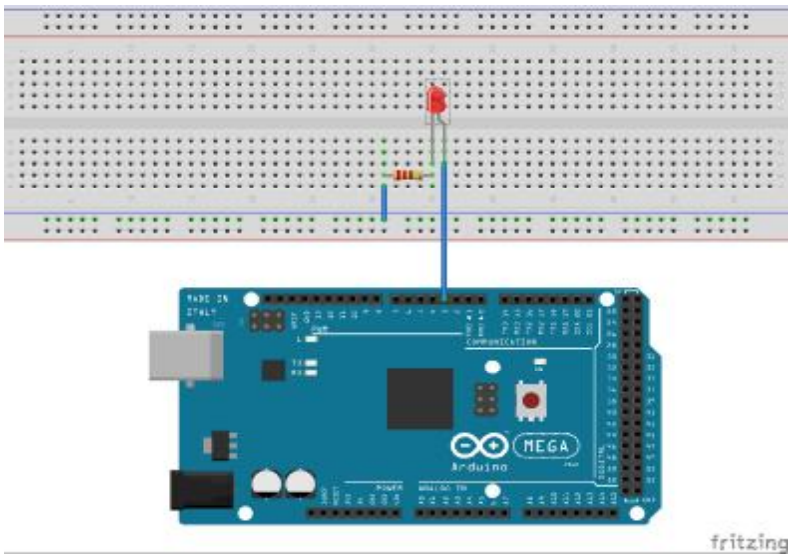
Προφανώς απαιτούνται καλώδια για τις συνδέσεις του κυκλώματος. Οι συνδέσεις των καλωδίων με το χέρι ή με το κολλητήρι απαιτούν χρόνο και δεξιότητα με βασικό μειονέκτημα ότι δύσκολα τροποποιούνται.

Για αυτό το λόγο χρησιμοποιούμε το γνωστό breadboard το οποίο διαθέτει στο εσωτερικό του ένα τυποποιημένο κύκλωμα και στην επιφάνειά του έχει υποδοχές για καλώδια και εξαρτήματα.



Εικόνα 2.10-Εσωτερική συνδεσμολογία Breadboard

όπως φαίνεται στην παραπάνω εικόνα (Εικόνα 2.10) υπάρχουν δύο τύποι εσωτερικών συνδέσεων. Οι υποδοχείς στις κεντρικές λωρίδες είναι συνδεδεμένες κατακόρυφα ενώ στις άκρες οριζόντια. Οι υποδοχές στα άκρα χρησιμοποιούνται για τροφοδοσία(+/-) ώστε να είναι προσβάσιμη από όλα τα σημεία του breadboard. Στην επόμενη εικόνα (Εικόνα 2.11) φαίνεται η τελική υλοποίηση.

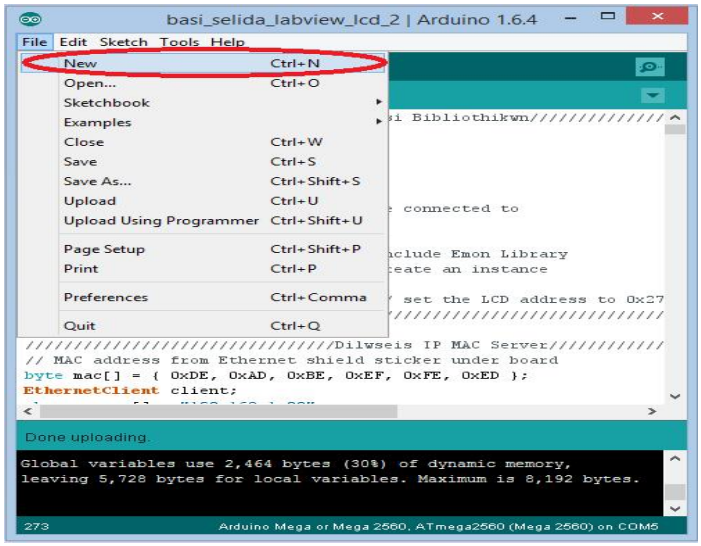


Εικόνα 2.11-Συνδεσμολογία ελέγχου Led

Βήμα 2: ***Εισαγωγή κώδικα στο περιβάλλον Arduino IDE***

Πληκτρολογούμε τον κώδικα στο περιβάλλον Arduino IDE το οποίο έχουμε κατεβάσει δωρεάν από τη διεύθυνση http://arduino.cc

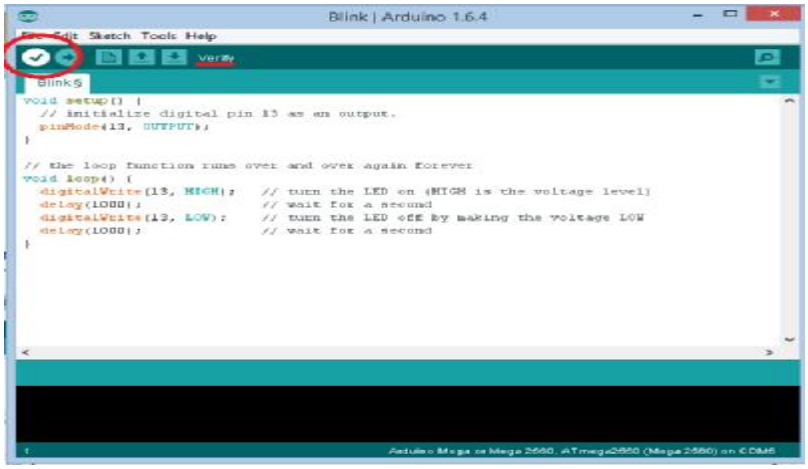
Για να ανοίξουμε ένα κενό πρόγραμμα και να ξεκινήσουμε να γράφουμε ένα καινούριο κώδικα επιλέγουμε File→New όπως βλέπουμε στην επόμενη εικόνα (Εικόνα 2.12)



Εικόνα 2.12-Νέο πρόγραμμα

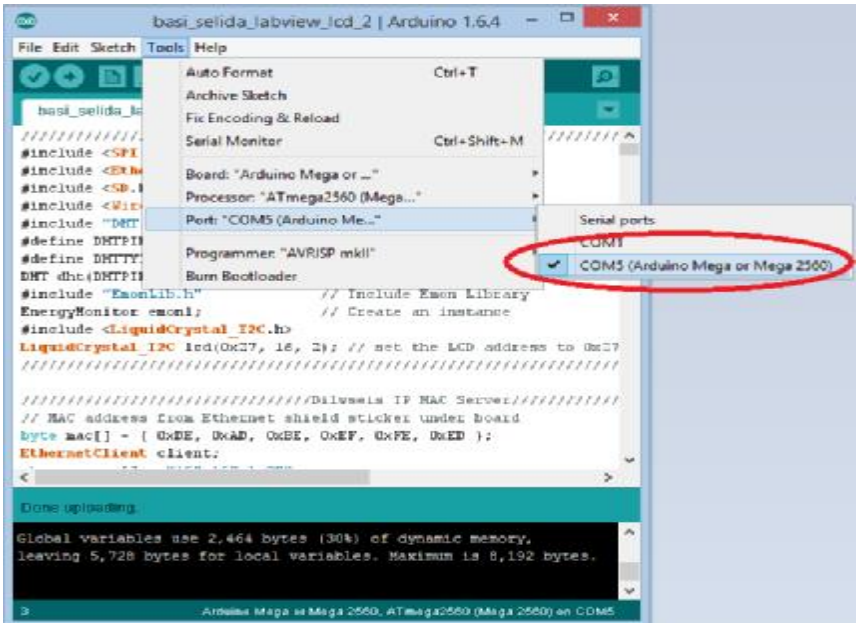
Βήμα 3: **Μεταφορά (μεταφόρτωση) κώδικα**

Πριν κάνουμε upload ένα κώδικα, πρέπει πάντα να κάνουμε έλεγχο για συντακτικά λάθη. Ο έλεγχος αυτός γίνεται με το κουμπί Verify το οποίο βλέπουμε στην Εικόνα 2.13. Εάν στο μαύρο πλαίσιο δεν μας βγάλει κάποιο σφάλμα μπορούμε να συνεχίσουμε και να κάνουμε upload του κώδικά μας.

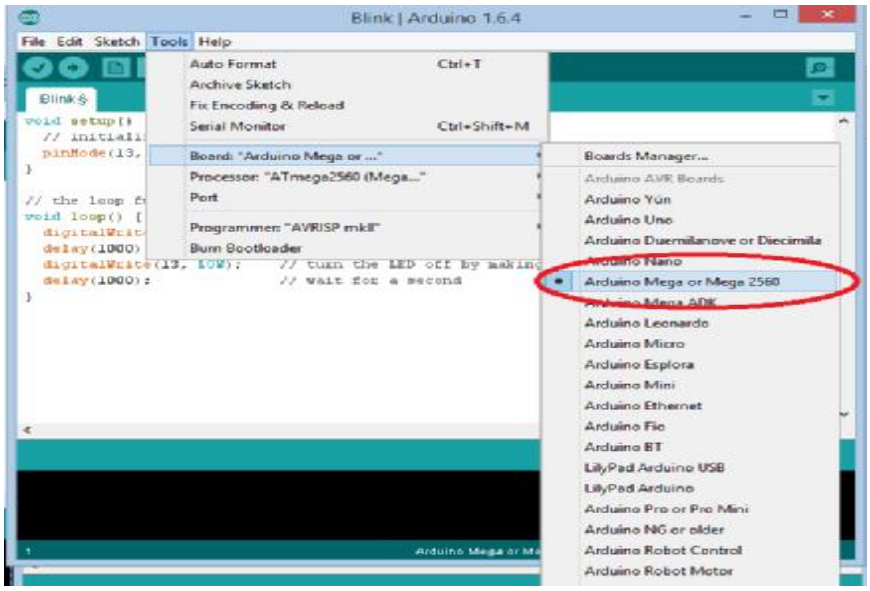


Εικόνα 2.13-Έλεγχος κώδικα

Η μεταφορά του κώδικα στον μικροελεγκτή Arduino περιλαμβάνει αυτόματα τη μεταγλώττιση του. Πριν τη μεταφορά του θα πρέπει να συνδέσουμε σε μια θύρα USB το Arduino και να ρυθμίσουμε και την ίδια τη θύρα στο περιβάλλον IDE (Εικόνα 2.14). Επίσης, θα πρέπει να δηλώσουμε και την έκδοση του Arduino board που χρησιμοποιούμε (Εικόνα 2.15).

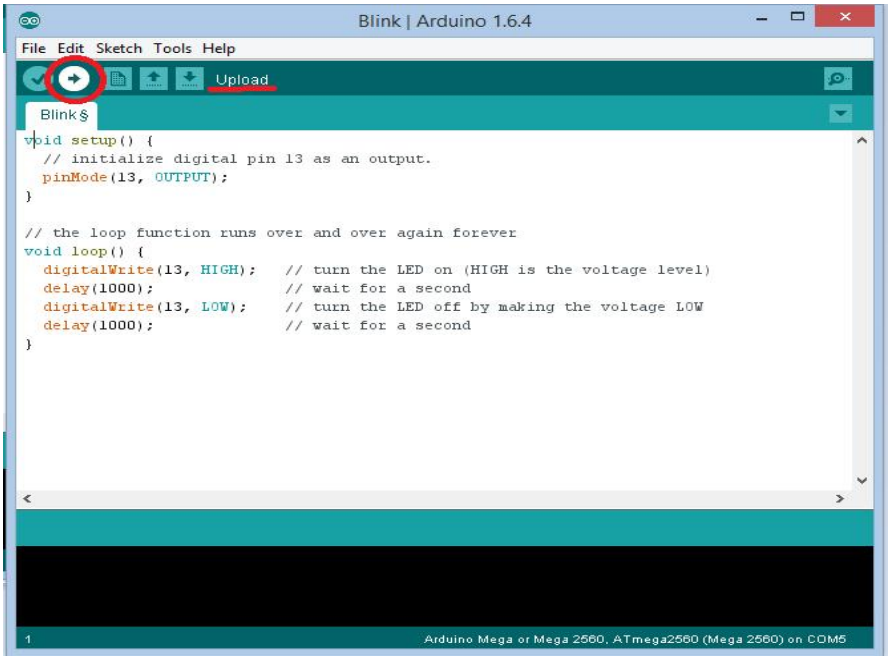


Εικόνα 2.14-Επιλογή Port



Εικόνα 2.15-Επιλογή πλακέτας

Αφού γίνουν οι κατάλληλες ρυθμίσεις και το περιβάλλον IDE αναγνωρίσει το Arduino board και τη θύρα USB μεταφέρουμε τον κώδικα στον μικροελεγκτή επιλέγοντας File→ Upload ή απλά το κουμπί upload (Εικόνα 2.16) και παρατηρούμε το αποτέλεσμα.



Εικόνα 2.16-Upload κώδικα στο Arduino

Επειδή πρόκειται να υλοποιήσουμε πολλές εφαρμογές, υπάρχει η πιθανότητα να βρίσκεται ήδη φορτωμένο ένα άλλο πρόγραμμα. Αυτό μπορεί να προκαλέσει προβλήματα με τη νέα εφαρμογή. Για αυτό το λόγο προτείνεται να κάνουμε μεταφόρτωση ενός κενού προγράμματος (του ελάχιστου κώδικα που αναφέραμε) πριν συνδέσουμε το κύκλωμα.

2.12.5 Αναφορά στη γλώσσα του Arduino

Τα προγράμματα του Arduino μπορούν να χωριστούν σε τρία κύρια μέρη: τη δομή (Structure), τις μεταβλητές (Variables), και τις συναρτήσεις (Functions). Παρακάτω θα γίνει αναφορά στις πιο βασικές εντολές και αυτές που χρησιμοποιήσαμε στο project μας.

Δομή

setup()	Καλείται στην αρχή του προγράμματος και χρησιμοποιείται για να αρχικοποιήσουμε μεταβλητές, για καταστάσεις pin, για να αρχίσει η χρήση βιβλιοθηκών κλπ. Τρέχει μόνο μία φορά όταν τροφοδοτήσουμε την πλακέτα ή όταν κάνουμε restart.	`int buttonPin = 3;`
		`void setup()`
		`{`
		`Serial.begin(9600);`
		`pinMode(buttonPin, INPUT);`
		`}`
		`void loop()`
		`{`
		`  // ...`
		`}`

loop()	Μετά το setup, τρέχει συνεχώς επιτρέποντας το πρόγραμμα να κάνει αλλαγές και να δίνει απαντήσεις.	`const int buttonPin = 3;`
		`void setup()`
		`{`
		`  Serial.begin(9600);`
		`  pinMode(buttonPin, INPUT);`
		`}`
		`void loop()`
		`{`
		`  if (digitalRead(buttonPin) == HIGH)`
		`  Serial.write('H');`
		`  else`
		`  Serial.write('L');`
		`  delay(1000);`
		`  }`

Δομή ελέγχου

if	Χρησιμοποιείται σε συνδυασμό με ένα τελεστή σύγκρισης, δοκιμάζει αν μια συγκεκριμένη κατάσταση έχει επιτευχθεί, όπως αν μία είσοδος είναι πάνω από ένα ορισμένο αριθμό	`if (someVariable > 50)`
		`{`
		`  // do something here`
		`}`

if...else	Επιτρέπει μεγαλύτερο έλεγχο της ροής του κώδικα από τη βασική εντολή if, επιτρέποντας πολλαπλές δοκιμές να ομαδοποιηθούν. Για παράδειγμα, μια αναλογική είσοδος θα μπορούσε να δοκιμαστεί και μια δράση να γίνει εάν η είσοδος ήταν μικρότερη από 500, και μια άλλη δράση να γίνει εάν η είσοδος ήταν 500 ή μεγαλύτερη.	`if (pinFiveInput < 500)`
		`{`
		`  // action A`
		`}`
		`else`
		`{`
		`  // action B`
		`}`

for	Η δήλωση for χρησιμοποιείται για να επαναλάβει ένα μπλοκ δηλώσεων που περικλείονται σε αγκύλες. Ένας μετρητής προσαύξησης συνήθως χρησιμοποιείται για να αυξάνεται και να τερματίσει την επανάληψη.	`// Dim an LED using a PWM pin`
		`int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10`
		
		`void setup()`
		`{`
		`  // no setup needed`
		`}`
		
		`void loop()`
		`{`
		`  for (int i=0; i <= 255; i++){`
		`  analogWrite(PWMpin, i);`
		`  delay(10);`
		`  }`
		`}`

switch case	Όπως και στην περίπτωση του if,η switch...case ελέγχει τη ροή των προγραμμάτων, επιτρέποντας στους προγραμματιστές να καθορίσουν διαφορετικό κώδικα που πρέπει να εκτελεστεί σε διάφορες συνθήκες. Ειδικότερα, μια δήλωση switch συγκρίνει την τιμή μιας μεταβλητής με τις τιμές που ορίζονται στις δηλώσεις case. Όταν για μια δήλωση case βρεθεί η τιμή των οποίων ταιριάζει με αυτή της μεταβλητής, ο κώδικας στην εν λόγω δήλωση περίπτωση τρέχει.
	
	Με το break βγαίνουμε από την case, και χρησιμοποιείται συνήθως στο τέλος της κάθε case. Χωρίς μια δήλωση break, η δήλωση switch θα συνεχίσει να εκτελεί τις ακόλουθες εκφράσεις μέχρι ένα break, ή το τέλος της δήλωσης switch να επιτευχθεί.
		`switch (var) {`
		`  case 1:`
		`  //do something when var equals 1`
		`  break;`
		`  case 2:`
		`  //do something when var equals 2`
		`  break;`
		`  default:`
		`  // if nothing else matches, do the default`
		`  // default is optional`
		`  break;`
		`  }`

while	Οι βρόχοι while εκτελούνται συνεχώς, και απείρως, μέχρι η έκφραση μέσα στην παρένθεση (), να γίνεται ψευδής. Κάτι πρέπει να αλλάξει τη δοκιμαζόμενη μεταβλητή όπως ένας μετρητής ή ένας αισθητήρας, αλλιώς ο βρόχο while δεν θα τελειώσει ποτέ.	`var = 0;`
		`while(var < 200){`
		`  // do something repetitive 200 times`
		`  var++;`
		`}`

do... while	Το do loop λειτουργεί με τον ίδιο τρόπο όπως το βρόχο, while, με την διαφορά ότι η κατάσταση ελέγχεται κατά το τέλος του βρόχου, έτσι ώστε ο βρόχος do θα τρέχει πάντα τουλάχιστον μία φορά.	`do`
		`{`
		`  delay(50);  // wait for sensors to stabilize`
		`  x = readSensors();  // check the sensors`
		
		`} while (x < 100);`

break	Χρησιμοποιείται για την έξοδο από ένα do, for, ή while, παρακάμπτοντας την κανονική κατάσταση βρόχου. Είναι, επίσης,	`for (x = 0; x < 255; x ++)`
		`{`
		`  analogWrite(PWMpin, x);`
		`  sens =`

	χρησιμοποιείται για την έξοδο από μια δήλωση switch.	`analogRead(sensorPin);`
		`  if (sens > threshold){`
		`// bail out on sensor detect`
		`  x = 0;`
		`  break;`
		`  }`
		`  delay(50);}`

continue	Η εντολή continue παρακάμπτει το υπόλοιπο της τρέχουσας επανάληψη ενός βρόχου (do, for, ή while). Συνεχίζει με τον έλεγχο της έκφρασης συνθήκης του βρόχου, και προχωρά με τυχόν μεταγενέστερες επαναλήψεις.	`for (x = 0; x < 255; x ++)`
		`{`
		`  if (x > 40 && x < 120){`
		`// create jump in values`
		`  continue;`
		`  }`
		
		`  analogWrite(PWMpin, x);`
		`  delay(50);`
		`}`

return	Τερματίζει μια λειτουργία και επιστρέφει μια τιμή στη συνάρτηση κλήσης, εάν είναι επιθυμητό.	`int checkSensor(){`
		`  if (analogRead(0) > 400) {`
		`  return 1;`
		`  else{`
		`  return 0;`
		`  }`
		`}`

goto	Μεταφέρει τη ροή του προγράμματος σε ένα επισημασμένο σημείο στο πρόγραμμα	`for(byte r = 0; r < 255; r++){`
		`  for(byte g = 255; g > -1; g--){`
		`  for(byte b = 0; b < 255; b++){`
		`  if (analogRead(0) > 250){ goto bailout;}`
		`  // more statements ...`
		`  }`
		`  }`
		`}`
		`bailout:`

δυαδικοί τελεστές

&	Δυαδικό and
|	Δυαδικό or
^	Δυαδικό xor

~	Δυαδικό not
<<	Δυαδική ολίσθηση αριστερα
>>	Δυαδική ολίσθηση δεξιά

Μεταβλητές

HIGH | LOW	Κατά την ανάγνωση ή την εγγραφή σε ψηφιακό pin υπάρχουν μόνο δύο δυνατές καταστάσεις που μπορεί να πάρει: HIGH και LOW.	`digitalWrite(13,HIGH);` `digitalWrite(13,LOW);`
INPUT |OUTPUT |	Δήλωση για το αν ένα pin είναι είσοδος ή έξοδος	`pinMode(13,INPUT);` `pinMode(13,OUTPUT);`

void	Χρησιμοποιείται μόνο σε δηλώσεις συναρτήσεων. Αυτό δείχνει ότι η συνάρτηση αναμένεται να μην επιστρέψει κάποια πληροφορία για τη συνάρτηση από την οποία κλήθηκε.	`void setup()`
		`{`
		`  // ...`
		`}`
		`void loop()`
		`{`
		`  // ...}`

char	Ένα είδος δεδομένου που καταλαμβάνει μόλις 1 byte και αποθηκεύει χαρακτήρες.	`char myChar = 'A';`
		`char myChar = "ABC";`
		
		`char myChar = 65;`

byte	Είναι ένα 8-bit αριθμός απο 0 έως 255	`byte b = B10010;`
int	Είναι οι ακέραιοι αριθμοί. Ανάλογα με τον επεξεργαστή του Arduino μπορούν να αποθηκευτούν μικρότεροι ή μεγαλύτεροι αριθμοί.	`int ledPin = 13;` `int var = val;`
float	Είδος δεδομένων για αριθμούς κινούμενης υποδιαστολής.	`float myfloat;` `float sensorCalbrate = 1.117;`

Συναρτήσεις

pinMode()	Προσδιορίζει την συμπεριφορά ενός pin ως είσοδο ή ως έξοδο	`int ledPin = 13; // LED connected to digital pin 13`
		
		`void setup()`
		`{`
		`  pinMode(ledPin, OUTPUT); // sets the digital pin as output`
		`}`
		
		`void loop()`
		`{`
		`  digitalWrite(ledPin, HIGH); // sets the LED on`
		`  delay(1000); // waits for a second`
		`  digitalWrite(ledPin, LOW); // sets the LED off`
		`  delay(1000); // waits for a second`
		`}`

digitalWrite()	Καταχωρεί σε ένα pin την τιμή HIGH ή την τιμή LOW	`int ledPin = 13; // LED connected to digital pin 13`
		
		`void setup()`
		`{`
		`  pinMode(ledPin, OUTPUT); // sets the digital pin as output`
		`}`
		
		`void loop()`
		`{`
		`  digitalWrite(ledPin, HIGH); // sets the LED on`
		`  delay(1000); // waits for a second`
		`  digitalWrite(ledPin, LOW); // sets the LED off`
		`  delay(1000); // waits for a second`
		`}`

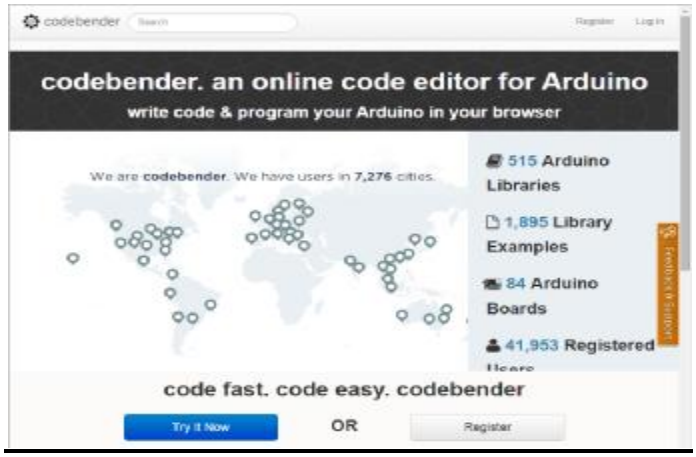
digitalRead()	Διαβάζει την τιμή ενός ψηφιακού pin	`digitalRead(pin);`
analogRead()	Διαβάζει την τιμή ενός αναλογικού pin	`analogRead(pin);`

analogWrite()	Γράφει μια αναλογική (PWM) τιμή σε ένα αναλογικό pin. Στην πραγματικότητα παράγει μια κυματομορφή με συχνότητα που διαφέρει από board σε board.	`int ledPin = 9;  // LED connected to digital pin 9`
		`int analogPin = 3;  // potentiometer connected to analog pin 3`
		`int val = 0;  // variable to store the read value`
		`void setup()`
		`{`
		`  pinMode(ledPin, OUTPUT);  // sets the pin as output`
		`}`
		`void loop()`
		`{`
		`  val = analogRead(analogPin);  // read the input pin`
		`  analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023, analogWrite values from 0 to 255`
		`}`

delay()	Διακόπτει το πρόγραμμα για όσο χρόνο(ms) ορίζει η παράμετρος μέσα στις παρενθέσεις.	`int ledPin = 13;  // LED connected to digital pin 13`
		
		`void setup()`
		`{`
		`  pinMode(ledPin, OUTPUT);  // sets the digital pin as output`
		`}`
		`void loop()`
		`{`
		`  digitalWrite(ledPin, HIGH);  // sets the LED on`
		`  delay(1000);  // waits for a second`
		`  digitalWrite(ledPin, LOW);  // sets the LED off`
		`  delay(1000);  // waits for a second`
		`}`

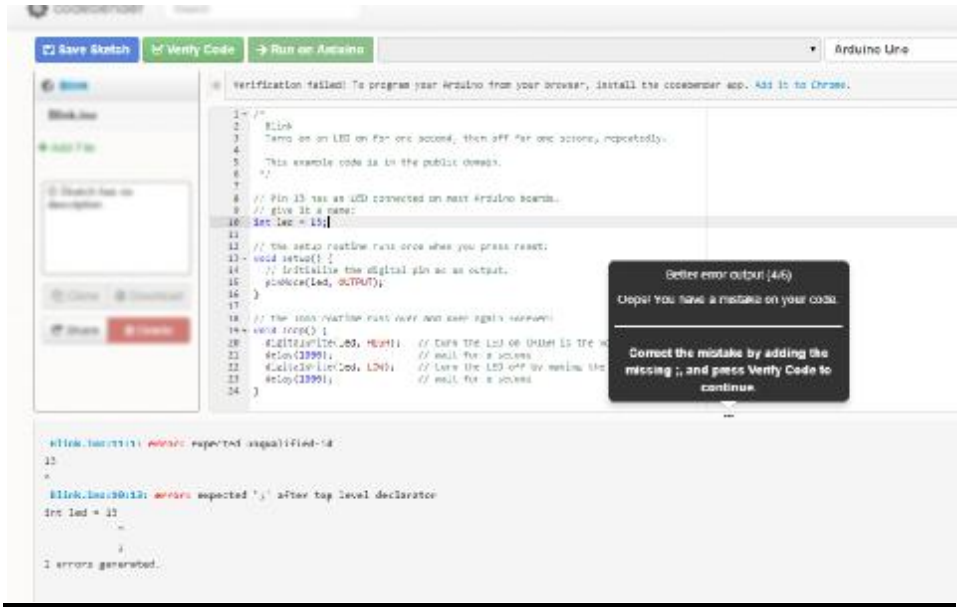
map()	Αλλάζει έναν αριθμό από μία κλίμακα σε μία άλλη.	`/* Map an analog value to 8 bits (0 to 255) */`
		`void setup() {}`
		
		`void loop()`
		`{`
		`  int val = analogRead(0);`
		`  val = map(val, 0, 1023, 0, 255);`
		`  analogWrite(9, val);}`

2.13 Διαδικτυακή πλατφόρμα προγραμματισμού codebender.cc



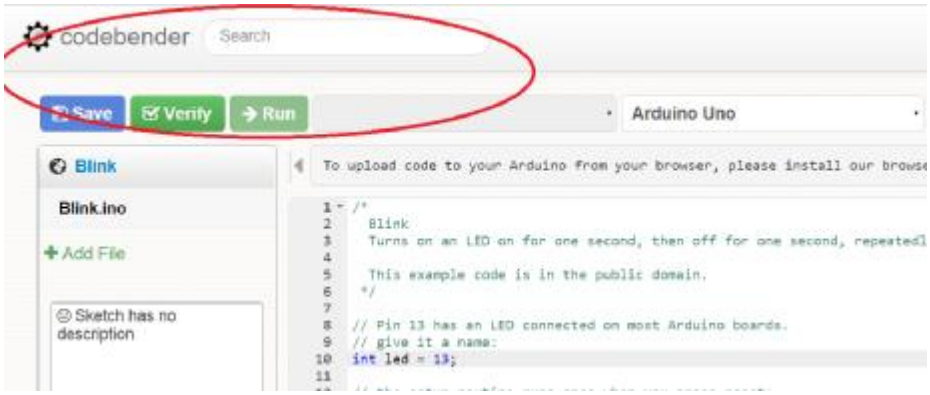
Εικόνα 2.17-codebender.cc

Το codebender (Εικόνα 2.17) είναι ένας online προγραμματιστής Arduino. Υποστηρίζει σχεδόν όλες τις εκδόσεις του Arduino και περιέχει μια μεγάλη βάση με κώδικες, παραδείγματα και βιβλιοθήκες. Το πιο βασικό απ' όλα είναι πως δίνει τη δυνατότητα στον χρήστη να προγραμματίσει την πλακέτα του από οποιοδήποτε μέρος του κόσμου στον browser του εύκολα και γρήγορα . Προφανώς μπορεί κάποιος να προσθέσει και τις δικιές του βιβλιοθήκες, καθώς επίσης δίνεται και η δυνατότητα να μοιραστεί κάποιος τον κώδικά του με τους υπόλοιπους χρήστες. Προγραμματίζεται με τον ίδιο ακριβώς τρόπο και έχει το πλεονέκτημα ότι σε περίπτωση συντακτικού λάθους προσδιορίζει ακριβώς το λάθος και δίνει άμεσα τη σωστή λύση. Στην επόμενη εικόνα (Εικόνα 2.18) βλέπουμε το περιβάλλον προγραμματισμού του Codebender.



Εικόνα 2.18-Περιβάλλον προγραμματισμού στο Codebender

Επίσης διαθέτει αναζήτηση (Εικόνα 2.19) για παραδείγματα και αλλά κώδικες άλλων χρηστών



Εικόνα 2.19-Μηχανή αναζήτησης Codebender

Τέλος όσον αφορά τις βιβλιοθήκες αρκεί να γράψουμε στον κώδικά μας #include και το όνομα τις βιβλιοθήκης που επιθυμούμε. Στις περισσότερες περιπτώσεις η βιβλιοθήκη υπάρχει στη βάση της πλατφόρμας και δεν χρειάζεται να την κατεβάσουμε εμείς. Εάν θέλουμε να κάνουμε αναζήτηση στις βιβλιοθήκες του Codebender αρκεί μια αναζήτηση στο site https://codebender.cc/libraries.

ΚΕΦΑΛΑΙΟ 3: Υλικό Μέρος Συστήματος

3.1 Arduino MEGA 2560

Εικόνα 3.1-Arduino Mega

Το Arduino /Genuino Mega 2560 είναι ένα board βασισμένο στον ATmega2560 και πρόκειται για αναβάθμιση του Arduino Mega το οποίο αντικαθιστά. Έχει 54 digital input/output pins (από τα οποία τα 15 μπορούν να χρησιμοποιηθούν σαν PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), ένα κρυσταλλικό ταλαντωτής 16 MHz, μια σύνδεση USB, ένα power jack, μια κεφαλή ICSP, και ένα reset button. Περιέχει ότι χρειάζεται για να υποστηρίξει τον μικροελεγκτή. Συνδέεται με τον υπολογιστή μέσω USB, ή τροφοδοτείται με ένα τροφοδοτικό AC-DC 7-12Volt ή με μια μπαταρία 9 Volt. Το Arduino Mega 2560 είναι συμβατό με τα περισσότερα shield που έχουν σχεδιαστεί για Arduino/Genuino Uno αλλά και για προηγούμενα boards όπως Duemilanove ή Diecimila.

Τεχνικά Χαρακτηριστικά

Microcontroller	**ATmega2560**
Operating Voltage	**5V**
Input Voltage (recommended)	**7-12V**
Input Voltage (limit)	**6-20V**
Digital I/O Pins	**54 (of which 15 provide PWM output)**
Analog Input Pins	**16**
DC Current per I/O Pin	**20 mA**
DC Current for 3.3V Pin	**50 mA**
Flash Memory	**256 KB of which 8 KB used by bootloader**
SRAM	**8 KB**
EEPROM	**4 KB**
Clock Speed	**16 MHz**
Length	**101.52 mm**
Width	**53.3 mm**
Weight	**37 g**

Το Arduino / Genuino Mega 2560 είναι ένα υλικό ανοικτής πηγής (open source). Πράγμα που σημαίνει πως μπορούμε να φτιάξουμε το δικό μας board ακλουθώντας τα σχέδια που δίνονται στην επίσημη σελίδα του Arduino http://www.arduinno.cc

3.1.1Προγραμματισμός Arduino Mega

Το Arduino/Genuino 2560 Mega μπορεί να προγραμματιστεί με το Arduino IDE για το οποίο θα μιλήσουμε αργότερα. Είναι προγραμματισμένο εκ των προτέρων με ένα bootloader ο οποίος μας επιτρέπει να "ανεβάσουμε" νέο κώδικα σε αυτό χωρίς να χρειαζόμαστε εξωτερικό προγραμματιστή υλικού. Η επικοινωνία γίνεται με το πρωτόκολλο STK500.

Μπορούμε να προσπεράσουμε τον bootloader και να προγραμματίσουμε τον μικροελεγκτή μέσω της κεφαλής ICSP (In-Circuit Serial Programming) χρησιμοποιώντας ένα Arduino ISP.

3.1.2 Προσοχή/Προειδοποιήσεις για Arduino/Genuino 2560 Mega

Το Arduino/Genuino 2560 Mega έχει επαναρρυθμιζόμενη ασφάλεια που προστατεύει τις θύρες USB του υπολογιστή μας από βραχυκύκλωμα και υπερένταση. Ωστόσο οι περισσότεροι υπολογιστές έχουν δικό τους σύστημα προστασίας, η ασφάλεια παρέχει μια επιπλέον βαθμίδα προστασίας. Εάν εφαρμοστούν περισσότερα από 500mA στη θύρα USB τότε η ασφάλεια θα διακόψει την επικοινωνία μέχρι το βραχυκύκλωμα ή η υπερένταση να αποκατασταθεί.

3.1.3 Τροφοδοσία Arduino/Genuino 2560 Mega

Το Arduino/Genuino 2560 Mega μπορεί να τροφοδοτηθεί από θύρα USB ή από εξωτερική τροφοδοσία ρεύματος. Η πηγή τροφοδοσίας επιλέγεται αυτόματα. Η εξωτερική τροφοδοσία ρεύματος μπορεί να είναι είτε ένα τροφοδοτικό AC-DC ή με μια μπαταρία. Το τροφοδοτικό μπορεί να συνδεθεί συνδέοντας ένα βύσμα 2.1 με θετικό κέντρο στο power jack του board. Η μπαταρία μπορεί να συνδεθεί στο GRD και στο Vin. Η πλακέτα μπορεί να λειτουργήσει με εξωτερική πηγή από 6 έως 20 Volt. Εάν τροφοδοτείται με λιγότερο από 7V, εντούτοις, το Pin 5V μπορεί να παρέχει λιγότερο από 5 Volt και η πλακέτα μπορεί να γίνει ασταθής. Εάν χρησιμοποιηθεί τάση μεγαλύτερη από 12 Volt, ο ρυθμιστής τάσης μπορεί να υπερθερμανθεί και να προκαλέσει ζημιά στην πλακέτα. Το συνιστώμενο εύρος τάσης τροφοδοσίας είναι 7 έως 12 Volt.

3.1.4 Τα Pin τροφοδοσίας του Arduino/Genuino 2560 Mega

Vin. Η τάση εισόδου στην πλακέτα όταν αυτή χρησιμοποιεί μια εξωτερική πηγή (σε αντίθεση με τα 5 Volt από τη σύνδεση USB ή άλλες ρυθμιζόμενες πηγή ενέργειας). Μπορούμε να παρέχουμε τάση μέσω αυτού του pin, ή, αν έχουμε παροχή τάσης μέσω του power jack, έχουμε πρόσβαση σε αυτή την τάση μέσω αυτού του pin.

5V. Αυτό το pin εξάγει μια ρυθμιζόμενη 5V από τον ρυθμιστή πάνω στην πλακέτα. Η πλακέτα μπορεί να τροφοδοτείται με ρεύμα είτε από το βύσμα συνεχούς ρεύματος (7 - 12V), τη θύρα USB (5V), ή το pin Vin της πλακέτας (7-12V). Παρέχοντας τάση μέσω των pin 5V ή 3.3V παρακάμπτει τον ρυθμιστή, και μπορεί να προκαλέσει βλάβη στην πλακέτα.

3,3V. Μια παροχή 3.3 volt από τον ρυθμιστή της πλακέτας με μέγιστη παροχή ρεύματος 50 mA.

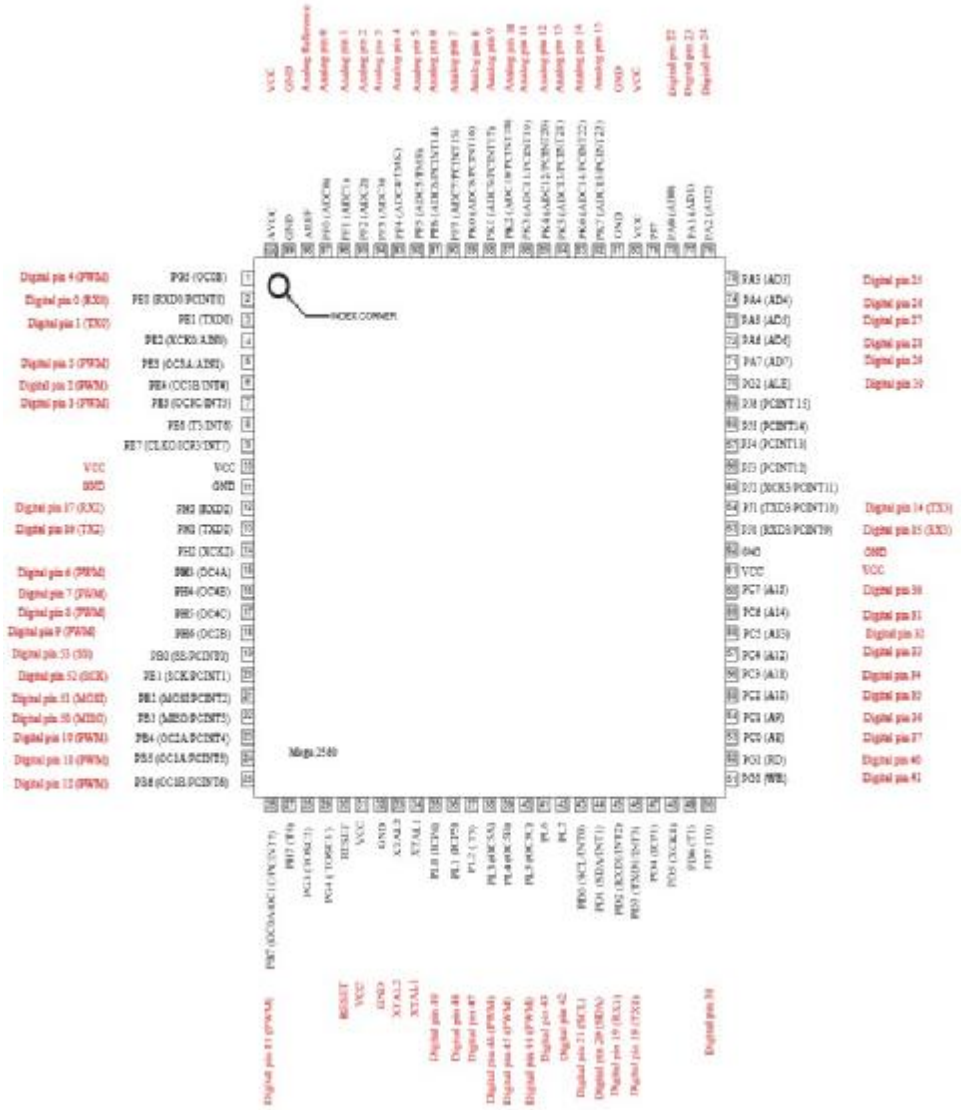
GND. Pins γείωσης.

IOREF. Αυτό το pin πάνω στην πλακέτα παρέχει την τάση αναφοράς με την οποία ενεργεί ο μικροελεγκτής. Ένα σωστά ρυθμισμένο Shield μπορεί να διαβάσει την τάση του pin IOREF και επιλέξτε την κατάλληλη πηγή ενέργειας ή να ενεργοποιήσει τους μεταφραστές τάσης στις εξόδους για εργασία με 5V ή 3.3V.

3.1.5 Μνήμη Arduino/Genuino 2560 Mega

Ο ATmega2560 έχει 256KB flash μνήμης για να αποθηκεύει κώδικα (από την οποία 8KB χρησιμοποιούνται από τον bootloader), 8KB SRAM και 4KB EEPROM (η οποία μπορεί να διαβαστεί ή να γραφτεί με τη βιβλιοθήκη EEPROM).

3.1.6 Είσοδοι και Έξοδοι Arduino/Genuino 2560 Mega



Εικόνα 3.2- Διάγραμμα Pin μικροελεγκτή Arduino Mega 2560

Arduino Mega 2560 PIN mapping table

Pin	Pin Name	Mapped
1	PG5 (OC0B)	Digital pin
2	PE0 (RXD0/PCINT8)	Digital pin
3	PE1 (TXD0)	Digital pin
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin
6	PE4 (OC3B/INT4)	Digital pin
7	PE5 (OC3C/INT5)	Digital pin
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin
13	PH1 (TXD2)	Digital pin
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin
16	PH4 (OC4B)	Digital pin
17	PH5 (OC4C)	Digital pin
18	PH6 (OC2B)	Digital pin
19	PB0 (SS/PCINT0)	Digital pin
20	PB1 (SCK/PCINT1)	Digital pin
21	PB2 (MOSI/PCINT2)	Digital pin
22	PB3 (MISO/PCINT3)	Digital pin
23	PB4 (OC2A/PCINT4)	Digital pin
24	PB5 (OC1A/PCINT5)	Digital pin
25	PB6 (OC1B/PCINT6)	Digital pin
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin
36	PL1 (ICP5)	Digital pin
37	PL2 (T5)	Digital pin
38	PL3 (OC5A)	Digital pin
39	PL4 (OC5B)	Digital pin
40	PL5 (OC5C)	Digital pin
41	PL6	Digital pin
42	PL7	Digital pin
43	PD0 (SCL/INT0)	Digital pin
44	PD1 (SDA/INT1)	Digital pin
45	PD2 (RXDI/INT2)	Digital pin
46	PD3 (TXD1/INT3)	Digital pin
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin
51	PG0 (WR)	Digital pin
52	PG1 (RD)	Digital pin
53	PC0 (A8)	Digital pin
54	PC1 (A9)	Digital pin
55	PC2 (A10)	Digital pin
56	PC3 (A11)	Digital pin
57	PC4 (A12)	Digital pin
58	PC5 (A13)	Digital pin
59	PC6 (A14)	Digital pin
60	PC7 (A15)	Digital pin
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin
64	PJ1 (TXD3/PCINT10)	Digital pin
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin
71	PA7 (AD7)	Digital pin
72	PA6 (AD6)	Digital pin
73	PA5 (AD5)	Digital pin
74	PA4 (AD4)	Digital pin
75	PA3 (AD3)	Digital pin
76	PA2 (AD2)	Digital pin
77	PA1 (AD1)	Digital pin
78	PA0 (AD0)	Digital pin
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog
83	PK6 (ADC14/PCINT22)	Analog
84	PK5 (ADC13/PCINT21)	Analog
85	PK4 (ADC12/PCINT20)	Analog
86	PK3 (ADC11/PCINT19)	Analog
87	PK2 (ADC10/PCINT18)	Analog
88	PK1 (ADC9/PCINT17)	Analog
89	PK0 (ADC8/PCINT16)	Analog
90	PF7 (ADC7)	Analog
91	PF6 (ADC6)	Analog
92	PF5 (ADC5/TMS)	Analog
93	PF4 (ADC4/TMK)	Analog
94	PF3 (ADC3)	Analog
95	PF2 (ADC2)	Analog
96	PF1 (ADC1)	Analog
97	PF0 (ADC0)	Analog
98	AREF	Analog
99	GND	GND
100	AVCC	VCC

Κάθε ένα από τα 54 ψηφιακά pin του Mega μπορούν να χρησιμοποιηθούν είτε ως είσοδοι είτε ως έξοδοι, χρησιμοποιώντας της εντολές/συναρτήσεις pinMode(), digitalWrite(), και digitalRead(). Λειτουργούν στα 5Volt. Κάθε pin μπορεί να παρέχει ή να δεχθεί 20mA σε συνιστώμενες συνθήκες και διαθέτει εσωτερικές αντιστάσεις ανύψωσης (pull-up resistor) οι οποίες είναι αποσυνδεδεμένες από προεπιλογή. Η μέγιστη τιμή των 40mA είναι μία τιμή πού δεν πρέπει να ξεπεραστεί έτσι ώστε να αποφευχθούν μόνιμές βλάβες στον μικροελεγκτή.

Επιπλέον μερικά pin έχουν ειδικές λειτουργίες:

Serial: 0 (RX) και 1 (TX); Serial 1: 19 (RX) και 18 (TX); Serial 2: 17 (RX) και 16 (TX); Serial 3: 15 (RX) και 14 (TX). Χρησιμοποιούνται για να δέχεται (RX) και να

στέλνει (TX) TTL σειριακά δεδομένα. Τα Pins 0 και 1 συνδέονται στα αντίστοιχα pins του ATmega16U2 USB-σε-TTL Serial chip.

External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). Αυτά τα pin μπορούν να ρυθμιστούν έτσι ώστε να ενεργοποιούν μια διακοπή (interrupt) σε ένα χαμηλό επίπεδο, μια αύξηση ή πτώση ή την αλλαγή επιπέδου.

PWM: Τα Pin 2 με 13 και 44 με 46. Παρέχουν 8-bit PWM έξοδο με τη συνάρτηση analogWrite().

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Αυτά τα pin παρέχουν επικοινωνία SPI με τη χρήση της βιβλιοθήκης SPI. Τα Pins SPI συνδέονται επίσης με την ICSP κεφαλή που είναι φυσικά συμβατή με το Arduino/Genuino Uno και τα παλιά Duemilianove και Diecimila.

LED: 13 Υπάρχει ένα ενσωματωμένο LED το οποίο ελέγχεται το pin 13. Όταν το pin έχει την τιμή HIGH το LED είναι αναμμένο, και όταν το pin έχει την τιμή LOW το LED είναι σβηστό.

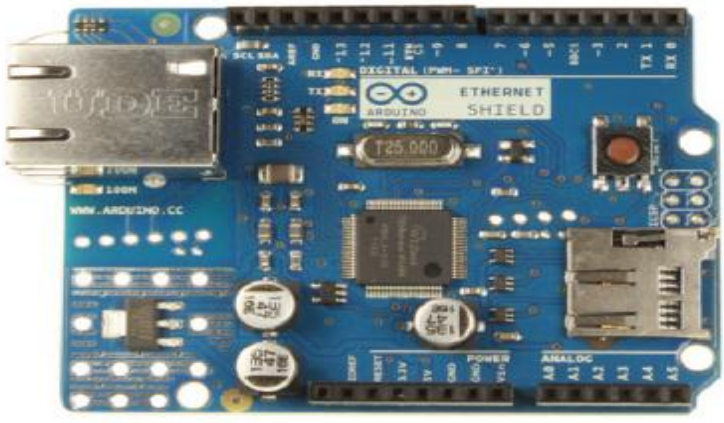
TWI: 20 (SDA) και 21 (SCL). Αυτά τα pin υποστηρίζουν επικοινωνία TWI χρησιμοποιώντας τη βιβλιοθήκη Wire. Υπόψη πως δεν βρίσκονται στην ίδια θέση με τα αντίστοιχα pin παλιότερων board.

Το Arduino/Genuino Mega 2560 έχει 16 αναλογικές εισόδους, κάθε μια από τις οποίες μπορεί να παρέχει 10Bits ανάλυσης δηλαδή 1024 διαφορετικές τιμές. Από προεπιλογή μετρούν από γείωση μέχρι 5 volt, ωστόσο είναι δυνατό να αλλάξουμε το άνω όριο χρησιμοποιώντας το pin AREF και την συνάρτηση analogReference().

AREF. Τάση αναφοράς των αναλογικών pin. Χρησιμοποιείται με την εντολή analogReference().

Reset. Βάζοντας σε αυτό το Pin LOW κάνουμε επαναφορά στον μικροελεγκτή. Τυπικά χρησιμοποιείται για να προσθέσουμε Button επαναφοράς όταν κάποιο shield εμποδίζει το εγκατεστημένο.

3.2 Επικοινωνία Arduino με το διαδίκτυο (Ethernet Shield)



Εικόνα 3.3-Ethernet Shield

Το Arduino Ethernet Shield συνδέει το Arduino με το internet. Τοποθετώντας το πάνω στο Arduino, συνδέοντας το στο δίκτυο με ένα καλώδιο RJ45 και με λίγες γραμμές κώδικα μπορούμε να ελέγχουμε την πλακέτα μας από απόσταση, να στέλνουμε αλλά και να δεχόμαστε πληροφορίες. Οι απαιτήσεις του Ethernet Shield είναι:

- μια πλακέτα Arduino
- τροφοδοσία 5Volt την οποία λαμβάνει από το Arduino
- Ethernet Controller W5100 με εσωτερικό buffer 16K
- ταχύτητα σύνδεσης 10/100Mb
- σύνδεση με το Arduino στη θύρα SPI

Το W5100 παρέχει δίκτυο (IP) ικανό για TCP και για UDP. Υποστηρίζει μέχρι και τέσσερις ταυτόχρονες συνδέσεις socket. Για να συνδεθούμε στο Internet, πέρα από τον κώδικά μας, απαραίτητη προϋπόθεση είναι και η βιβλιοθήκη Ethernet. Το Ethernet Shield συνδέεται στο Arduino χρησιμοποιώντας μακριά σύρματα-ακίδες που εξέχουν από το Shield και τοποθετούνται στα αντίστοιχα Pin του Arduino. Αυτός ο τρόπος επιτρέπει τη διάταξη των pin να παραμένει σχεδόν άθικτη και δίνει τη δυνατότητα να προσθέσουμε και να στοιβάξουμε κι άλλα Shields.

Το Arduino Ethernet Shield έχει μια πρότυπη RJ-45 σύνδεση με ένα ενσωματωμένο μετασχηματιστή για τροφοδοσία μέσω Ethernet.

Επίσης υπάρχει ενσωματωμένη θέση για κάρτα μνήμης micro-SD, η οποία μπορεί να χρησιμοποιηθεί για αποθήκευση αρχείων για την εξυπηρέτηση μέσω του δικτύου. Ο ενσωματωμένος αναγνώστης microSD μπορεί να γίνει προσβάσιμος μόνο με τη βιβλιοθήκη SD. Όταν εργαζόμαστε με αυτή τη βιβλιοθήκη το SS βρίσκεται στο pin 4.

Το Arduino επικοινωνεί με το W5100 την κάρτα SD χρησιμοποιώντας το SPI bus (μέσω της ICSP κεφαλής). Αυτά βρίσκονται στα ψηφιακά pin 10,11,12 και 13 στο Uno και στα pin 50,51,52 στο Mega. Και στις δύο πλακέτες το pin 10 χρησιμοποιείται για να επιλέγει τον W5100 και το pin 4 για την SD κάρτα. Αυτά τα pin επομένως δεν μπορούν να χρησιμοποιηθούν για γενικές είσοδοι ή έξοδοι. Στο Mega, το υλικό SS pin 53 δεν χρησιμοποιείται ούτε για να επιλέξει τον W5100 ούτε την κάρτα SD, αλλά πρέπει να είναι πάντα δηλωμένο ως έξοδος διαφορετικά το περιβάλλον SPI δεν θα λειτουργεί.

Επίσης, επειδή και η κάρτα SD και ο W5100 χρησιμοποιούν τον δίαυλο SPI, μπορεί μόνο ένα από τα δύο να λειτουργεί κάθε φορά. Εάν στον κώδικά μας χρησιμοποιούμε και τα δύο αυτά περιφερειακά θα πρέπει να χρησιμοποιήσουμε τις κατάλληλες βιβλιοθήκες. Εάν δεν χρησιμοποιούμε κάποιο από τα δυο αυτά περιφερειακά στο πρόγραμμά μας ωστόσο, πρέπει ρητά να το απενεργοποιήσουμε. Για να γίνει αυτό, στην περίπτωση του W5100, θέτουμε το pin 10 HIGH ως έξοδο, ενώ στην περίπτωση της SD, θέτουμε το pin 4 ως HIGH έξοδο.

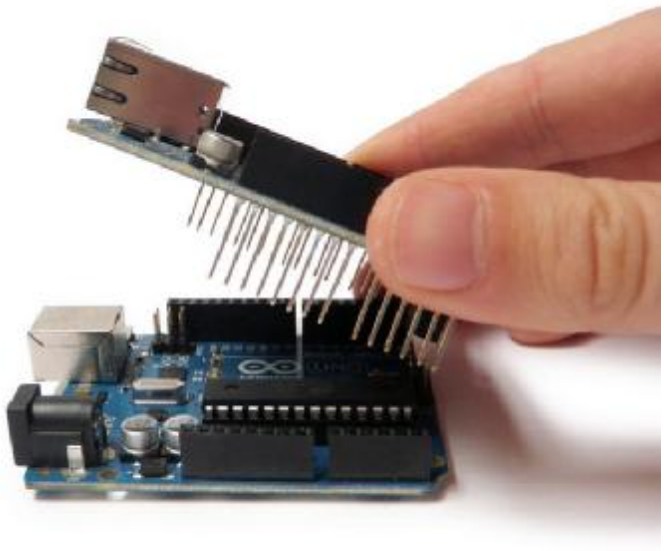
Το reset button στο Ethernet Shield επανεκκινεί και το W5100 και το Arduino.

Επιπλέον το Arduino Ethernet Shield έχει έναν αριθμό LED ενδείξεων:

- PWR: Ένδειξη πως η πλακέτα και το shield έχουν τροφοδοσία.

- LINK: Ένδειξη παρουσίας δικτύου, αναβοσβήνει όταν το shield στέλνει ή δέχεται δεδομένα.
- FULLD: Ένδειξη πως η σύνδεση είναι αμφίδρομη στο ίδιο κανάλι (full duplex).
- 100M: Ένδειξη πως έχουμε σύνδεση δικτύου 100Mb/s (σε αντίθεση με 10 Mb / s).
- RX: αναβοσβήνει όταν το shield δέχεται δεδομένα.
- TX: αναβοσβήνει όταν το shield στέλνει δεδομένα.
- COLL: αναβοσβήνει όταν ανιχνεύονται συγκρούσεις δικτύου.

Στην επόμενη φωτογραφία (Εικόνα 3.4) φαίνεται ο τρόπος σύνδεσης του Shield με το Arduino



Εικόνα 3.4-Σύνδεση Arduino με Ethernet Shield

3.3 Αισθητήρες που χρησιμοποιήθηκαν

3.3.1 Microbot DHT11: Ψηφιακός αισθητήρας υγρασίας και θερμοκρασίας

Πρόκειται για μία πλακέτα για τον αισθητήρα DHT11 ο οποίος δίνει μία ψηφιακή έξοδο που είναι ανάλογη με την θερμοκρασία και την υγρασία που μετράται από τον αισθητήρα. Η τεχνολογία που χρησιμοποιείται για αυτό το προϊόν εγγυάται μεγάλη αξιοπιστία, εξαιρετική μακροχρόνια σταθερότητα και γρήγορο χρόνο απόκρισης.

Κάθε στοιχείο του αισθητήρα DHT11 είναι βαθμονομημένο με ακρίβεια στο εργαστήριο. Ο συντελεστής βαθμονόμησης είναι αποθηκευμένος σε μια εσωτερική μνήμη OTP και αυτή η τιμή χρησιμοποιείται από την εσωτερική διαδικασία ανίχνευσης σήματος.

Η ενιαία σειριακή διασύνδεση καθιστά την ενσωμάτωση αυτού του αισθητήρα σε ψηφιακό σύστημα γρήγορα και εύκολα.

Η φυσική διασύνδεση του αισθητήρα γίνεται μέσω ενός βύσματος τριών pin τα οποία είναι +5Volt, GND και DATA(δεδομένα). Τα πρώτα δύο pin αποτελούν την παροχή τάσης τροφοδοσίας και γείωσης του αισθητήρα και το τρίτο pin είναι η ψηφιακή έξοδος του αισθητήρα.

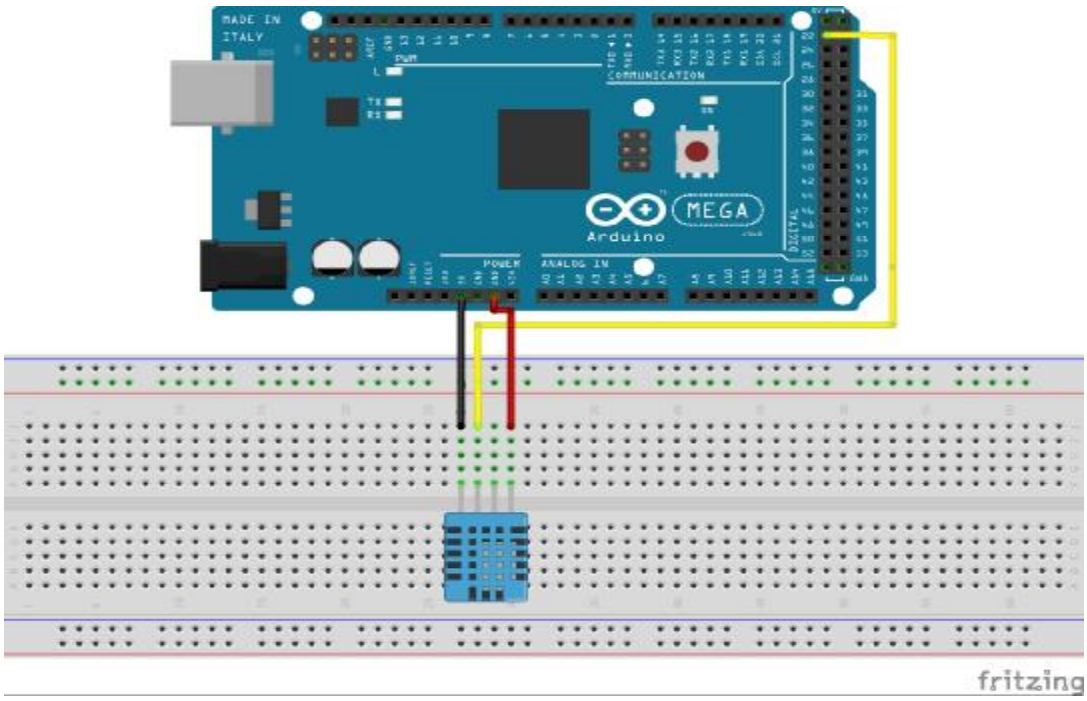
Είναι μικρός σε μέγεθος(1.05 "x0.7") και πολύ ελαφρύ (μόλις 0,1oz=2,7gr), γεγονός που τον κάνουν ιδανικό για κατασκευές ρομπότ και συστημάτων παρακολούθησης περιβάλλοντος.

Χαρακτηριστικά

Supply voltage	+5V
Supply current (running)	0.5mA typ. (2.5mA max.)
Supply current (stand-by)	100uA typ. (150uA max.)
Temperature range	0 / +50°C ±2°C
Humidity range	20-90%RH ±5%RH
Interface	Digital
Dimensions	1.05” x 0.7” (connectors excluded)
Weight	2.7g (0.1 oz)

Σύνδεση με το Arduino

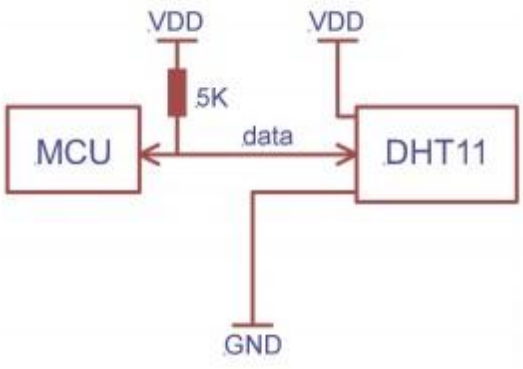
DATA	Serial data output
GRD	Ground
+5V	Power supply(+5Volt)



Εικόνα 3.5-Συνδεσμολογια για προγραμματισμό DHT11

3.3.1.1 Αξιοποίηση αισθητήρα

Ο μονόκλωνος δίαυλος χρειάζεται μια αντίσταση ανόρθωσης 5KΩ (pull-up resistor). Το κύκλωμα του συστήματος είναι το εξής (Εικόνα 3.6):



Εικόνα 3.6-Κύκλωμα αισθητήρα DHT11

Όταν παρέχεται τροφοδοσία στον αισθητήρα δεν πρέπει να του στείλουμε κανένα σήμα για ένα δευτερόλεπτο, μέχρι να περάσει δηλαδή ο χρόνος εκκίνησης. Μόλις ο DHT11 τροφοδοτείται μπαίνει σε κατάσταση αναμονής(standby mode) και περιμένει να αναγνωρίσει ένα σήμα εκκίνησης στη γραμμή δεδομένων. Το σήμα εκκίνησης αποτελείται από χαμηλού επιπέδου τάση για έναν ελάχιστο χρόνο περίπου 18ms. Για να εξασφαλιστεί πως ο DHT11 θα το ανιχνεύσει, στη συνέχεια ακολουθεί μια μεγαλύτερη τάση (Pull-up) για διάρκεια 40μs. Τώρα ο μικροελεγκτής πρέπει να περιμένει την μετάδοση του αισθητήρα. Μόλις ο αισθητήρας ανιχνεύσει το σήμα εκκίνησης θα στείλει μια χαμηλή τιμή τάσης διάρκειας 80μs. Στη συνέχεια, καθορίζει το επίπεδο τις τάσης από χαμηλή σε υψηλή και την διατηρεί για 80μs. Τώρα μπορεί να ξεκινήσει η μετάδοση δεδομένων. Κάθε bit δεδομένων ξεκινάει από το επίπεδο 50ms χαμηλής τάσης και στη συνέχεια μεταβαίνει σε υψηλότερη τάση. Η διάρκεια της υψηλής τάσης εξαρτάται από την τιμή των bit που πρέπει να μεταδοθούν. 1bit έχει υψηλή τάση για διάρκεια 27μs ενώ 0bit έχει υψηλή τάση για διάρκεια 70μs.

Όταν το τελευταίο bit δεδομένων μεταδοθεί, ο αισθητήρας μηδενίζει την τάση και τη διατηρεί για 50μs. Στη συνέχεια αφήνει την γραμμή ανορθωμένη(pulled-up) και επιστρέφει σε κατάσταση standby.

Για να κάνουμε έναν άλλο αισθητήρα να τον διαβάσει χρειάζεται να επαναληφθεί αυτός ο κύκλος, στέλνοντας ξανά το σήμα εκκίνησης ένα δευτερόλεπτο μετά από τον πρώτο κύκλο.

Μία πλήρης διαδικασία μετάδοσης δεδομένων είναι 40bit και η διαδικασία επικοινωνίας περίπου 4ms. Ο DHT11 στέλνει τα πιο σημαντικά bit δεδομένων(MSB) με την ακόλουθη μορφή:

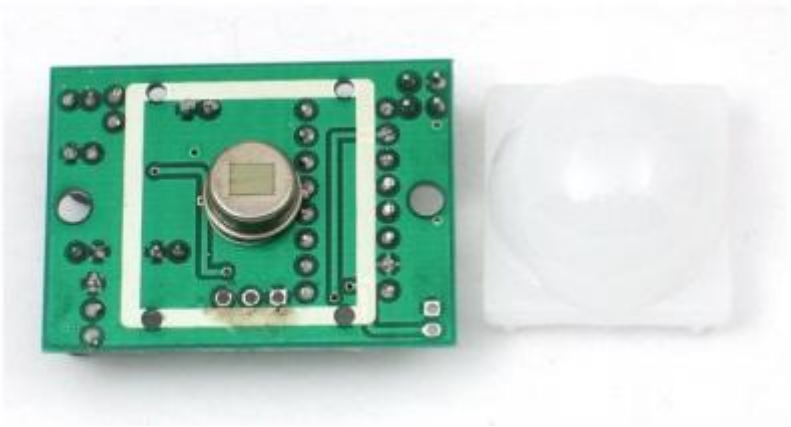
Data = 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check-sum.

Εάν η μετάδοση δεδομένων είναι σωστή τότε το check-sum θα είναι ίσο με τα τελευταία 8bit του αθροίσματος από τα 4 byte που μεταφέρθηκαν.

3.3.2 PIR Ανιχνευτής Κίνησης

Οι αισθητήρες PIR μας επιτρέπουνε να αναγνωρίσουμε κίνηση, σχεδόν πάντα χρησιμοποιούνται για να εντοπίσουν εάν ένας άνθρωπος έχει κινηθεί εντός ή εκτός της εμβέλειας του αισθητήρα. Είναι μικροί, φθηνοί, καταναλώνουν χαμηλή ισχύ, είναι εύκολοι στη χρήση και δεν φθείρονται. Για το λόγο αυτό βρίσκονται συνήθως σε συσκευές και gadgets που χρησιμοποιούνται στα σπίτια ή τις επιχειρήσεις. Συχνά αναφέρεται ως PIR, «Παθητικοί υπερύθρων" (Passive Infrared), "πυροηλεκτρικοί" (Pyroelectric), ή "IR κίνησης "αισθητήρες.

Οι αισθητήρες PIR είναι κατά βάση κατασκευασμένοι από έναν πυροηλεκτρικό αισθητήρα, ένα υλικό που παράγει τάση όταν θερμανθεί ή ψυχθεί, και μπορεί να αντιληφτεί επίπεδα υπέρυθρης ακτινοβολίας. Στης επόμενη φωτογραφία (Εικόνα 3.7) ο αισθητήρας φαίνεται σαν ένα στρογγυλό μεταλλικό δοχείο με ένα ορθογώνιο κρύσταλλο στο κέντρο το οποίο είναι το υλικό που αναφέραμε παραπάνω.



Εικόνα 3.7-Αισθητήρας PIRκαι φακός

Οτιδήποτε εκπέμπει κάποια ακτινοβολία χαμηλού επιπέδου, και όσο πιο θερμό είναι τόσο μεγαλύτερη ακτινοβολία εκπέμπει. Ο αισθητήρας στην πράξη είναι χωρισμένος σε δύο ίσα μέρη. Ο λόγος είναι πως ψάχνουμε για την ανίχνευση κίνησης και επομένως ψάχνουμε για κάποια αλλαγή και όχι για κάποια μέση τιμή υπέρυθρης ακτινοβολίας. Τα δύο μισά είναι μεταξύ τους ενωμένα έτσι ώστε να αλληλοεξουδετερώνονται. Εάν το ένα από τα δύο μισά ανιχνεύσει περισσότερη ή λιγότερη ακτινοβολία από το άλλο μισό η έξοδος γίνεται HIGH ή LOW.

Μαζί με τον αισθητήρα υπάρχει και ένα ολόκληρο κύκλωμα γεμάτο αντιστάσεις και πυκνωτές. Το κύκλωμα συνήθως οδηγείται από έναν επεξεργαστή BISS0001 όπως φαίνεται παρακάτω (Εικόνα 3.8). Όπως οι περισσότεροι αισθητήρες έχει τρία άκρα GRD, Digital Output και τροφοδοσία 3-5Vdc.

Εικόνα 3.8-Κύκλωμα Αισθητήρα PIR

Για πολλά βασικά projects που χρειάζονται ανίχνευση κίνησης για το πότε δηλαδή ένα άτομο έχει προσεγγίσει την περιοχή εμβέλειας του αισθητήρα, οι PIR είναι μια εξαιρετική επιλογή. Είναι χαμηλής ισχύος και χαμηλού κόστους, έχουν ένα ευρύ φάσμα φάσμα φακού και είναι εύκολοι στη σύνδεση. Τα αρνητικά τους είναι πως δεν έχουν τη δυνατότητα να μας ενημερώσουν για το πόσα άτομα βρίσκονται στον χώρο ή σε ποιά απόσταση βρίσκεται κάποιος από τον αισθητήρα.

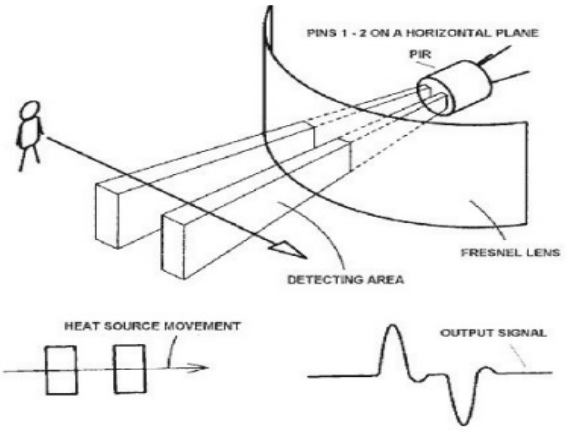
Βασικά χαρακτηριστικά

- Color: White + Green
- Dimension: 3.2cm x 2.4cm x 1.8cm (approx)
- Infrared sensor with control circuit board
- The sensitivity and holding time can be adjusted
- Working Voltage Range: DC 4.5V- 20V
- Current drain:<60μA
- Voltage Output: High/Low level signal:3.3V TTL output
- Detection distance: 3--7M(can be adjusted)
- Detection range: <140°
- Delay time: 5-200S(can be adjusted, default 5s +-3%)

3.3.2.1 Πώς λειτουργεί ένας PIR;

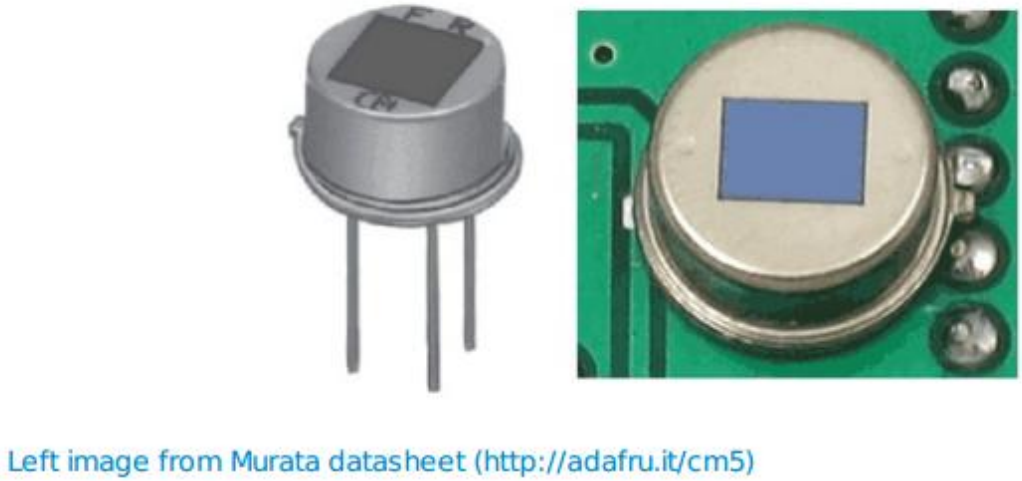
Ο ίδιος ο αισθητήρας αποτελείται από δύο κομμάτια, κάθε ένα φτιαγμένο από ένα ειδικό υλικό που είναι ευαίσθητο στην υπέρυθρη ακτινοβολία. Όταν ο αισθητήρας βρίσκεται σε αναμονή και τα δύο κομμάτια αναγνωρίζουν το ίδιο πόσο υπέρυθρης ακτινοβολίας, δηλαδή το ποσό της ακτινοβολίας που αντανακλάται από το δωμάτιο, τους τοίχους κλπ. Όταν όμως ένα θερμό σώμα όπως ένας άνθρωπος ή ένα ζώο περάσει μπροστά από τον αισθητήρα, πρώτα ανιχνεύεται από το πρώτο μισό του αισθητήρα και αυτό δημιουργεί μια θετική διαφορά δυναμικού μεταξύ των δύο μερών

του αισθητήρα. Όταν το θερμό σώμα φύγει από το πεδίο ανίχνευσης τότε συμβαίνει το αντίθετο και δημιουργείται μια αρνητική διαφορά δυναμικού. Αυτή η εναλλαγή των παλμών είναι η αιτία που προκαλεί την ανίχνευση της κίνησης όπως φαίνεται στην Εικόνα 3.9.



Εικόνα 3.9-Λειτουργία ανίχνευσης κίνησης PIR

Ο αισθητήρας στεγάζεται μέσα σε ένα ερμητικά σφραγισμένο μέταλλο έτσι ώστε να είναι προστατευμένος από θόρυβο θερμοκρασία και υγρασία. Υπάρχει ένα παράθυρο φτιαγμένο από υλικό διαπερατό από υπέρυθρη ακτινοβολία(συνήθως πυρίτιο) που προστατεύει το αισθητήριο. Πίσω από αυτό το παράθυρο βρίσκονται τα δύο ίσα κομμάτια του αισθητήρα που έχουμε αναφέρει. Η κατασκευή αυτή φαίνεται παρακάτω (Εικόνα 3.10).



Εικόνα 3.10-Μεταλλικό περίβλημα και παράθυρο αισθητήρα PIR

Παρακάτω (Εικόνα 3.11) μπορούμε να δούμε στο διάγραμμα το παράθυρο του στοιχείου που αναφέραμε καθώς και τα δύο μέρη του αισθητήρα:

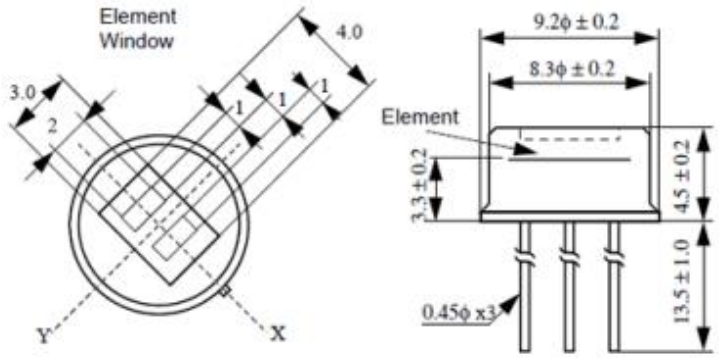


Image from RE200B datasheet (http://adafru.it/clS)

Εικόνα 3.11-Παράθυρο αισθητήρα και διαστάσεις

Η παρακάτω εικόνα (Εικόνα 3.12) δείχνει το εσωτερικό του αισθητήρα. Στην πραγματικότητα αποτελείται από ένα JFET(σε συνεργασία με chip όπως το BIS0001).

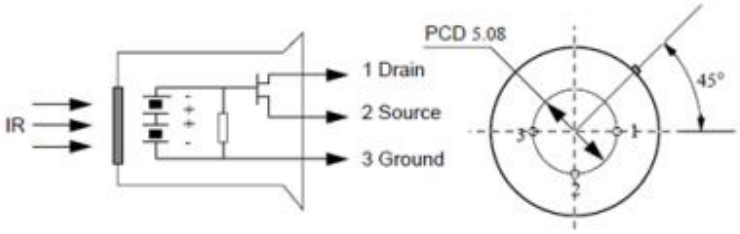


Image from RE200B datasheet (http://adafru.it/clS)

Εικόνα 3.12-Εσωτερικό κύκλωμα αισθητήρα PIR

3.3.2.2 Ο φακός του PIR

Οι αισθητήρες PIR σε γενικές γραμμές διαφέρουν μόνο στην τιμή και την ευαισθησία τους. Αυτό όπως που παίζει πολύ σημαντικό ρόλο είναι ο φακός ο οποίος μπορεί να κοστίζει μόνο μερικά λεπτά αλλά είναι ικανός να αλλάξει το πλάτος, το εύρος και το μοντέλο αίσθησης πολύ εύκολα.

Συνήθως θέλουμε να έχουμε μια μεγάλη περιοχή ανίχνευσης. Για να γίνει αυτό θα μπορούσαμε να χρησιμοποιήσουμε ένα φακό σαν αυτό που βρίσκουμε στις κάμερες, αφού συμπυκνώνουν μια μεγάλη περιοχή όπως ένα τοπίο σε ένα μικρό φιλμ. Για λόγους που θα εξηγηθούν παρακάτω οι φακοί που χρησιμοποιούνται εν τέλει

αποτελούνται από φθηνό πλαστικό με παραμορφώσεις όπως οι φακοί Fresnel (Εικόνα 3.13). Οι φακοί αυτοί συμπυκνώνουν το φώς παρέχοντας ένα ευρύ φάσμα υπέρυθρων στον αισθητήρα.

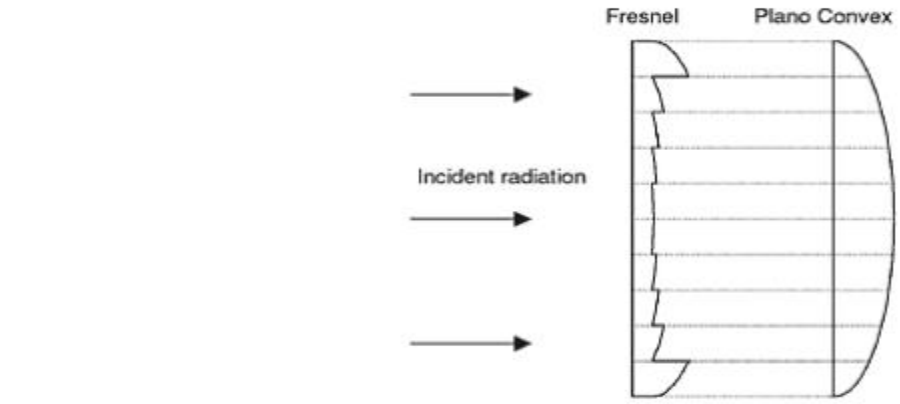


Image from Sensors Magazine (http://adafru.it/aKs)

Εικόνα 3.13-Φακός Fresnel

Στην επόμενη εικόνα (Εικόνα 3.14) βλέπουμε τον τρόπο λειτουργίας των φακών Fresnel

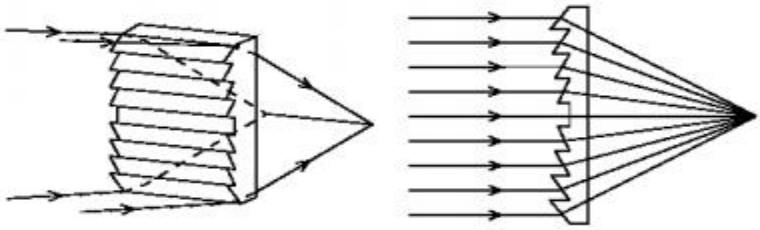


Image from BHlens.com (http://adafru.it/aKt)

Εικόνα 3.14-Λειτουργία φακού Fresnel

Εδώ βλέπουμε τη λειτουργία του φακού σε συνδυασμό με τον αισθητήρα (Εικόνα 3.15)

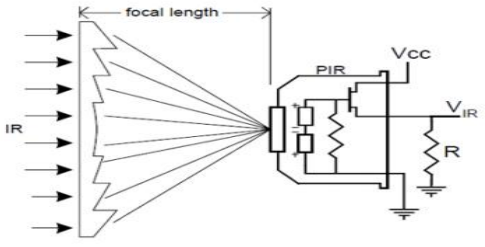


Image from Cypress appnote 2105 (http://adafru.it/cm6)

Εικόνα 3.15-Συγκεντρωση ακτινών από τον φακό στον αισθητήρα

Επομένως με αυτόν τον τρόπο, χρησιμοποιώντας δηλαδή φακό Fresnel, επιτυγχάνουμε μεγάλο εύρος. Ωστόσο έχουμε στην πράξη δύο αισθητήρες και πιο συγκεκριμένα δεν θέλουμε δυο μεγάλα ορθογώνια πεδία ανίχνευσης. Αυτό που επιθυμούμε είναι μια διασπορά, μια σκέδαση από πολλές μικρές περιοχές. Έτσι αυτό που κάνουμε είναι να χωρίσουμε τον φακό σε πολλές μικρές περιοχές, κάθε μια από τις οποίες είναι ένας φακός Fresnel. Τον διαχωρισμό αυτό μπορούμε να τον διακρίνουμε στην παρακάτω εικόνα (*Εικόνα 3.16*).

Εικόνα 3.16-Τμηματοποίηση Φακού

Στην επόμενη φωτογραφία (Εικόνα 3.17) φαίνεται ο κάθε φακός Fresnel σε κάθε πλευρά.

Εικόνα 3.17-Χωρισμός σε υπο-φακούς

Ο διαχωρισμός σε υπο-φακούς δημιουργεί μια σειρά από περιοχές ανίχνευσης που εναλλάσσονται η μια με την άλλη. Αυτός είναι λόγος που κάθε κέντρο του κάθε φακού βρίσκεται πάνω από διαφορετικό ήμισυ του αισθητηρίου.

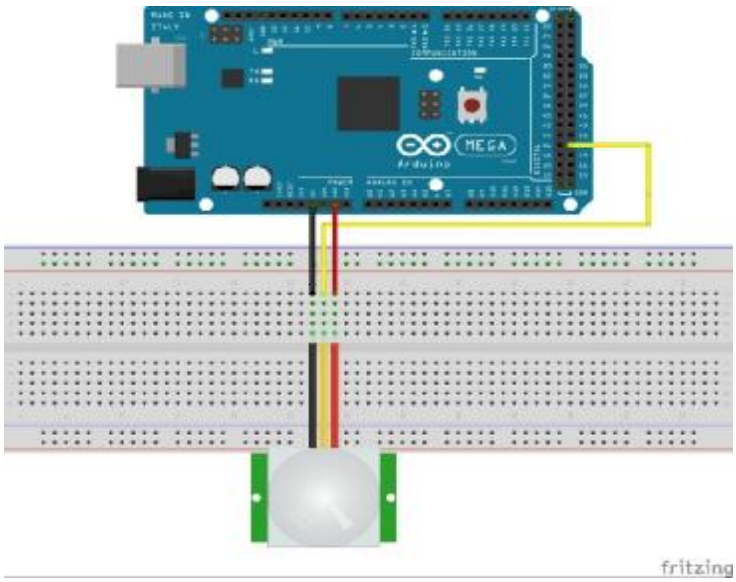
3.3.2.3 Συνδεσιμότητα και χρήση του PIR

Οι περισσότεροι PIR αισθητήρες έχουν τρία pin στο πλάι ή στο κάτω μέρος. Ένα pin για γείωση , ένα για το σήμα και ένα για τροφοδοσία η οποία είναι 3-5VDC.

Η σύνδεση του αισθητήρα με ένα μικροελεγκτή είναι πολύ απλή όπως βλέπουμε στην επόμενη εικόνα (Εικόνα 3.18) . Ο αισθητήρας συμπεριφέρεται σαν ψηφιακή

έξοδος. Έτσι το μόνο που έχουμε να κάνουμε είναι να ελέγχουμε πότε το Pin είναι HIGH(κίνηση) ή LOW(όχι κίνηση).

Στην περίπτωση του Arduino ο κώδικας είναι πολύ εύκολος, απλά ελέγχουμε την είσοδο του pin 47 στο Arduino Mega όπου έχουμε συνδέσει τον αισθητήρα και αναλόγως τυπώνουμε το αντίστοιχο μήνυμα.



Εικόνα 3.18-Κύκλωμα ελέγχου PIR

```
int MotionPin = 47;
void setup()
{
pinMode(13,OUTPUT);
Serial.begin(9600);
}
void loop()
{
int motion = digitalRead(MotionPin);

if (motion == HIGH)
  {
    digitalWrite (13, HIGH);
    Serial.println("Motion Detected!")
  }
  else
  {
    digitalWrite(13, LOW);
  }
}
```

3.3.3 Photo Resistor LDR

Αποτελεί έναν αισθητήρα φωτός. Είναι ένα φωτοκύτταρο (που ονομάζεται επίσης φωτοανιχνευτής, φωτοαντίσταση ή φωτοαγώγιμο κύτταρο) το οποίο ανάλογα με την ποσότητα φωτός που είναι εκτεθειμένο, μεταβάλει την αντίστασή του. Στην πράξη χρησιμοποιούνται για να κάνουν μια ενέργεια όταν αντιληφθούν φώς ή σκοτάδι. Είναι μικρός σε μέγεθος όπως βλέπουμε και στην επόμενη φωτογραφία (Εικόνα 3.19), φθηνός, χαμηλής ισχύος, εύκολος στη χρήση και δεν χαλάει εύκολα. Γι' αυτό το λόγο τον συναντάμε συχνά σε παιχνίδια, gadget και συσκευές.

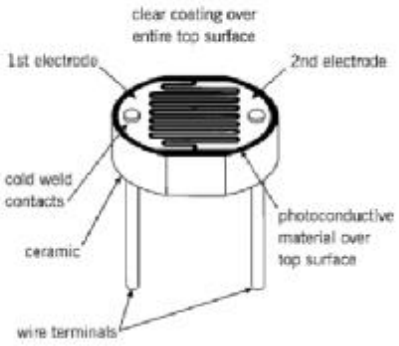
Εικόνα 3.19-Μεγεθος φωτοαντίστασης

Είναι κατά βάση μία αντίσταση που αλλάζει την τιμή της (σε Ω) ανάλογα με το πόσο φως λάμπει πάνω στην επιφάνειά της. Είναι εύκολο να αγοραστούν σε διάφορα μεγέθη και με διαφορετικά χαρακτηριστικά αλλά είναι πολύ ανακριβείς. Κάθε φωτοαντίσταση θα συμπεριφέρεται διαφορετικά από μία άλλη ακόμα κι αν είναι από την ίδια παρτίδα. Οι διακυμάνσεις μπορούν να ξεπεράσουν και το 50% σε κάποιες περιπτώσεις. Γι' αυτό το λόγο καλό είναι να αποφεύγεται η μέτρηση σε lux με αυτούς τους αισθητήρες και να χρησιμοποιούνται για γενικό προσδιορισμό μεταβολής φωτεινότητας.

Εικόνα 3.20-Φωτοαντίσταση

Η χρήση των αισθητήρων αυτών είναι καλή σε γενικά σενάρια όπως είναι ημέρα ή νύχτα; υπάρχει κάτι μπροστά από τον αισθητήρα που εμποδίζει τον φωτισμό;

Μία τυπική κατασκευή φωτοαντίστασης φαίνεται παρακάτω (Εικόνα 3.21):



Εικόνα 3.21-Κατασκευή φωτοαντίστασης

Χαρακτηριστικά του αισθητήρα που χρησιμοποιήσαμε

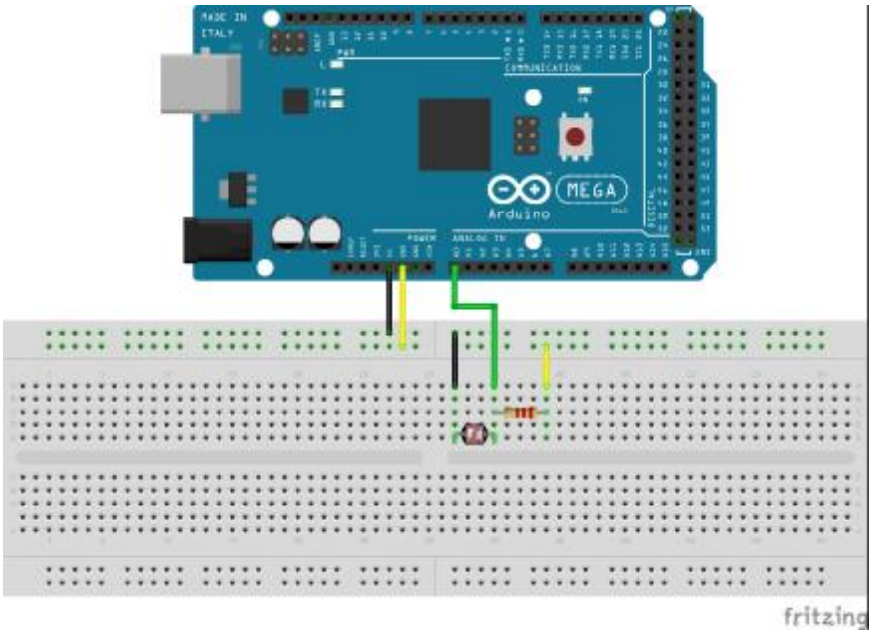
- Light resistance : 10k Ohm
- Dark resistance : 20k Ohm
- Max voltage : 150V
- Max power: 100mW

Διαστάσεις:

- 2 x 4 x 5mm
- 4mm between pins
- 31mm lead length

3.3.3.1 Σύνδεση Photo Resistor με το Arduino

Ο LDR όπως είπαμε αλλάζει την αντίστασή του με το φώς, έτσι μπορούμε να μετρήσουμε την αλλαγή αυτή χρησιμοποιώντας ένα αναλογικό pin του Arduino. Για να το κάνουμε αυτό όμως χρειαζόμαστε και μια σταθερή αντίσταση έτσι ώστε να μπορούμε να κάνουμε σύγκριση(στην περίπτωσή μας 10KΩ). Στην πράξη φτιάχνουμε ένα διαιρέτη τάσης ο οποίος διαιρεί τα 5Volt μεταξύ του αισθητήρα και της αντίστασης. Στη συνέχεια μετράμε πόση τάση βρίσκεται στα άκρα του αισθητήρα διαβάζοντας την αναλογική είσοδο του Arduino. Το Arduino συνεπώς θα διαβάσει 1024 διαφορετικές τιμές, με 1023 στα 5Volt και 0 στα 0Volt. Έτσι αν ο αισθητήρας και η αντίσταση έχουν την ίδια τιμή αντίστασης τότε τα 5Volt θα χωρίζονται ομοιόμορφα σε κάθε τμήμα.



Εικόνα 3.22-Συνδεσμολογία για έλεγχο φωτοαντίστασης

Ένας απλός κώδικας

Παρακάτω βλέπουμε έναν απλό κώδικα που μας τυπώνει τις τιμές του αισθητήρα

```
int LDR_Pin = A0; //analog pin 0

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

void loop(){
  int LDRReading = analogRead(LDR_Pin);

  Serial.println(LDRReading);
  delay(250);} //just here to slow down the output for easier reading
```

Έστω ότι θέλουμε να φτιάξουμε ένα σύστημα, το οποίο θα μετράει το πόσο έντονο είναι το φως σε ένα δωμάτιο, και αν η μέτρηση πέσει κάτω από κάποιο όριο να

ενεργοποιεί αυτόματα τα φώτα. Σε πρώτη φάση στο κύκλωμα θα ενεργοποιούμε ένα Led, ενώ στην συνέχεια θα μπορούσαμε να στέλνουμε trigger 5V σε ένα ac relay ώστε να ανάβει το φώτα σε ένα δωμάτιο. Το κύκλωμά μας δηλαδή θα είναι ως εξής:
Στην πιο απλή μορφή ο κώδικας που χρησιμοποιήσαμε στο Arduino μπορεί να εξηγηθεί ως εξής. Αρχικά δηλώνουμε τις μεταβλητές light_sensor και yellow_led στις οποίες βάζουμε το αναλογικό Pin 0, και το ψηφιακό PIN 13 του Arduino.

Η συνάρτηση void setup εκτελείται μία φορά κατά την εκκίνηση, ενώ η void loop εκτελείται κατ' επανάληψη.

Το Serial.begin(9600) ορίζει ότι θα χρησιμοποιήσουμε το Serial Monitor (κυρίως το θέλουμε για Debugging), με ταχύτητα 9600bps.

Στην συνέχεια στην μεταβλητή light_reading τύπου int, βάζουμε την μέτρηση που διαβάζουμε από τον αισθητήρα φωτός που είναι συνδεδεμένος στο αναλογικό Pin 0.

Ακόμα, ορίζουμε 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, βάζουμε να εμφανίζεται η τιμή της μεταβλητής light_reading στο Serial Monitor για Debugging.

```
int light_sensor = A0; //analogiko pin 0
int yellow_led = 13; //digital pin 13
void setup(){
  Serial.begin(9600); //serial gia debug
  pinMode(yellow_led, OUTPUT);
}
void loop(){
  int light_reading = analogRead(light_sensor);
  if (light_reading<600) {
    digitalWrite(yellow_led, HIGH); //anavw to led
  }
  else {
    digitalWrite(yellow_led,LOW); //svinw to led
  }
  Serial.println(light_reading); //serial gia debug
delay(300); //ena mikro delay gia kalitero output
}
```

3.3.4 Non-Invasive Current Sensor - 30A

Αυτός ο μη-επεμβατικός αισθητήρας ρεύματος (επίσης γνωστός ως "split core current transformer") μπορεί να σταθεροποιηθεί γύρω από τη γραμμή παροχής ενός ηλεκτρικού φορτίου και να διαβάσει πόσο ρεύμα περνάει μέσα από αυτό. Αυτό το επιτυγχάνει ενεργώντας ως επαγωγέας και ανταποκρινόμενος στο μαγνητικό πεδίο γύρω από τον αγωγό ρεύματος. Με την ανάγνωση του ρεύματος που παράγεται από το πηνίο, μπορείτε να υπολογίσετε πόσο ρεύμα διέρχεται από τον αγωγό.

Ο συγκεκριμένος αισθητήρας ρεύματος μπορεί να μετρήσει φορτίο έως 30 Amps που τον καθιστά ιδανικό για την κατασκευή μετρητή κατανάλωσης ρεύματος, ή ακόμα και τη δημιουργία μιας συσκευής προστασίας για υπερ-φόρτισης. Αυτός ο αισθητήρας δεν έχει ενσωματωμένη αντίσταση φορτίου, έτσι στις περισσότερες περιπτώσεις θα είναι αναγκαίο να τοποθετήσουμε έναν αντιστάτη σε ολόκληρη την έξοδο για να μετατρέψει το επαγόμενο ρεύμα του πηνίου σε μία πολύ μικρή μετρήσιμη τάση για το Arduino.

3.3.4.1 Πώς να κατασκευάσουμε έναν μετρητή τάσης και ρεύματος

Με τον παρακάτω οδηγό μπορούμε να φτιάξουμε έναν απλό μετρητή ο οποίος θα μετρά την τάση σε ένα μετασχηματιστή και το ρεύμα με τη βοήθεια ενός CT αισθητήρα. Θα μπορούμε να υπολογίσουμε στη συνέχεια με τη βοήθεια μιας βιβλιοθήκης στο Arduino την πραγματική ισχύ, φαινόμενη ισχύς, συντελεστής ισχύος, ενεργού τάσεως, ρεύματος RMS.

Τι θα χρειαστούμε;

Ηλεκτρονικά για τη μέτρηση τάσης:

- 1x 9V AC-AC Power Adapter
- 1x 100kOhm resistor for step down voltage divider.
- 1x 10kOhm resistor for step down voltage divider.
- 2x 470kOhm (for voltage divider, any matching value resistor pair down to 10K)
- 1x 10uF capacitor

Ηλεκτρονικά για μέτρηση ρεύματος:

- 1x CT sensor SCT-013-000
- 1x Burden resistor 18 Ohms if supply voltage is 3.3V or 33 Ohms if supply voltage is 5V.
- 2x 470kOhm (for voltage divider, any matching value resistor pair down to 10K)
- 1x 10uF capacitor

Οι μετασχηματιστές ρεύματος (CTS) είναι αισθητήρες που χρησιμοποιούνται για τη μέτρηση εναλλασσόμενου ρεύματος. Είναι ιδιαίτερα χρήσιμοι για τη μέτρηση της κατανάλωσης (ή παραγωγής) ηλεκτρικής ενέργειας σε ένα κτίριο.

Είναι κατάλληλοι για DIY(Do It Yourself) χρήση, καθώς μπορούν να τοποθετηθούν σε κάθε καλώδιο που μπαίνει μέσα στο κτίριο με κλιπ, χωρίς να χρειάζεται δηλαδή οποιαδήποτε ηλεκτρική εργασία.

Όπως μετασχηματιστής, έχει ένα πρωτεύον τύλιγμα, ένα μαγνητικό πυρήνα, και ένα δευτερεύον τύλιγμα.

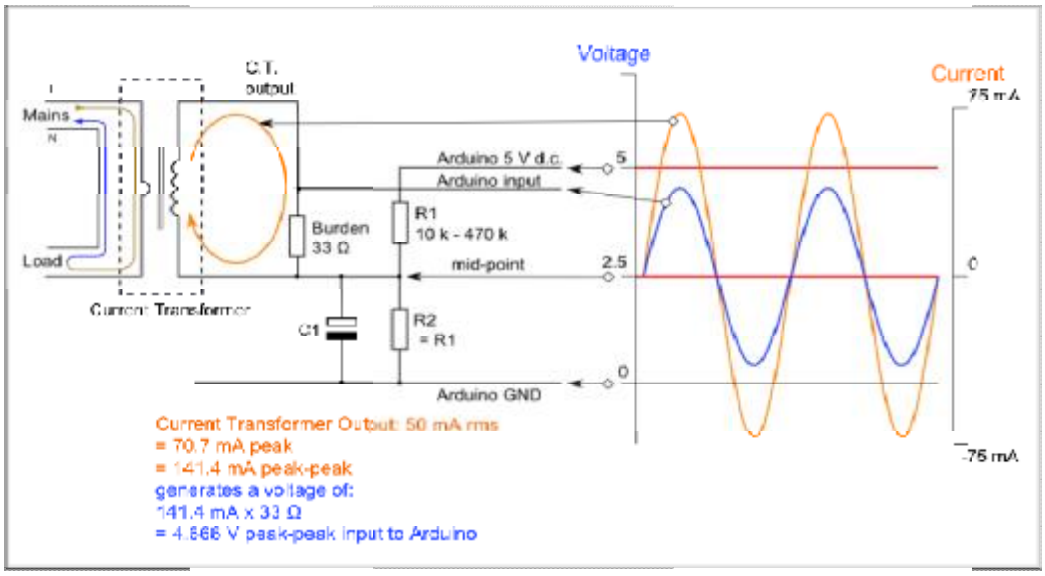
Στην περίπτωση παρακολούθησης κτιρίου, μετράμε τον ουδέτερο ή τη φάση (όχι και τα δύο!)

Το εναλλασσόμενο ρεύμα που ρέει στο πρωτεύον παράγει ένα μαγνητικό πεδίο στον πυρήνα, το οποίο στη συνέχεια προκαλεί ένα ρεύμα στο δευτερεύον τύλιγμα.

Το ρεύμα στο δευτερεύον τύλιγμα είναι ανάλογο προς την ροή ρεύματος στο πρωτεύον τύλιγμα

Για να μετρήσουμε το ρεύμα στο δευτερεύον πρέπει να χρησιμοποιήσουμε και μια αντίσταση φορτίου (η οποία στην ουσία κλείνει το κύκλωμα). Η τιμή της επιλέγεται ανάλογα με την τάση του δευτερεύοντος. Επίσης η τιμή της πρέπει να είναι αρκετά χαμηλή έτσι ώστε να μην προκληθεί κορεσμός του πυρήνα.

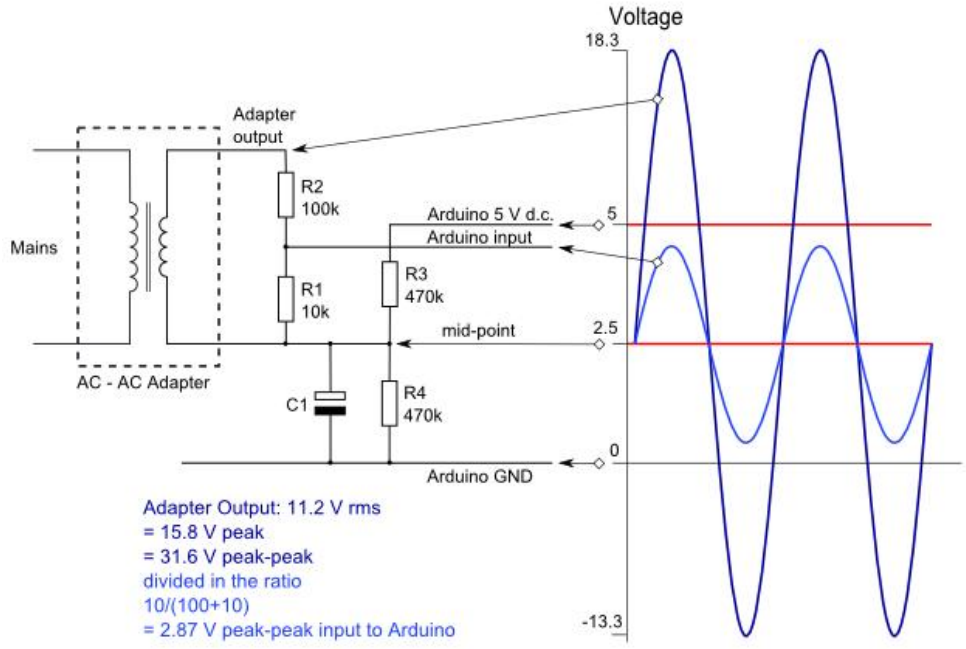
3.3.4.2 Σύνδεση του αισθητήρα με το Arduino



Εικόνα 3.23-Αναλυτικό κύκλωμα για τη μέτρηση ρεύματος από CT αισθητήρα

Αφού φτιάξουμε το παραπάνω κύκλωμα που στην ουσία πρόκειται για έναν διαιρέτη τάσης, με λίγες γραμμές κώδικα και με τη βοήθεια της βιβλιοθήκης EmonLib από της http://openenergymonitor.org/ (https://github.com/openenergymonitor/EmonLib) μπορούμε να μετρήσουμε το ρεύμα που διαρρέεται από έναν αγωγό.

Στη συνέχεια μπορούμε να προχωρήσουμε το κύκλωμά μας με σκοπό να μπορούμε να μετράμε και τάση.



Εικόνα 3.24-Αναλυτικό κύκλωμα για τη μέτρηση ρεύματος από το Arduino

Μια μέτρηση της τάσης εναλλασσόμενου ρεύματος είναι απαραίτητη για να υπολογιστούν η πραγματική ισχύς, η φαινόμενη ισχύς και ο συντελεστής ισχύος. Αυτή η μέτρηση μπορεί να γίνει με ασφάλεια με τη χρήση ενός AC-AC μετασχηματιστή. Ο μετασχηματιστής παρέχει απομόνωση μεταξύ της υψηλής και χαμηλής τάσης εναλλασσόμενου ρεύματος.

Για την ηλεκτρονική επεξεργασία του σήματος χρειάζεται να μετατρέψουμε την έξοδο του προσαρμογέα σε μια κυματομορφή που έχει μια θετική αιχμή που είναι μικρότερη από 5V (3.3V στην περίπτωση του emonTx) και μια αρνητική κορυφή που είναι περισσότερο από 0V και έτσι πρέπει να έχουμε κλίμακα κάτω από την κυματομορφή και να προσθέσουμε μια μετατόπιση έτσι ώστε να μην υπάρχει αρνητική συνιστώσα.

Η κυματομορφή μπορεί να μετακινηθεί προς τα κάτω χρησιμοποιώντας ένα διαιρέτη τάσης που συνδέεται στους ακροδέκτες του προσαρμογέα και η μετατόπιση μπορεί να προστεθεί χρησιμοποιώντας μια πηγή τάσης που δημιουργείται από έναν άλλο διαιρέτη τάσεως που συνδέεται με την τροφοδοσία του Arduino (με τον ίδιο τρόπο όπως έχουμε φτιάξει το κύκλωμα ανίχνευσης ρεύματος με τον CT).

Οι αντιστάσεις R1 και R2 σχηματίζουν τον διαιρέτη τάσης που μειώνει την ισχύ του μετασχηματιστή και οι αντιστάσεις R3 και R4 παρέχουν την τάση πόλωσης. Ο πυκνωτής C1 παρέχει μια διαδρομή χαμηλής αντίστασης προς τη γείωση για το σήμα AC.

Η τάση που παρέχεται από τις R3 και R4 πρέπει να είναι η μισή από την τάση τροφοδοσίας του Arduino γι' αυτό πρέπει να είναι ίσες.

Αν το Arduino λειτουργεί στα 5V η προκύπτουσα κυματομορφή του κυκλώματος έχει μια θετική κορυφή 2.5V + 1.15V = 3.65V και αρνητική κορυφή των 1.35V ικανοποιώντας τις απαιτήσεις τάσης των αναλογικών εισόδων του Arduino και αφήνοντας αρκετό χώρο ώστε να μην υπάρχει κανένας κίνδυνος μεγαλύτερων ή μικρότερων μη επιτρεπτών τάσεων.

Ο συνδυασμός 10k και 100k των R1 και R2 λειτουργεί επίσης ικανοποιητικά στα 3.3V όπως χρησιμοποιείται από την emonTx. Με μια θετική κορυφή του 2.8V και αρνητική κορυφή του 0.5V.

Ένας πολύ απλός κώδικας που μετρά τάση και ρεύμα και μας τυπώνει την πραγματική ισχύ, την φαινόμενη ισχύ, τον συντελεστή ισχύος, την ενεργό τάση και το ενεργό ρεύμα είναι ο εξής. Όλοι οι υπολογισμοί πραγματοποιούνται στην βιβλιοθήκη EmonLib.

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

void setup()
{
  Serial.begin(9600);

  emon1.voltage(2, 234.26, 1.7);  // Voltage: input pin, calibration, phase_shift
  emon1.current(1, 111.1);        // Current: input pin, calibration.
}

void loop()
{
  emon1.calcVI(20,2000);          // Calculate all. No.of crossings, time-out
  emon1.serialprint();}           // Print out all variables
```

3.3.5 Ανιχνευτής βροχής

Εικόνα 3.25-Αισθητήρας ανίχνευσης βροχής

Ο αισθητήρας βροχής είναι ένα εύκολο εργαλείο για την ανίχνευση βροχής. Μπορεί να χρησιμοποιηθεί σαν διακόπτης όταν σταγόνες βροχής πέσουν πάνω του ή ακόμα να μπορεί να μετρήσει και την ένταση της βροχής. Αποτελείται από μια πλακέτα βροχής και μια πλακέτα ελέγχου ξεχωριστά για ευκολία, με LED ένδειξής λειτουργίας και ποτενσιόμετρο ρύθμισης ευαισθησίας.

Η αναλογική έξοδος χρησιμοποιείται για να ανιχνεύει την ένταση της βροχής, ενώ η ψηφιακή το αν βρέχει ή όχι. Εάν συνδεθεί στα 5Volt, το LED θα ανάψει και εάν δεν υπάρχει βροχή η ψηφιακή έξοδος θα είναι HIGH αλλιώς θα είναι LOW εάν ρίξουμε λίγες σταγόνες νερού.

Χαρακτηριστικά

- Adopts high quality of RF-04 double sided material.
- Area: 5cm x 4cm nickel plate on side,
- Anti-oxidation, anti-conductivity, with long use time;
- Comparator output signal clean waveform is good, driving ability, over 15mA;
- Potentiometer adjust the sensitivity;
- Working voltage 5V;
- Output format: Digital switching output (0 and 1) and analog voltage output AO;
- With bolt holes for easy installation;
- Small board PCB size: 3.2cm x 1.4cm;
- Uses a wide voltage LM393 comparator

Διάταξη Pin

Στην επόμενη εικόνα (Εικόνα 3.26) βλέπουμε ον ελεγκτή του αισθητήρα βροχής και την διάταξη των pin

Εικόνα 3.26-Διαταξή Pin αισθητήρα

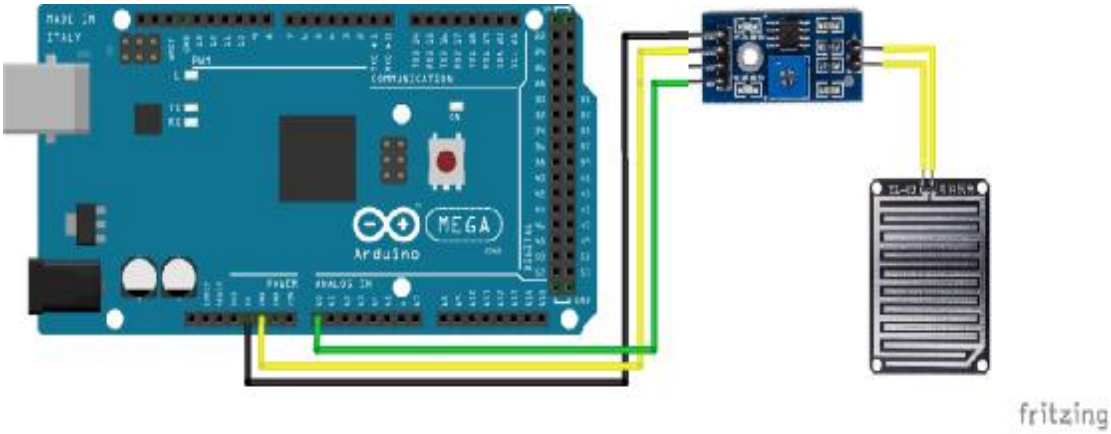
1. VCC: 5V DC

2. GND: ground

3. DO: high/low output

4. AO: analog output

Σύνδεση με το Arduino



Εικόνα 3.26-Συνδεσμολογία ελέγχου αισθητήρα βροχής

Ενδεικτικός κώδικας

```
void setup(){
 Serial.begin(9600);
 pinMode(2, OUTPUT);
}
void loop(){
 //analog output
 if(analogRead(0)<300) {
   Serial.println("Heavy Rain");
}
 else if(analogRead(0)<500) {
 Serial.println("Moderate Rain");}
 else {
Serial.println("No Rain");
}
// //digital output
// if(digitalRead(2) == HIGH) {Serial.println("No Rain
Detected");}
// else{ Serial.println("Rain Detected");}
 delay(250);
}
```

3.3.6 Bluetooth HC-06

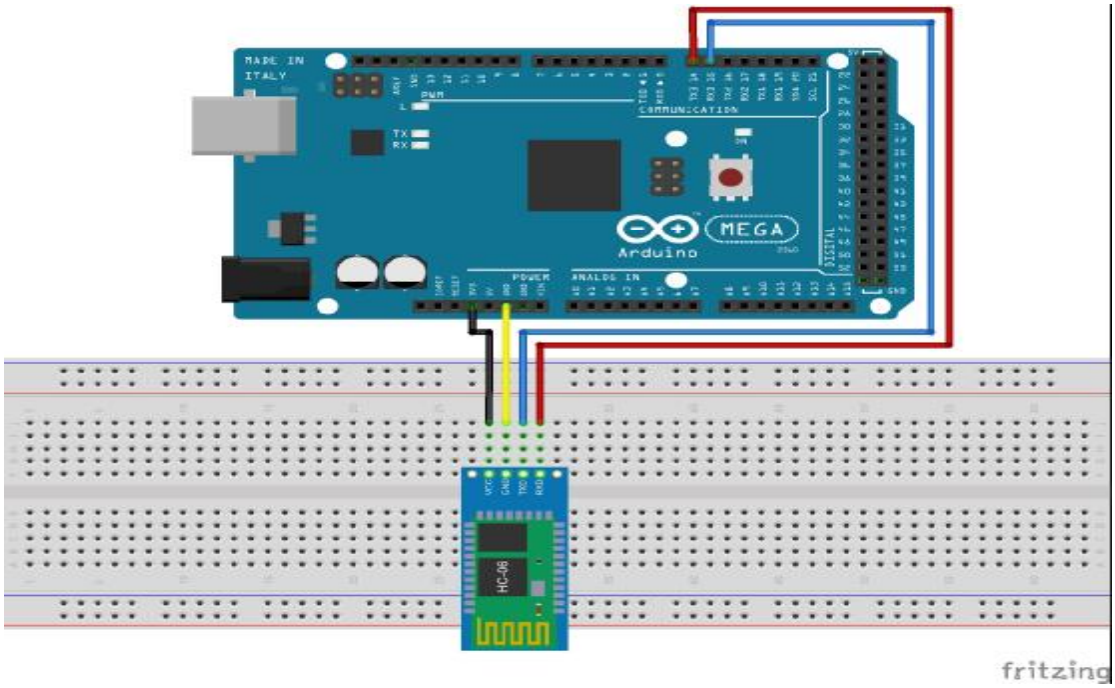
Οι πλακέτες Bluetooth HC χρησιμοποιούνται με σκοπό να μετατρέψουν την σειριακή θύρα σε Bluetooth. Μπορούν να λειτουργήσουν και σαν Master και σαν Slave συσκευές.

Χαρακτηριστικά:

- TTL data transparent transfer between a host Bluetooth device.
- Works with any USB Bluetooth adapters.
- Default Baud Rate: 9600 (8 N 1)
- Range up to 30ft.

- Built in antenna.
- Power input: 3.6-5.5V
- Mini Size.
- Dimensions: 1.73 in x 0.63 in x 0.28 in (4.4 cm x 1.6 cm x 0.7 cm)
- Weight: 0.25 oz (7 g)

Σύνδεση με το Arduino



Εικόνα 3.27-Συνδεσμολογία HC 06 με Arduino

Στο παρόν project στόχος ήταν να μπορούμε να δίνουμε φωνητικές εντολές στο Arduino χρησιμοποιώντας το android κινητό μας. Για να τα καταφέρουμε εκμεταλλευόμαστε την αναγνώριση φωνής που διαθέτει κάθε android κινητό και μπορούμε να τη χρησιμοποιήσουμε για να ελέγχουμε λειτουργίες του Arduino μέσω Bluetooth.Η εφαρμογή που χρησιμοποιούμε είναι αρκετά απλή και τη βρίσκουμε εδώ https://play.google.com/store/apps/details?id=robotspace.simplelabs.amr_voice,
αφού πατήσουμε το κουμπί του μικροφώνου, το τελευταίο περιμένει να του δώσουμε την εντολή μας. Στη συνέχεια η εφαρμογή μας γράφει τις λέξεις που είπαμε και τις στέλνει μετατρέποντάς τες σε string έτσι ώστε να μπορεί το Arduino να τις επεξεργαστεί.

```
String voice;

int
led1 = 2, //Connect LED 1 To Pin #2
led2 = 3, //Connect LED 2 To Pin #3
//-------------------------Call A Function-----------------------------//
void allon(){
     digitalWrite(led1, HIGH);
     digitalWrite(led2, HIGH);
}
void alloff(){
     digitalWrite(led1, LOW);
     digitalWrite(led2, LOW);
}
//---------------------------------------------------------------------//
void setup() {
  Serial.begin(9600);
```

```
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}
//-------------------------------------------------------------------------//
void loop() {
  while (Serial.available()){  //Check if there is an available byte to read
  delay(10); //Delay added to make thing stable
  char c = Serial.read(); //Conduct a serial read
  if (c == '#') {break;} //Exit the loop when the # is detected after the word
  voice += c; //Shorthand for voice = voice + c
  }
  if (voice.length() > 0) {
    Serial.println(voice);
//-------------------------------------------------------------------------//
  //----------Control Multiple Pins/ LEDs----------//
       if(voice == "*all on") {allon();}  //Turn Off All Pins (Call Function)
  else if(voice == "*all off"){alloff();} //Turn On  All Pins (Call Function)

  //----------Turn On One-By-One----------//
  else if(voice == "*TV on") {digitalWrite(led1, HIGH);}
  else if(voice == "*fan on") {digitalWrite(led2, HIGH);}
  //----------Turn Off One-By-One----------//
  else if(voice == "*TV off") {digitalWrite(led1, LOW);}
  else if(voice == "*fan off") {digitalWrite(led2, LOW);}
 //-------------------------------------------------------------------------//
voice="";}} //Reset the variable after initiating
```

3.3.7 Οθόνη LCD IIC/TWI 20x4

Η οθόνη LCD μπορεί να δώσει στον χρήστη άμεσα αποτελέσματα σχετικά με τις μετρήσεις των αισθητήρων και τις καταστάσεις των διαφόρων συσκευών στο σπίτι. Επίσης ο ίδιος ο χρήστης μπορεί να επέμβει κάνοντας χειρισμούς και να αλλάξει διάφορες παραμέτρους και καταστάσεις. Προγραμματίζεται με απλές εντολές από το Arduino. Η οθόνη που χρησιμοποιούμε διαφέρει κάπως από τις απλές διότι δεν είναι απλό shield που να κουμπώνει πάνω στο Arduino. Ούτε χρειάζεται πολλά pin για να λειτουργήσει. Εφαρμόζει το πρωτόκολλο επικοινωνίας IIC και χρησιμοποιεί 4 pin. 5V για τροφοδοσία, GRD για γείωση και SPI και SCL για σειριακή επικοινωνία. Διαθέτει ποτενσιόμετρο για ρύθμιση φωτεινότητας και για να λειτουργήσει σωστά χρειάζεται δυο βιβλιοθήκες. Την Wire.h για την επικοινωνία με I2C και την LiquidCrystal_I2C.h για να αναγνωρίσει το Arduino τα αντίστοιχα Pin και τις αντίστοιχες εντολές.

Χαρακτηριστικά:

Interface	I2C
I2C Address	0x27
Pin Definition	GND,VCC,SDA,SCL
Back lit	Blue with White char color
Supply voltage	5V
Size	60mm×99mm
Contrast Adjust	Potentiometer
Backlight	Jumper

Άρα όσον αφορά τις βιβλιοθήκες αρκεί να προσθέσουμε τις εντολές

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

καθώς επίσης και να δηλώσουμε τα pin της οθόνης

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

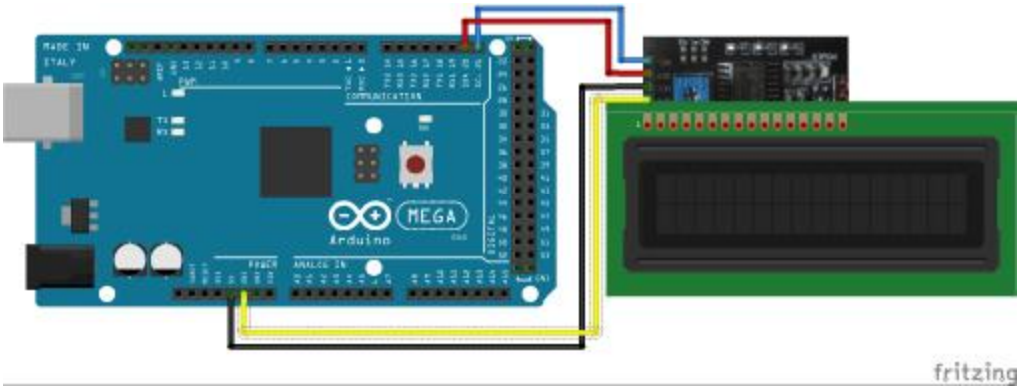
Στο `void setup()`

```
lcd.init();                      // initialize the lcd
lcd.backlight();
```

```
lcd.clear();
```

Μετά από αυτές τις ρυθμίσεις η οθόνη LCD είναι έτοιμη να τυπώσει ότι επιθυμούμε.

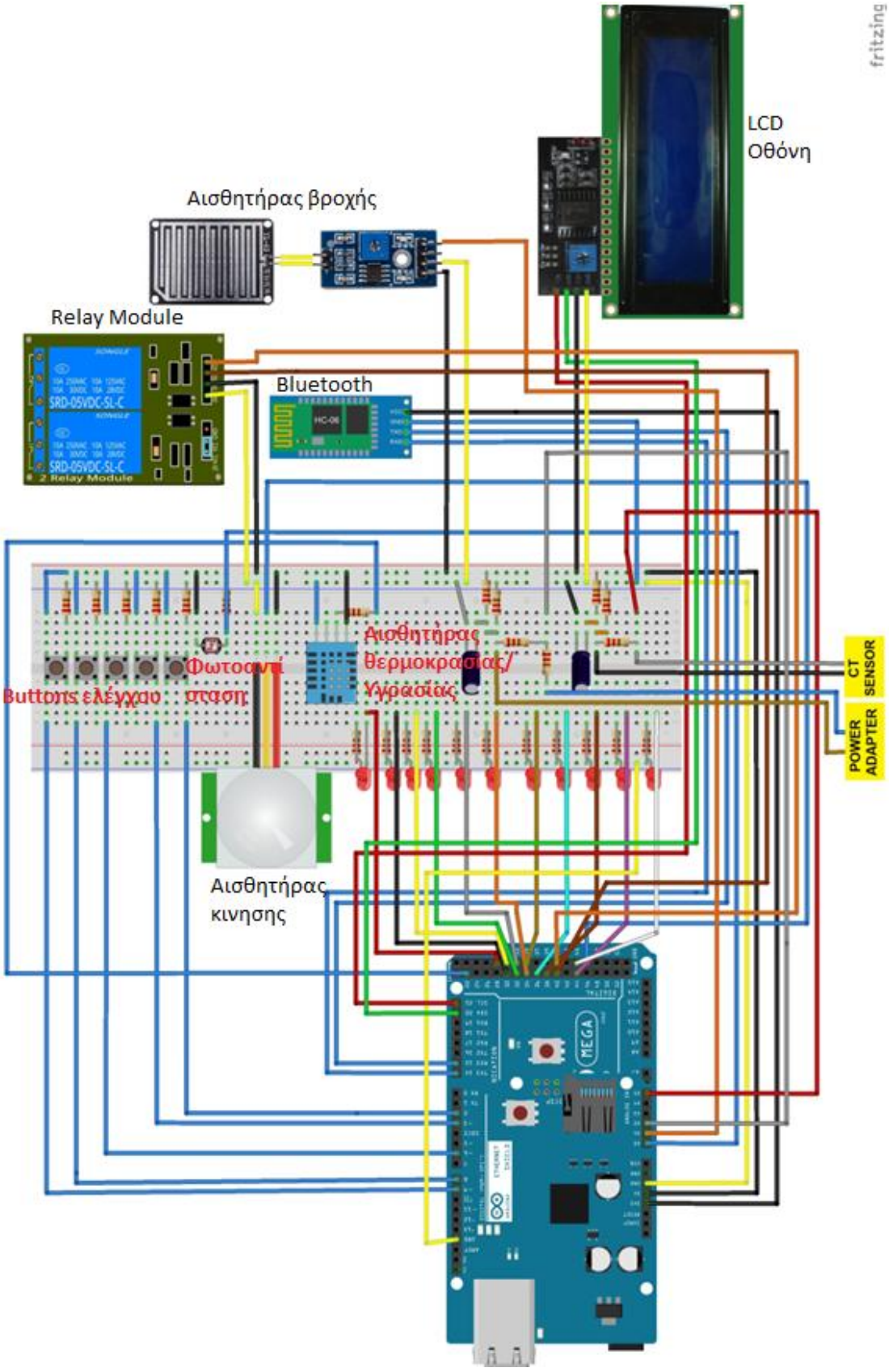
Σύνδεση με το Arduino



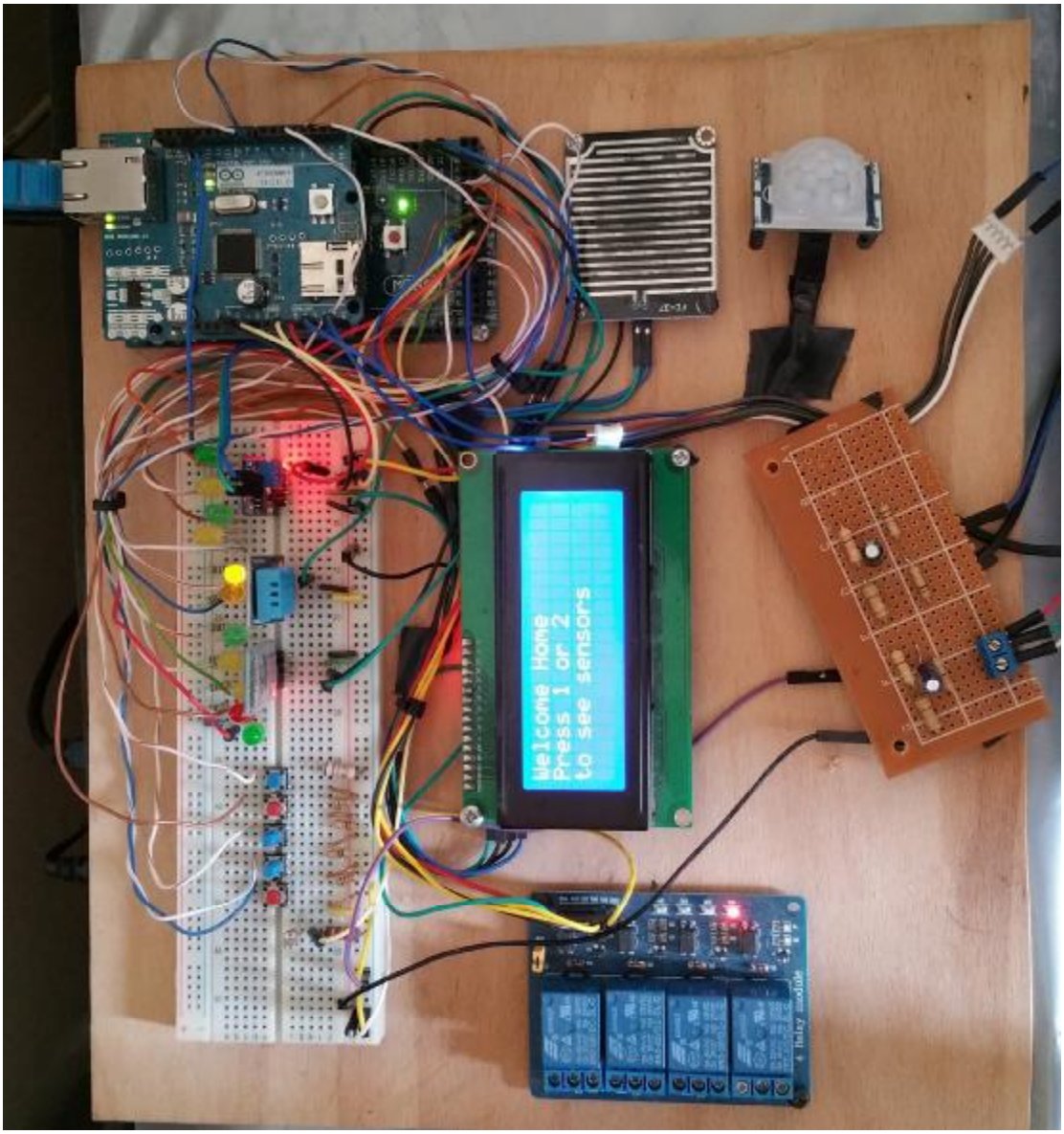
Εικόνα 3.28-Συνδοσμολογία οθόνης LCD με I2C Interface

3.4 Κύκλωμα

Το κύκλωμα που υλοποιήσαμε αποτελείται από ένα Arduino Mega για ελεγχόμενα I/O, ένα Ethernet Shield για επικοινωνία με το διαδίκτυο, όλους τους αισθητήρες που αναφέραμε στις προηγούμενες ενότητες καθώς και έναν αριθμό από leds ως εξόδους για να έχουμε ορατά αποτελέσματα των αυτοματισμών και των σεναρίων που υλοποιήσαμε αλλά και σαν δόκιμη πριν συνδέσουμε πραγματικές συσκευές. Επίσης διαθέτει πέντε Buttons που χειρίζονται leds αλλά και το menu της οθόνης. Στις προηγούμενες ενότητες είδαμε κάθε ένα αισθητήρα πως συνδέθηκε στο Arduino. Ας δούμε τώρα το τελικό μας κύκλωμα (Εικόνα 3.29) και το πραγματικό κύκλωμα στην (Εικόνα 3.30).



Εικόνα 3.29-Σχέδιο τελικού κυκλώματος



Εικόνα 3.30-Τελικό υλοποιημένο κύκλωμα σε λειτουργία

ΚΕΦΑΛΑΙΟ 4: Λογισμικό Μέρος Συστήματος

4.1 Προγραμματισμός Arduino

4.1.1 **Καταχώρηση βιβλιοθηκών**

Στην αρχή του προγράμματος γίνεται πρώτα απ' όλα η εισαγωγή των βιβλιοθηκών. Οι βιβλιοθήκες του κώδικά μας αφορούν το Ethernet Shield, τον αισθητήρα DHT11, την οθόνη και τη μέτρηση ηλεκτρικών μεγεθών. Επίσης προσθέτουμε βιβλιοθήκες και για την κάρτα SD. Οι βιβλιοθήκες που προσθέτουμε είναι οι εξής:

SPI	Χρησιμοποιείται για την σειριακή επικοινωνία μεταξύ περιφερειακών συσκευών. Στην περίπτωσή μας του Ethernet Shield με το Arduino ως Master.
Ethernet	Επιτρέπει τη σύνδεση του Arduino με το Internet. Είτε αποδέχοντας συνέσεις ως server είτε αποστέλοντας ως client.
Wire	Επιτρέπει την επικοινωνία με συσκευές που υποστηρίζουν το πρωτόκολλο i2c.
DHT	Είναι η βιβλιοθήκη για την αναγνώριση του αισθητήρα θερμοκρασίας και υγρασίας.
EmonLib	Πρόκειται για τη βιβλιοθήκη μέτρησης καταναλώσεων που χρησιμοποιήσαμε.
SD	Επιτρέπει την εγγραφή και την ανάγνωση από την κάρτα μνήμης SD
LiquidCrystal2C	Είναι η βιβλιοθήκη της LCD οθόνης με πρωτόκολλο επικοινωνίας I2C

Για να εισάγουμε τις βιβλιοθήκες αρκεί να γράψουμε στην αρχή του κώδικα τις εξής εντολές με τις αντίστοιχες δηλώσεις:

```
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <Wire.h>
#include "DHT.h"
#define DHTPIN 22    // Σε ποιό Pin έχουμε συνδέσει τον αισθητήρα
#define DHTTYPE DHT11   // Δήλωση πως έχουμε τον DHT 11
DHT dht(DHTPIN, DHTTYPE);
#include "EmonLib.h"             // Include Emon Library
EnergyMonitor emon1;             // Create an instance
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line
display
```

4.1.2 **Δηλώσεις** Ethernet, **αρχικοποιήσεις μεταβλητών και ονοματοδοσία** Pin

Στη συνέχεια δηλώνουμε τη διεύθυνση IP του server της βάσης δεδομένων(του υπολογιστή μας), την IP του server της ιστοσελίδας(δηλαδή του Arduino) και MAC Address του Arduino. Επίσης αρχικοποιούμε ένα πίνακα `LED_state[4] = {0};` ο

οποίος αποθηκεύει την κατάσταση των led που ελέγχει η ιστοσελίδα καθώς επίσης ορίζουμε και ένα buffer στο οποίο σώζουμε τα HTTP requests.

```
// MAC address from Ethernet shield sticker under board
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetClient client;
char server[] = "192.168.1.22";
IPAddress ip(192, 168, 1, 250); // IP address, may need to change depending on network
EthernetServer server1(80);  // create a server at port 80
File webFile;               // the web page file on the SD card
#define REQ_BUF_SZ   60
char HTTP_req[REQ_BUF_SZ] = {0}; // buffered HTTP request stored as null terminated string
char req_index = 0;              // index into HTTP_req buffer
boolean LED_state[4] = {0}; // stores the states of the LEDs
int  interval = 500; // Wait between dumps
```

Έπειτα δίνουμε ονόματα στα pin πράγμα που θα μας βοηθήσει στη συνέχεια του κώδικα να θυμόμαστε τι κάνει κάθε pin. Επίσης αρχικοποιούμε κάποιες μεταβλητές και κάποιους μετρητές που θα χρειαστούμε στη συνέχεια

```
int MotionPin = 47;
int pirState = LOW;             // we start, assuming no motion detected
int val = 0;
int light_sensor = A0;
String voice;
int button = 2;
int button1 = 3;
int button2 = 5;
int button3 = 8;
int button4 = 9;
int buttonState = 0;
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int buttonState4 = 0;
int ledstate = 0;
int garagestate = 0;
int kitchenstate = 0;
int clima = 30;
int outlight = 31;
int windows = 32;
int sirina = 40;
int led = 35;
int garage = 33;
int kitchen = 34;
int potisma = 36;
int siskeyi = 37;
int templcd = 0;
char g;
float wh = 0;
String buffer;
int  Wait_between_dumps = 0;
```

4.1.3 **Προσδιορισμός I/O**

Το `void setup()`όπως έχουμε αναφέρει και σε προηγούμενο κεφάλαιο θα τρέξει μόνο μια φορά τις εντολές του. Δηλώνουμε ποιά pin είναι είσοδοι, ποιά έξοδοι και δίνουμε εντολές για να ξεκινήσει να λειτουργεί η σειριακή θύρα και η ανάγνωση της κάρτας SD, απενεργοποιώντας το W5100 θέτοντας το pin 10 HIGH.

```
void setup()
{
  emon1.voltage(3, 13.5, 1.7);  // Voltage: input pin, calibration, phase_shift
  emon1.current(2, 59);       // Current: input pin, calibration.
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);  //Apenergopoiisi W5100
  Serial.begin(9600); //Xekinima Seiriakwn gia Debagging kai epimerous leitoureiwn
  Serial3.begin(9600);
  SD.begin(4);  // Xekinima Anagnwsis SD
  pinMode(2, INPUT); //Dilwseis Eisodwn
  pinMode(3, INPUT);
  pinMode(5, INPUT);
  pinMode(8, INPUT);
  pinMode(9, INPUT);                      //Dilwseis Exodwn// LEDs
  pinMode(33, OUTPUT);
  pinMode(34, OUTPUT);
  pinMode(35, OUTPUT);
  pinMode(36, OUTPUT);
```

```
  pinMode(37, OUTPUT);
  pinMode(38, OUTPUT);
```

Δίνουμε εντολή να ξεκινήσει η αναζήτηση για clients

```
  Ethernet.begin(mac);
  Ethernet.begin(mac, ip);  // initialize Ethernet device
  server1.begin();           // start to listen for clients
```

Προετοιμάζουμε την οθόνη

```
  lcd.init();                      // initialize the lcd
  lcd.backlight();
  lcd.clear();
  }
////////////////////////////////////////////////////////////////////////////////
```

Το void loop επαναλαμβάνεται από την αρχή μέχρι το τέλος για όσο υπάρχει τροφοδοσία στο Arduino. Στην αρχή ελέγχουμε έχει πατηθεί κάποιο button. Εισάγουμε ένα μικρό delay έτσι ώστε εφόσον πατήσουμε κάποιο button να προλάβει το Arduino διαβάσει και να αποθηκεύσει την κατάσταση πριν αυτή αλλάξει.

```
void loop()
{
  //////Diabazw tin katastasi twn Button//////////
  buttonState = digitalRead(button);
  delay(10);
  buttonState1 = digitalRead(button1);
  delay(10);
  buttonState2 = digitalRead(button2);
  delay(10);
  buttonState3 = digitalRead(button3);
  delay(10);
  buttonState4 = digitalRead(button4);
  delay(10);
```

4.1.4 **Μετρήσεις αισθητήρων**

Εδώ αποθηκεύουμε σε μεταβλητές τις μετρήσεις των αισθητήρων. Οι τιμές της πραγματικής και της φαινόμενης ισχύος καθώς και ο συντελεστής ισχύος υπολογίζεται από τη βιβλιοθήκη emonlib. Επίσης από τη μέτρηση της πραγματικής ισχύος υπολογίζουμε τις Wh που καταναλώνει ένα φορτίο. Το ρεύμα μετράται από τον αισθητήρα ρεύματος ενώ η τάση από το Arduino. Στον παρακάτω κώδικα βλέπουμε και τις υπόλοιπες μετρήσεις πως δηλώνονται και αποθηκεύονται στις αντίστοιχες μεταβλητές για να χρησιμοποιηθούν στον υπόλοιπο κώδικα.

```
  emon1.calcVI(20, 2000);
  float realPower       = emon1.realPower;        //extract Real Power into variable
  float apparentPower   = emon1.apparentPower;    //extract Apparent Power into
variable
  float powerFActor     = emon1.powerFactor;      //extract Power Factor into Variable
  float supplyVoltage   = emon1.Vrms;             //extract Vrms into Variable
  float Irms            = emon1.Irms;             //extract Irms into Variable
  float h = dht.readHumidity();                   //μεταβλητή υγρασίας
  float t = dht.readTemperature();                //μεταβλητή θερμοκρασίας
  float light_reading = analogRead(A0);           //μεταβλητή φωτοαντίστασης
  float rain = analogRead(A1);                    //μεταβλητή αισθητήρα βροχής
  int motion = digitalRead(MotionPin);            //μεταβλητή αισθητήρα κίνησης
  wh = wh + ((realPower / 7200));                 //υπολογισμός κατανάλωσης σε Wh
  ledstate = digitalRead(led);                    //αρχικοποιήσεις καταστάσεων led
  garagestate = digitalRead(garage);
  kitchenstate = digitalRead(kitchen);
```

4.1.5 **Σενάρια αυτοματισμών**

Στη συνέχεια εισάγουμε κάποια σενάρια αυτοματισμών με βάσει τις μετρήσεις των αισθητήρων.

- **Αισθητήρας κίνησης:** Όταν ανιχνευτεί κίνηση ενεργοποιείται μία σειρήνα.
- **Αισθητήρας βροχής:** Όταν παρατηρηθεί βροχή κλείνουν αυτόματα τα παράθυρα και σταματά το αυτόματο σύστημα ποτίσματος.
- **Φωτοαντίσταση:** Μετρώντας φωτεινότητα κάτω από ένα όριο ενεργοποιείται ο εξωτερικός φωτισμός.

- **Αισθητήρας ρεύματος:** Λειτουργεί σαν ηλεκτρονόμος προστασίας από υπερένταση. Όταν ανιχνεύσει συγκεκριμένα ρεύμα πάνω από 4Α κλείνει μια συσκευή.
- **Αισθητήρας θερμοκρασίας**: Όταν μετρήσει θερμοκρασία πάνω από 27oC ενεργοποιείται ο κλιματισμός

Ας τα δούμε από άποψη προγραμματισμού:

Χρησιμοποιούμε τον αισθητήρα ρεύματος σαν ηλεκτρονόμο και έτσι όταν αναγνωρίσει ρεύμα πάνω από 4 Α τότε κλείνει αυτόματα μια συσκευή.

```
if (emon1.Irms < 4)
  {
    digitalWrite(siskeyi, HIGH);
    // digitalWrite(41, LOW);
  }
  else if (emon1.Irms > 4)
  {
    digitalWrite(siskeyi, LOW);
    // digitalWrite(41, HIGH);
  }
```

Εάν τώρα ο αισθητήρας θερμοκρασίας μετρήσει θερμοκρασία πάνω από 27^{o}C ενεργοποιείται μία έξοδος που θεωρούμε πως είναι ένα κλιματιστικό.

```
  if (t > 27)


  {
    digitalWrite(clima, HIGH);
  }
  else if (t < 27)
  {
    digitalWrite(clima, LOW);
  }
```

Έπειτα εάν μετρηθεί φωτεινότητα κάτω από μία τιμή δίνουμε εντολή να ανοίξει αυτόματα ο εξωτερικός φωτισμός του σπιτιού.

```
if (light_reading  < 600)
  {
    digitalWrite(outlight, HIGH); //anavw to led
  }
  else if (light_reading  > 600)
 {
    digitalWrite(outlight, LOW); //svinw to led
  }
```

Στη συνέχεια ένα άλλο σενάριο είναι σε περίπτωση βροχής να κλείνουν αυτόματα τα παράθυρα και να σταματάει το σύστημα ποτίσματος.

```
 if (rain < 300)
  {
    digitalWrite(windows, HIGH);
    digitalWrite(potisma, HIGH);
  }
  else
  {
    digitalWrite(windows, LOW);
    digitalWrite(potisma, LOW);
  }
```

Τέλος σε περίπτωση ανίχνευσης κίνησης ενεργοποιούμε ένα led που στην πράξη θα μπορούσε να είναι μια σειρήνα.

```
  if (digitalRead(MotionPin) == HIGH)
  {
    digitalWrite(sirina, HIGH);  // turn LED ON
  }
  else
  {
    digitalWrite(sirina, LOW);
  }
```

Μετά από τη σύνδεση και τις ρυθμίσεις της οθόνης LCD που είπαμε σε προηγούμενο κεφάλαιο, είμαστε έτοιμοι να την προγραμματίσουμε να τυπώσει ότι επιθυμούμε.

4.1.6 **Έλεγχος από οθόνη** LCD

Στα πλαίσια της πτυχιακής φτιάξαμε ένα menu το οποίο στην αρχική οθόνη χαιρετίζει τον χρήστη και στη συνέχεια με τη χρήση δυο button για back και next αλλάζουν καρτέλες με τις τρέχουσες μετρήσεις των αισθητήρων. Τέλος δίνεται η δυνατότητα να αλλάξει η κατάσταση ενός led, στην πράξη θα μπορούσε να είναι οποιαδήποτε συσκευή. Η λογική του menu είναι πως με τη βοήθεια δύο button αυξάνουμε και μειώνουμε μια μεταβλητή μέσα σε συγκεκριμένα όρια. Η μεταβλητή αυτή μπορεί να πάρει τιμές όσες και οι καρτέλες του μενού που θέλουμε να φτιάξουμε. Με εντολές if..else ανάλογα, καθαρίζουμε την οθόνη και τυπώνουμε τις μετρήσεις μας. Τέλος μας δίνει τη δυνατότητα να ανοίξουμε ένα led και από το menu αυτό, πέρα από τους άλλους τρόπους.

```
if (buttonState3 == HIGH)
{
  templcd++;
}
if (buttonState4 == HIGH)
{
  templcd--;
}
```

```
if (templcd == 0)

{
  lcd.clear();
  lcd.print("Welcome Home");
  lcd.setCursor(0, 1);
  lcd.print("Press 1 or 2");
  lcd.setCursor(0, 2);
  lcd.print("to see sensors");
}
```

Εικόνα 4.1-Αρχική καρτέλα οθόνης

```
else if (templcd == 1)
  {
    lcd.clear();
    lcd.print("Irms:");
    lcd.print(Irms);
    lcd.print("A");
    lcd.setCursor(0, 1);
    lcd.print("Vrms:");
    lcd.print(emon1.Vrms);
    lcd.print("V");
    lcd.setCursor(0, 2);
    lcd.print("Wh:");
    lcd.print(wh);
    }
```

Εικόνα 4.2-Δεύτερη καρτέλα menu οθόνης

```
else if (templcd == 2)
  {
    lcd.clear();
    lcd.print("cos");
    lcd.print(powerFActor);
    lcd.setCursor(0, 1);
    lcd.print("P:");
    lcd.print(realPower);
    lcd.print("W");
    lcd.setCursor(0, 2);
    lcd.print("S:");
    lcd.print(apparentPower);
    lcd.print("VA");
  }
```

Εικόνα 4.2-Τρίτη καρτέλα menu οθόνης

```
  else if (templcd == 3)

  {
    lcd.clear();
    lcd.print("Room Temp:");
    lcd.print(t);
    lcd.print("oC");
    lcd.setCursor(0, 1);
    lcd.print("Room Humid:");
    lcd.print(h);
    lcd.print("%");
    lcd.setCursor(0, 2);
    lcd.print("Room Mov:");
    lcd.print(motion);
  }
```

Εικόνα 4.3-Τέταρτη καρτέλα menu οθόνης

```
  else if (templcd == 4)
  {
    lcd.clear();
    lcd.print("LED STATE:");
    lcd.print(ledstate);
    lcd.setCursor(0, 1);
    lcd.print("Press button to");
    lcd.setCursor(0, 2);
    lcd.print("change led state");
    if (buttonState2 == HIGH)
    {
      ledstate = digitalRead(led);
      ledstate =
change_status(ledstate);
      digitalWrite(led, ledstate);
    }
  }
```

Εικόνα 4.4-Πέμπτη καρτέλα menu οθόνης

```
  else if (templcd > 4 || templcd < 0)
  {
    templcd = 0;
  }
```

Επειδή μπορεί να πατήσουμε τα buttons παραπάνω φορές από τον αριθμό των καρτελών του menu, ορίζουμε αυτή τη συνθήκη έτσι ώστε ο μετρητής πάνω από την τιμή 4 ή κάτω από την τιμή 0 να επιστρέφει στην αρχική οθόνη

4.1.7 Έλεγχος σειριακών θηρών

Ελέγχω τη σειριακή θύρα 3 του Arduino Mega για strings που προέρχονται από τις φωνητικές εντολές που δίνουμε μέσω του κινητού μας

```
while (Serial3.available()) { //Check if there is an available byte to read
  delay(10); //Delay added to make thing stable
  char k = Serial3.read(); //Conduct a serial read
  if (k == '#') {
    break; //Exit the loop when the # is detected after the word
  }
  voice += k; //Shorthand for voice = voice + c
}
```

Ελέγχω τη σειριακή θύρα 1 του Arduino Mega για strings που προέρχονται από το LabVIEW. Αν το τελευταίο γράψει στη σειριακή θύρα το string "sensors" τότε το Arduino γράφει στη σειριακή θύρα μια της μετρήσεις των αισθητήρων.

```
if (Serial.available() > 0)
{
  delay(100);
  while (Serial.available() > 0)
  {

    g = Serial.read();
    buffer += g;
  }
  if (buffer == "sensors\n")
  {
    Serial.println(emon1.Irms);
    Serial.println(emon1.Vrms);
    Serial.println(emon1.powerFactor);
    Serial.println(emon1.apparentPower);
    Serial.println(emon1.realPower);
    Serial.println(h);
    Serial.println(t);
    Serial.println(light_reading);
    Serial.println(rain);
    Serial.println(digitalRead(MotionPin));
    Serial.println(wh);
    buffer = "";
  }
```

Στη συνέχεια ελέγχω εάν το LabVIEW στέλνει στο Arduino το string led. Κάθε φορά που ο buffer έχει το string "led" καλείται μια υπορουτίνα που διαβάζει την τελευταία κατάσταση του led και την αλλάζει.

```
  else if (buffer == "led\n")
  {
    // Serial.println("1");
    ledstate = digitalRead(led);
    ledstate = change_status(ledstate);
    digitalWrite(led, ledstate);
    buffer = "";
```

```
    }
```

Ομοίως λειτουργούμε για άλλα led-συσκευές.

```
    else if (buffer == "garage\n")
    {
      // Serial.println("1");
      garagestate = digitalRead(garage);
      garagestate = change_status(garagestate);
      digitalWrite(garage, garagestate);
      buffer = "";
    }
    else if (buffer == "kitchen\n")
    {
      // Serial.println("1");
      kitchenstate = digitalRead(kitchen);
      kitchenstate = change_status(kitchenstate);
      digitalWrite(kitchen, kitchenstate);
      buffer = "";
    }
```

Εάν ο Buffer έχει το string ledstatus το Arduino τυπώνει όλες τις καταστάσεις των διαθέσιμων led έτσι ώστε να διαβαστούν από το LabVIEW.

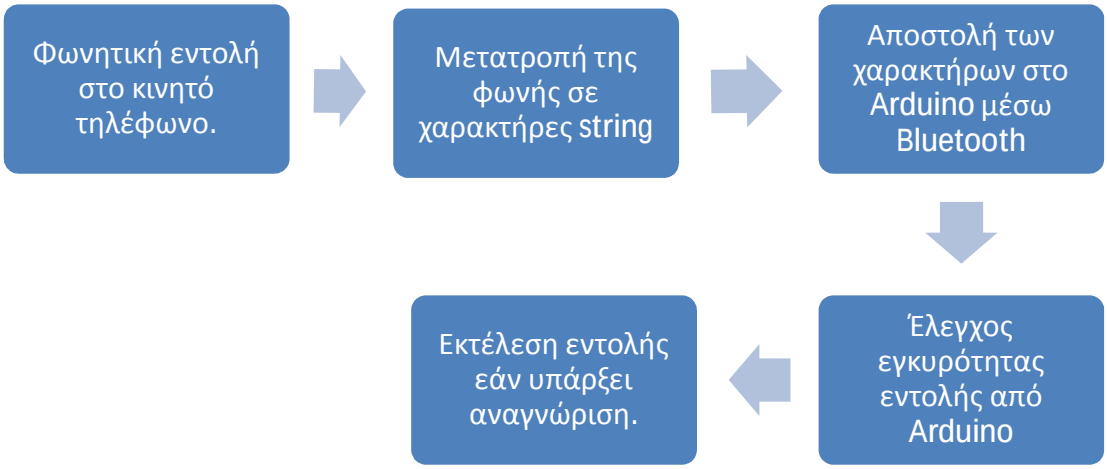
```
    else if (buffer == "ledstatus\n")
    {
      ledstate = digitalRead(led);
      Serial.println(ledstate);
      garagestate = digitalRead(garage);
      Serial.println(garagestate);
      kitchenstate = digitalRead(kitchen);
      Serial.println(kitchenstate);
      buffer = "";
    }

    buffer = "";
  }
```

Ελέγχουμε εάν έχει πατηθεί κάποιο από τα button. Εάν η συνθήκη είναι αληθής τότε καλείται πάλι η υπορουτίνα για να αλλάξει η κατάσταση του led αντίστοιχου.

```
  if (buttonState == HIGH)
  {

    ledstate = digitalRead(led);
    ledstate = change_status(ledstate);
    digitalWrite(led, ledstate);
  }
  if (buttonState1 == HIGH)
  {
    garagestate = digitalRead(garage);
    garagestate = change_status(garagestate);
    digitalWrite(garage, garagestate);
  }
```

4.1.8 Έλεγχος με φωνητικές εντολές

Όπως αναφέραμε και νωρίτερα ελέγχουμε τη σειριακή θύρα 3 του Arduino Mega και περιμένουμε κάποιο string προερχόμενο από φωνητική εντολή. Τις φωνητικές εντολές όπως είδαμε και στην ενότητα 3.3.6 τις στέλνουμε μέσω bluetooth από το κινητό μας τηλέφωνο με τη χρήση της εφαρμογής που υποδείξαμε. Στον κώδικα που ακολουθεί βλέπουμε τις εντολές που έχουμε ορίσει με τις οποίες έχουμε τη δυνατότητα να ενεργοποιήσουμε και να απενεργοποιήσουμε μερικά led. Δίνουμε μία εντολή στο κινητό μας τηλέφωνο το οποίο μετατρέπει τη φωνή μας σε χαρακτήρες string και τους στέλνει μέσω bluetooth στο Arduino. Το Arduino όπως βλέπουμε και στον παρακάτω κώδικα αναμένει για κάποιες εντολές που έχουμε ορίσει. Αν αναγνωρίσει τις εντολές που του στείλαμε θα εκτελέσει τον κώδικα που αντιστοιχεί για κάθε εντολή. Στην φωτογραφία (Εικόνα 4.5) φαίνεται και σχηματικά η διαδικασία που ακολουθείται. Στη συνέχεια ακολουθεί και ο κώδικας ελέγχου των εντολών από το Arduino.



Εικόνα 5.5-Διαδικασία ελέγχου με φωνητικές εντολές

```
if (voice.length() > 0) {
  //Serial.println(voice);

  if (voice == "*lights")
  {
    //Serial.println(voice);
    ledstate = digitalRead(led);
    ledstate = change_status(ledstate);
    digitalWrite(led, ledstate);
    voice = "";
  }
  if (voice == "*garage")
  {
    //Serial.println(voice);
    garagestate = digitalRead(garage);
    garagestate = change_status(garagestate);
    digitalWrite(garage, garagestate);
    voice = "";
  }
  if (voice == "*kitchen")
  {
    //Serial.println(voice);
    kitchenstate = digitalRead(kitchen);
    kitchenstate = change_status(kitchenstate);
    digitalWrite(kitchen, kitchenstate);
    voice = "";
  }
  voice = "";
}
```

4.1.9 **Ιστοσελίδα στην κάρτα** SD **του** Arduino

Εκτελούμε τις απαραίτητες εντολές για να ανοίξουμε τα αρχεία htm που βρίσκονται στην κάρτα SD. Επίσης στέλνουμε με XML τα δεδομένα μας στη σελίδα.

```
EthernetClient client = server1.available();  // try to get client

if (client) {  // got client?
  boolean currentLineIsBlank = true;
  while (client.connected()) {
    if (client.available()) {   // client data available to read
      char c = client.read(); // read 1 byte (character) from client
      // limit the size of the stored received HTTP request
      // buffer first part of HTTP request in HTTP_req array (string)
      // leave last element in array as 0 to null terminate string (REQ_BUF_SZ - 1)
      if (req_index < (REQ_BUF_SZ - 1)) {
        HTTP_req[req_index] = c;          // save HTTP request character
        req_index++;
      }
      // last line of client request is blank and ends with \n
```

```
      // respond to client only after last line received
      if (c == '\n' && currentLineIsBlank) {
        // send a standard http response header
        client.println("HTTP/1.1 200 OK");
        // remainder of header follows below, depending on if
        // web page or XML page is requested
        // Ajax request - send XML file
        if (StrContains(HTTP_req, "ajax_inputs")) {
          // send rest of HTTP header
          client.println("Content-Type: text/xml");
          client.println("Connection: keep-alive");
          client.println();
          SetLEDs();
          // send XML file containing input states
          XML_response(client);
        }
        else {  // web page request
          // send rest of HTTP header
          client.println("Content-Type: text/html");
          client.println("Connection: keep-alive");
          client.println();
          if (StrContains(HTTP_req, "GET / ") || StrContains(HTTP_req, "GET /index.htm")) {
            webFile = SD.open("index.htm");        // open web page file
          }
          else if (StrContains(HTTP_req, "GET /led.htm")) {
            webFile = SD.open("led.htm");        // open web page file
          }
          if (webFile) {
            while (webFile.available()) {
              client.write(webFile.read()); // send web page to client
            }
            webFile.close();
          }
        }
        // display received HTTP request on serial port
        // Serial.print(HTTP_req);
        // reset buffer index and all buffer elements to 0
        req_index = 0;
        StrClear(HTTP_req, REQ_BUF_SZ);
        break;
      }
      // every line of text received from the client ends with \r\n
      if (c == '\n') {
        // last character on line of received text
        // starting new line with next character read
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // a text character was received from client
        currentLineIsBlank = false;
      }
    } // end if (client.available())
  } // end while (client.connected())
  delay(1);      // give the web browser time to receive the data
  client.stop(); // close the connection
} // end if (client)
```

4.1.10 **Αποστολή μετρήσεων στη βάση δεδομένων**

Στέλνουμε τις μετρήσεις των αισθητήρων στη βάση δεδομένων βάζοντας μία μικρή καθυστέρηση περίπου πέντε δευτερόλεπτα έτσι ώστε να περιοριστούν τυχόν σφάλματα και περιορισμοί του προγράμματος. Αν μειώσουμε το χρόνο αυτό παρατηρούνται προβλήματα στη ροή του υπόλοιπου προγράμματος. Συγκεκριμένα δεν μπορεί να γίνει σύνδεση με την ιστοσελίδα.

```
Wait_between_dumps++;

  if (Wait_between_dumps >= 10)
  {
    if (client.connect(server, 80)) {
      // Serial.println("-> Connected");
      // Make a HTTP request:
      client.print( "GET /4/add_data.php?");
      client.print("&&");
      client.print("temperature=");
      client.print( dht.readTemperature() );
      client.print("&&");
      client.print("humidity=");
      client.print( dht.readHumidity() );
      client.print("&&");
      client.print("Irms=");
      client.print( emon1.Irms );
      client.print("&&");
      client.print("Vrms=");
```

```
      client.print( emon1.Vrms );
      client.print("&&");
      client.print("realPower=");
      client.print( emon1.realPower );
      client.print("&&");
      client.print("apparentPower=");
      client.print( emon1.apparentPower );
      client.print("&&");
      client.print("powerFActor=");
      client.print( emon1.powerFactor );
      client.print("&&");
      client.print("Wh=");
      client.print( wh );
      client.print("&&");
      client.print("motion=");
      client.print( val );
      Wait_between_dumps = 0;
      client.println( " HTTP/1.1");
      client.print( "Host: " );
      client.println(server);
      client.println( "Connection: close" );
      client.println();
      client.println();
      client.stop();
    }
  }
}

int change_status(int i)  //συνάρτηση για την αλλαγή κατάστασης συσκευών/led
{
  int result;
  result = !i;
  return result;

}
```

4.1.11 **Δημιουργία** XML **Ιστοσελίδας**

Αποθηκεύουμε τις καταστάσεις των led και ελέγχουμε αν έχουμε λάβει κάποιο HTTP request για αλλαγή κατάστασης

```
// checks if received HTTP request is switching on/off LEDs
// also saves the state of the LEDs
void SetLEDs(void)
{
  // LED 1 (pin 6)
  if (StrContains(HTTP_req, "LED1=1")) {
    ///LED_state[0] = 1;  // save LED state
    ledstate = digitalRead(led);
    ledstate = change_status(ledstate);
    digitalWrite(led, ledstate);
    //*HTTP_req = '\0';

  }
  else if (StrContains(HTTP_req, "LED1=0")) {
    // LED_state[0] = 0;  // save LED state
    ledstate = digitalRead(led);
    ledstate = change_status(ledstate);
    digitalWrite(led, ledstate);
    //*HTTP_req = '\0';
  }
  // LED 2 (pin 7)
  if (StrContains(HTTP_req, "LED2=1")) {
    LED_state[1] = 1;  // save LED state
    digitalWrite(36, HIGH);

  }
  else if (StrContains(HTTP_req, "LED2=0")) {
    LED_state[1] = 0;  // save LED state
    digitalWrite(36, LOW);
  }
  // LED 3 (pin 8)
  if (StrContains(HTTP_req, "LED3=1")) {
    LED_state[2] = 1;  // save LED state
    digitalWrite(37, HIGH);
  }
  else if (StrContains(HTTP_req, "LED3=0")) {
    LED_state[2] = 0;  // save LED state
    digitalWrite(37, LOW);
  }
  // LED 4 (pin 9)
  if (StrContains(HTTP_req, "LED4=1")) {
    LED_state[3] = 1;  // save LED state
    digitalWrite(38, HIGH);
  }
```

```
  else if (StrContains(HTTP_req, "LED4=0")) {
    LED_state[3] = 0;  // save LED state
    digitalWrite(38, LOW);
  }
  if (digitalRead(led) == HIGH)
  {
    LED_state[0] = 1;
  }
  else if (digitalRead(led) == LOW)
  {
    LED_state[0] = 0;
  }

}
```

Εδώ φτιάχνουμε και στέλνουμε το XML για τη σελίδα με τα δεδομένα των αισθητήρων και των led.

```
 // send the XML file with analog values, switch status
//  and LED status
void XML_response(EthernetClient cl)
{
  float realPower       = emon1.realPower;        //extract Real Power into variable
  float apparentPower   = emon1.apparentPower;    //extract Apparent Power into variable
  float powerFActor     = emon1.powerFactor;      //extract Power Factor into Variable
  float supplyVoltage   = emon1.Vrms;             //extract Vrms into Variable
  float Irms            = emon1.Irms;             //extract Irms into Variable
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float light_reading = analogRead(A0);
  float rain = analogRead(A1);
  float motion = digitalRead(MotionPin);
  int analog_val;            // stores value read from analog inputs
  int count;                 // used by 'for' loops
  int sw_arr[] = {2, 3, 5};  // pins interfaced to switches
  float sensor_val;

  cl.print("<?xml version = \"1.0\" ?>");
  cl.print("<inputs>");


  // read sensors inputs

  sensor_val = dht.readTemperature();
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = dht.readHumidity();
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = emon1.Irms;
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = emon1.Vrms;
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = emon1.realPower;
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = emon1.apparentPower;
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = emon1.powerFactor;
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");
  sensor_val = digitalRead(MotionPin);
  cl.print("<sensor>");
  cl.print(sensor_val);
  cl.print("</sensor>");

  // checkbox LED states
  // LED1
  cl.print("<LED>");
  if (LED_state[0]) {
    cl.print("on");
  }
  else {
    cl.print("off");
  }
  cl.println("</LED>");
  // LED2
```

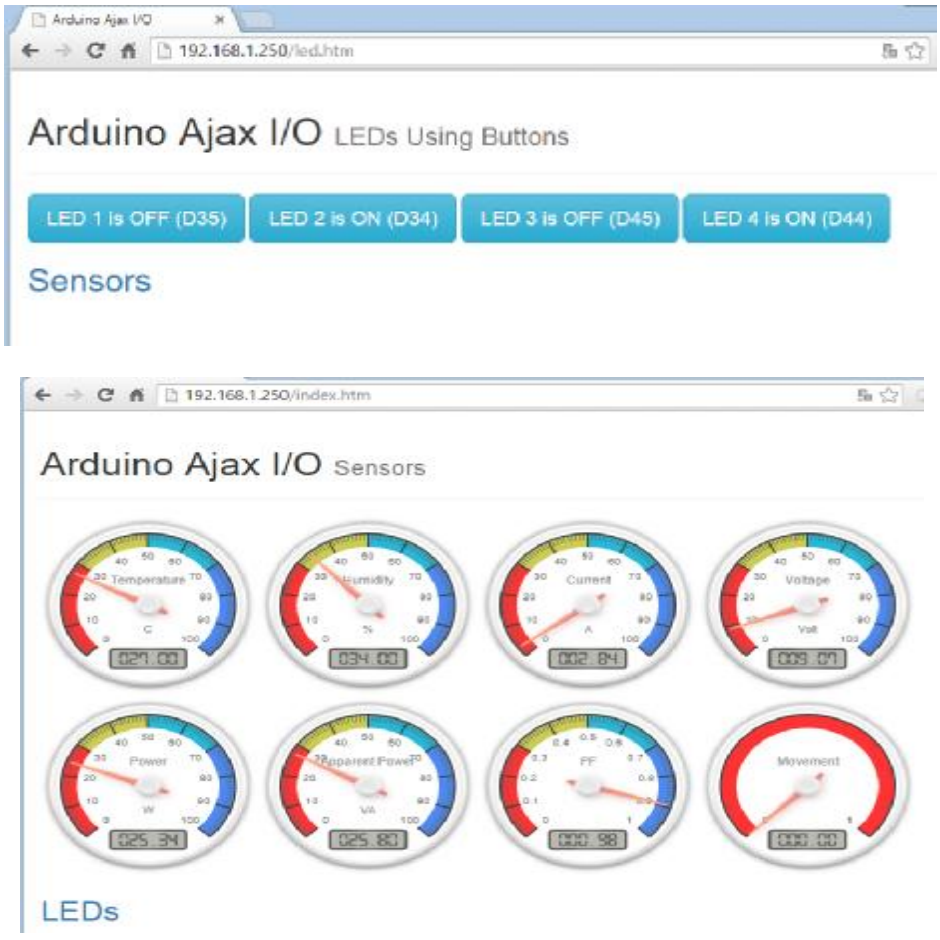
```
  cl.print("<LED>");
  if (LED_state[1]) {
    cl.print("on");
  }
  else {
    cl.print("off");
  }
  cl.println("</LED>");
  // button LED states
  // LED3
  cl.print("<LED>");
  if (LED_state[2]) {
    cl.print("on");
  }
  else {
    cl.print("off");
  }
  cl.println("</LED>");
  // LED4
  cl.print("<LED>");
  if (LED_state[3]) {
    cl.print("on");
  }
  else {
    cl.print("off");
  }
  cl.println("</LED>");

  cl.print("</inputs>");
}
// sets every element of str to 0 (clears array)
void StrClear(char *str, char length)
{
  for (int i = 0; i < length; i++) {
    str[i] = 0;
  }
}

// searches for the string sfind in the string str
// returns 1 if string found
// returns 0 if string not found
char StrContains(char *str, char *sfind)
{
  char found = 0;
  char index = 0;
  char len;

  len = strlen(str);
  if (strlen(sfind) > len) {
    return 0;
  }
  while (index < len) {
    if (str[index] == sfind[found]) {
      found++;
      if (strlen(sfind) == found) {
        return 1;
      }
    }
    else {
      found = 0;
    }
    index++;
  }

  return 0;
}
```

4.2 Ιστοσελίδα που υλοποιήθηκε και λειτουργίες της

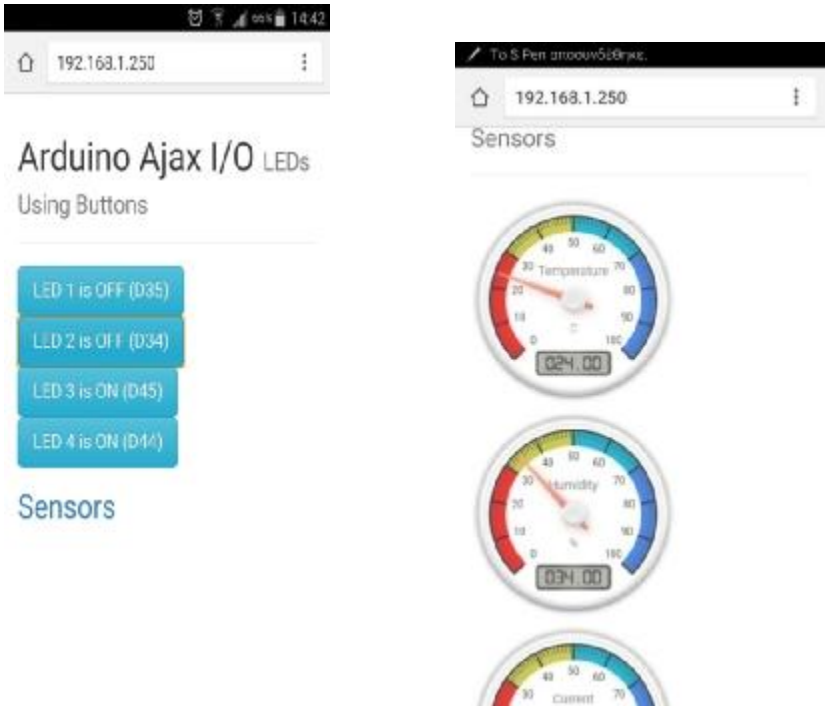
Στα πλαίσια της πτυχιακής εργασίας δημιουργήθηκε και μια ιστοσελίδα με σκοπό, εκτός από το LabVIEW, να μπορούμε και σε κάποιον άλλο υπολογιστή ή σε ένα κινητό τηλέφωνο να έχουμε ενδείξεις από τους αισθητήρες του συστήματος, καθώς επίσης να μπορούμε να ελέγχουμε και κάποιες συσκευές. Το LabVIEW βέβαια έχει web interface το οποίο μπορεί να λειτουργήσει στον browser ενός υπολογιστή, δεν μπορεί όμως να λειτουργήσει στον browser κινητών τηλεφώνων. Στις επόμενες φωτογραφίες (Εικόνα 4.7) έως (Εικόνα 4.9) φαίνεται η ιστοσελίδα ελέγχου του Arduino στον browser του υπολογιστή και στον browser του κινητού. Στην αμέσως επόμενη εικόνα (Εικόνα 4.6) φαίνεται μια σελίδα στον server του υπολογιστή που μας επιτρέπει

να περιηγηθούμε στην ιστοσελίδα που φτιάξαμε για να ελέγχουμε το Arduino και την οποία θα αναλύσουμε σε αυτή την ενότητα, καθώς και στη βάση και στα γραφήματα που θα δούμε στις επόμενες ενότητες 4.3 και 4.4.

Εικόνα 4.6-Αρχική σελίδα ελέγχου συστήματος



Εικόνα 4.7- Η ιστοσελίδα από τον Browser του υπολογιστή



Εικόνα 4.8- Η Ιστοσελίδα από τον Browser του κινητού

Εικόνα 4.9- Η ιστοσελίδα από τον Browser του κινητού

Επίσης λόγω του ότι η μνήμη του Arduino είναι σχετικά περιορισμένη επιλέξαμε να αξιοποιήσουμε τον αναγνώστη καρτών SD του Ethernet Shield και να τοποθετήσουμε σε μία κάρτα SD αρχεία html (htm για την ακρίβεια) τα οποία θα τα καλεί το Arduino. Άλλα προβλήματα που παρουσιάστηκαν ήταν η αδυναμία ανάγνωσης φωτογραφιών όπως επίσης και περισσότερων script. Λόγω του όγκου της πληροφορίας που πρέπει να επεξεργαστεί και να στείλει το Arduino δημιουργούνται καθυστερήσεις. Για το λόγο αυτό δημιουργήσαμε ξεχωριστά links για

την ανάγνωση των μετρήσεων και την ενεργοποίηση των Led και περιορίσαμε τα χρησιμοποιούμενα γραφικά μετρήσεων.

4.2.1 Πως λειτουργεί ο server της ιστοσελίδας

Αφού ένας web browser έχει ζητήσει και έχει φορτώσει τον τη σελίδα, το JavaScript στην ιστοσελίδα θα στείλει Ajax request για κάθε δεδομένο του Arduino κάθε δευτερόλεπτο. Αυτό σημαίνει πως δεν θα ανανεώνεται ολόκληρη η σελίδα αλλά μόνο οι μόνο τα δεδομένα.

Το Arduino θα απαντά σε κάθε Ajax request στέλνοντας πίσω ένα XML αρχείο στον web browser. Το XML αρχείο περιέχει τις τιμές των αισθητήρων που έχουμε ορίσει και τις καταστάσεις από τέσσερα led.

Όταν ένα LED ενεργοποιείται από τη σελίδα από ένα button τότε το JavaScript στέλνει την κατάσταση του button (στέλνει μια οδηγία για να ενεργοποιηθεί το led) με ένα καινούριο Ajax request.

Οι τιμές των αισθητήρων απεικονίζονται σε μετρητές τύπου gauge με τη βοήθεια JavaScript. Επίσης έχει γίνει διαχωρισμός σε δύο σελίδες για ευκολία στη χρήση, μια για τον έλεγχο των led και μία για τον έλεγχο των αισθητήρων.

Κώδικας ιστοσελίδων

Οι γραμμές όπως

```
temperature  =
this.responseXML.getElementsByTagName('sensor')[0].childNodes[0].nodeValue;
```

παίρνουν από το XML του Arduino την τιμή του πίνακα sensor που αντιστοιχεί στη θέση 0 και την αποθηκεύει στη μεταβλητή temperature. Αντίστοιχα λειτουργούν και για τους υπόλοιπους αισθητήρες. Στη συνέχεια το αντίστοιχο JavaScript διαβάζει την τιμή temperature και την εμφανίζει στον μετρητή gauge. Στον παρακάτω κώδικα, εσκεμμένα έχει αφαιρεθεί το κομμάτι του JavaScript. Ο ολοκληρωμένος κώδικας βρίσκεται στο παράρτημα.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Arduino Ajax I/O</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

        <!-- Optional theme -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">

        <link rel="stylesheet" type="text/css" href="custom.css">

        <!-- Latest compiled and minified JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>

        <meta name="viewport" content="width=device-width, initial-scale=1">

 <script>
           var temperature = 0;
           var humidity = 0;
           var current = 0;
           var voltage = 0;
           var power = 0;
           var apparentPower = 0;
           var pf = 0;
           var movement = 0;

eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e........0,{}))  //  JavaScript
```

```
        function GetArduinoIO() {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function() {
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseXML != null) {

                                                            // get sensor inputs
                            temperature  =
this.responseXML.getElementsByTagName('sensor')[0].childNodes[0].nodeValue;
                            humidity =
this.responseXML.getElementsByTagName('sensor')[1].childNodes[0].nodeValue;
                                                            current =
this.responseXML.getElementsByTagName('sensor')[2].childNodes[0].nodeValue;
                                                            voltage =
this.responseXML.getElementsByTagName('sensor')[3].childNodes[0].nodeValue;
                                                            power =
this.responseXML.getElementsByTagName('sensor')[4].childNodes[0].nodeValue;
                                                            apparentPower =
this.responseXML.getElementsByTagName('sensor')[5].childNodes[0].nodeValue;
                                                            pf =
this.responseXML.getElementsByTagName('sensor')[6].childNodes[0].nodeValue;
                                                            movement =
this.responseXML.getElementsByTagName('sensor')[7].childNodes[0].nodeValue;

                            }
                }
            }
        }
            // send HTTP GET request with LEDs to switch on/off if any
            request.open("GET", "ajax_inputs" + nocache, true);
            request.send(null);
            setTimeout('GetArduinoIO()', 1000);
        }

    </script>

    </head>
    <body onload="GetArduinoIO()">
        <div class="container-fluid">
        <div class="page-header">
                <h1>Arduino Ajax I/O <small>Sensors</small></h1>
        </div>
    <canvas id="an_gauge_1" width="200" height="200"  data-title="Temperature" data-units="C"
data-highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-ticks="0 10 20 30
40 50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-value="100" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_1').setValue(temperature);},
200);"></canvas>
    <canvas id="an_gauge_2" width="200" height="200"  data-title="Humidity" data-units="%" data-
highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-ticks="0 10 20 30 40
50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-value="100" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_2').setValue(humidity);},
200);"></canvas>
    <canvas id="an_gauge_3" width="200" height="200"  data-title="Current" data-units="A" data-
highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-ticks="0 10 20 30 40
50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-value="100" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_3').setValue(current);},
200);"></canvas>
        <canvas id="an_gauge_4" width="200" height="200"  data-title="Voltage" data-units="Volt"
data-highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-ticks="0 10 20 30
40 50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-value="100" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_4').setValue(voltage);},
200);"></canvas>
        <canvas id="an_gauge_5" width="200" height="200"  data-title="Power" data-units="W" data-
highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-ticks="0 10 20 30 40
50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-value="100" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_5').setValue(power);},
200);"></canvas>
        <canvas id="an_gauge_6" width="200" height="200"  data-title="Apparent Power" data-
units="VA" data-highlights="0 30 #f33, 30 50 #cc5,50 70 #25B8D9, 70 100 #4D89F2" data-major-
ticks="0 10 20 30 40 50 60 70 80 90 100"  data-type="canv-gauge" data-min-value="0" data-max-
value="100" data-onready="setInterval( function() {
Gauge.Collection.get('an_gauge_6').setValue(apparentPower);}, 200);"></canvas>
        <canvas id="an_gauge_7" width="200" height="200"  data-title="PF" data-units="" data-
highlights="0 0.3 #f33, 0.3 0.5 #cc5,0.5 0.7 #25B8D9, 0.7 1 #4D89F2" data-major-ticks="0 0.1 0.2
0.3 0.4 0.5 0.6 0.7 0.8 0.9 1"  data-type="canv-gauge" data-min-value="0" data-max-value="1" data-
onready="setInterval( function() { Gauge.Collection.get('an_gauge_7').setValue(pf);},
200);"></canvas>
    <canvas id="an_gauge_8" width="200" height="200"  data-title="Movement" data-units="" data-
highlights="0 1 #f33" data-major-ticks="0 1"  data-type="canv-gauge" data-min-value="0" data-max-
value="1" data-onready="setInterval( function() {
Gauge.Collection.get('an_gauge_8').setValue(movement);}, 200);"></canvas>
```

```
        <h2><a href="led.htm">LEDs</a></h2>

      </div>
    </body>
</html>
```

Όσον αφορά τον έλεγχο των Leds τώρα, στον κώδικα του Arduino έχουμε δημιουργήσει ένα πίνακα ο οποίος αποθηκεύει την κατάσταση των Led (αποθηκεύει 1 αν είναι on και 0 αν είναι off).

```
boolean LED_state[4] = {0}; // stores the states of the LEDs
```

Αυτή η γραμμή αρχικοποιεί όλα τα Led στο 0. Αντίστοιχα όταν αναφερόμαστε στο LED_state[0] αποθηκεύουμε την κατάσταση του Led 1, LED_state[1] αποθηκεύουμε την κατάσταση του Led 2 κλπ.

Στη σελίδα κάθε button έχει τη δική του συνάρτηση JavaScript η οποία τρέχει κάθε φορά που πατιέται. Το κείμενο στα button ανανεώνεται έτσι ώστε να φαίνεται η κατάσταση των Led η οποία αποθηκεύεται στις εξής μεταβλητές:

```
var LED3_state = 0;
var LED4_state = 0;
```

Η αποθήκευση της κατάστασης των Led είναι απαραίτητη έτσι ώστε όταν ένα button πατηθεί ξανά να γνωρίζει εάν πρέπει να ενημερώσει το string που έχει σταλεί μαζί με το GET request για να αλλάξει την κατάσταση του Led ή όχι.
Όταν το Arduino στέλνει την κατάσταση των Leds στο XML αρχείο με το οποίο ελέγχονται μέσω των button, εισάγει "on" ή "off" ανάμεσα στα tags <LED>.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Arduino Ajax I/O</title>
        <!-- Latest compiled and minified CSS -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

        <!-- Optional theme -->
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap-theme.min.css">

        <link rel="stylesheet" type="text/css" href="custom.css">

        <!-- Latest compiled and minified JavaScript -->
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>

        <meta name="viewport" content="width=device-width, initial-scale=1">

        <script>
        strLED1 = "";
        strLED2 = "";
        strLED3 = "";
        strLED4 = "";
        var LED3_state = 0;
        var LED4_state = 0;

        function GetArduinoIO()
        {
            nocache = "&nocache=" + Math.random() * 1000000;
            var request = new XMLHttpRequest();
            request.onreadystatechange = function()
            {
                if (this.readyState == 4) {
                    if (this.status == 200) {
                        if (this.responseXML != null) {
                            // XML file received - contains analog values, switch values, sensors
and LED states

                            // LED 1
                            if
(this.responseXML.getElementsByTagName('LED')[0].childNodes[0].nodeValue === "on") {
                                document.getElementById("LED1").innerHTML = "LED 1 is ON (D35)";
                                LED1_state = 1;
                            }
                            else {
                                document.getElementById("LED1").innerHTML = "LED 1 is OFF (D35)";
                                LED1_state = 0;
                            }
```

```
                            // LED 2
                            if
(this.responseXML.getElementsByTagName('LED')[1].childNodes[0].nodeValue === "on") {
                                document.getElementById("LED2").innerHTML = "LED 2 is ON (D36)";
                                LED2_state = 1;
                            }
                            else {
                                document.getElementById("LED2").innerHTML = "LED 2 is OFF (D36)";
                                LED2_state = 0
                            }
                            // LED 3
                            if
(this.responseXML.getElementsByTagName('LED')[2].childNodes[0].nodeValue === "on") {
                                document.getElementById("LED3").innerHTML = "LED 3 is ON (D37)";
                                LED3_state = 1;
                            }
                            else {
                                document.getElementById("LED3").innerHTML = "LED 3 is OFF (D37)";
                                LED3_state = 0;
                            }
                            // LED 4
                            if
(this.responseXML.getElementsByTagName('LED')[3].childNodes[0].nodeValue === "on") {
                                document.getElementById("LED4").innerHTML = "LED 4 is ON (D38)";
                                LED4_state = 1;
                            }
                            else {
                                document.getElementById("LED4").innerHTML = "LED 4 is OFF (D38)";
                                LED4_state = 0;

                            }
                        }
                    }
                }
            }
            // send HTTP GET request with LEDs to switch on/off if any
            request.open("GET", "ajax_inputs" + strLED1 + strLED2 + strLED3 + strLED4 + nocache,
true);
            request.send(null);
            setTimeout('GetArduinoIO()', 10000);
            strLED1 = "";
            strLED2 = "";
            strLED3 = "";
            strLED4 = "";
        }
        function GetButton1()
        {
            if (LED1_state === 1) {
                LED1_state = 0;
                strLED1 = "&LED1=0";
            }
            else {
                LED1_state = 1;
                strLED1 = "&LED1=1";
            }
        }

        function GetButton2()
        {
            if (LED2_state === 1) {
                LED2_state = 0;
                strLED2 = "&LED2=0";
            }
            else {
                LED2_state = 1;
               strLED2 = "&LED2=1";
            }
        }

        function GetButton3()
        {
            if (LED3_state === 1) {
                LED3_state = 0;
                strLED3 = "&LED3=0";
            }
            else {
                LED3_state = 1;
                strLED3 = "&LED3=1";
            }
        }
        function GetButton4()
        {
            if (LED4_state === 1) {
                LED4_state = 0;
                strLED4 = "&LED4=0";
            }
            else {
                LED4_state = 1;
```

```
                strLED4 = "&LED4=1";
            }
        }
    </script>

    </head>
    <body onload="GetArduinoIO()">
        <div class="container-fluid">
        <div class="page-header">
                <h1>Arduino Ajax I/O <small>LEDs Using Buttons</small></h1>
        </div>


            <button type="button" class="btn btn-info btn-lg" id="LED1" onclick="GetButton1()">LED
1 is OFF (D35)</button>
            <button type="button" class="btn btn-info btn-lg" id="LED2" onclick="GetButton2()">LED
2 is OFF (D36)</button>
            <button type="button" class="btn btn-info btn-lg" id="LED3" onclick="GetButton3()">LED
3 is OFF (D37)</button>
            <button type="button" class="btn btn-info btn-lg" id="LED4" onclick="GetButton4()">LED
4 is OFF (D38)</button>


        <h2><a href="index.htm">Sensors</a></h2>

        </div>

    </body>
</html>
```

4.3 Βάση δεδομένων

Εκτός από την απλή μέτρηση διάφορων τιμών από τους αισθητήρες που χρησιμοποιούμε, μας ενδιαφέρει να μπορούμε να καταγράφουμε τις μετρούμενες αυτές τιμές για περεταίρω έλεγχο και μελλοντική αξιοποίηση από τον χρήστη. Για να καταγράψουμε τις μετρήσεις των αισθητήρων μας δημιουργήσαμε μία βάση δεδομένων σε mySQL η οποία είναι μια ελεύθερη βάση δεδομένων, με τη βοήθεια του Apache XAMPP και κώδικα σε PHP. Στη συνέχεια για να έχουμε και μια καλύτερη εικόνα των μετρήσεων μας κάποιες βασικές μετρήσεις με τη βοήθεια την βιβλιοθήκης AmCharts καταφέραμε να τις παρουσιάσουμε σε γραφήματα ως προς το χρόνο όπως θα δείξουμε στο επόμενο κεφάλαιο. για το σύστημά μας λοιπόν ήταν απαραίτητο να φτιάξουμε ένα πίνακα σε μία βάση δεδομένων ο οποίος θα με Data Push από το Arduino θα γεμίζει με δεδομένα. Ας δούμε τα βήματα που ακολουθήσαμε για να φτιάξουμε τη βάση με τον πίνακα μας και πως καταφέραμε να τον συμπληρώνουμε με τις μετρήσεις μας.

Σε αυτό το σημείο λοιπών είπαμε θα κάνουμε Data Push τα δεδομένα του Arduino. Αυτό σημαίνει πως θα κάνουμε το Arduino να στέλνει δεδομένα σε ένα server, στον οποίο αποθηκεύουμε τις μετρήσεις των αισθητήρων σε μια βάση δεδομένων. Για να το κάνουμε αυτό χρειαζόμαστε έναν Apache, γνώσεις MySQL και PHP, με τα οποία μπορούμε να φτιάξουμε έναν ολοκληρωμένο server με εύκολο σχετικά τρόπο κάνοντας εγκατάσταση στον υπολογιστή μας το πρόγραμμα WAMPServer ή στη δική μας περίπτωση το XAMPP.

Από την πλευρά του Arduino χρειαζόμαστε, προφανώς ένα Arduino, και ένα Ethernet Shield. Σε προγραμματιστικό επίπεδο αρχικά θα χρειαστούμε δύο βιβλιοθήκες που έρχονται μαζί με το Arduino IDE. Οι βιβλιοθήκες αυτές είναι οι Ethernet.h και η SPI.h. Το Ethernet Shield που χρησιμοποιούμε βασίζεται στον W5100 άρα μας αρκούν οι παραπάνω βιβλιοθήκες. Για να εισάγουμε στον κώδικα μας τις βιβλιοθήκες αυτές απλά γράφουμε τις παρακάτω δυο γραμμές:

```
#include <SPI.h>
#include <Ethernet.h>
```
Στο παράδειγμά μας θα χρησιμοποιήσουμε έναν αισθητήρα DHT11 για να καταγράψουμε τιμές θερμοκρασίας στη βάση μας.
Τι σημαίνει όμως Pull και τι σημαίνει Push δεδομένων;

Έχουμε δύο δυνατότητες για το πώς θα πάρουμε δεδομένα από το Arduino μας. Η καλύτερη επιλογή εξαρτάται από την εφαρμογή μας.

- Το Pull δεδομένων αφορά μια εφαρμογή στον υπολογιστή μας ή τον server μας που πρόκειται να ζητήσει από το Arduino για τα δεδομένα.
- Στην περίπτωση του Push δεδομένων, το Arduino θα είναι αυτό που θα συνδεθεί με τον server για να στείλει τα δεδομένα του.

Όπως είπαμε και προηγουμένως εμείς θα χρησιμοποιήσουμε τη μέθοδο push data. Η εναλλακτική λύση, Pull data , είναι πολύ πιο εύκολη να εφαρμοστεί, δεδομένου ότι δεν χρειαζόμαστε έναν "server". Σε αυτή την εφαρμογή, ωστόσο θα χρησιμοποιήσουμε την πιο σύνθετη προσέγγιση Push δεδομένων. Στην ουσία πρόκειται να θέσουμε το Arduino να συμπληρώνει μια φόρμα ιστοσελίδας και να υποβάλει τα στοιχεία του στον server μας. Για το σκοπό αυτό θα χρησιμοποιήσουμε MySQL και PHP.

Pushing Data

Αρχικά θα χρειαστούμε έναν δέκτη δεδομένων, μια εφαρμογή server που να είναι έτοιμη και ικανή να δέχεται δεδομένα. Αυτή η εφαρμογή μπορεί να ακούει σε μία συγκεκριμένη IP και σε μια συγκεκριμένη θύρα IP. Μία τέτοια εφαρμογή μπορεί να είναι για παράδειγμα η MySQL (ένας ελεύθερος server βάσης δεδομένων που μπορεί να τρέχει στον υπολογιστή μας όπως το XAMPP).

Pushing Data μέσω PHP

Στον PHP server μας θα χρειαστούμε ένα αρχείο το οποίο θα δέχεται και θα επεξεργάζεται τις πληροφορίες. Επίσης θα χρειαστούμε έναν πίνακα στη mySQL για να αποθηκεύουμε τα δεδομένα. Αυτό το αρχείο θα το ονομάσουμε add_data.php και θα κάνει post τα δεδομένα που δέχεται στον πίνακα mySQL.

MySQL Table:

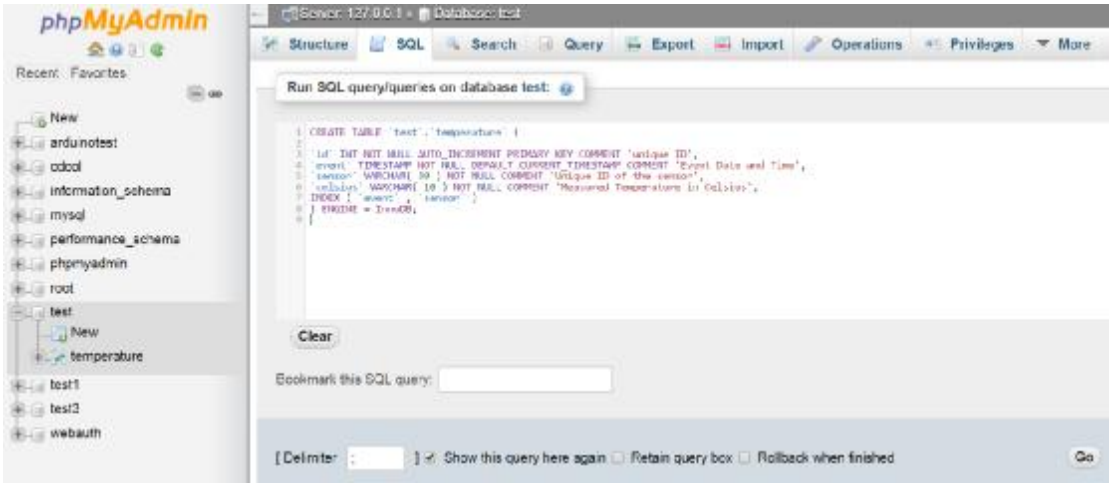
Έστω πως θέλουμε να αποθηκεύουμε τις εξής τιμές

- Unique ID (int),
- Event date and time (timestamp) για να ξέρουμε πότε έγινε η καταχώρηση,
- Sensor serial number (varchar) για να ξέρουμε ποιός αισθητήρας έστειλε τιμή για θερμοκρασία,
- Temperature in Celsius (varchar) για να το διατηρήσουμε απλό.

Για να δημιουργήσουμε τον πίνακα temperature στην SQL με τα παραπάνω δεδομένα θα χρειαστούμε τις εξής γραμμές κώδικα PHP (μπορούμε να φτιάξουμε τον πίνακα και από το phpmyadmin) όπως βλέπουμε και στην Εικόνα 4.8

```
CREATE TABLE `test`.`temperature` (
```

```
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT 'unique ID',
`event` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Event Date and Time',
`sensor` VARCHAR( 30 ) NOT NULL COMMENT 'Unique ID of the sensor',
`celsius` VARCHAR( 10 ) NOT NULL COMMENT 'Measured Temperature in Celsius',
INDEX ( `event` , `sensor` )
) ENGINE = InnoDB;
```



Εικόνα 4.9-Δημιουργία πίνακα σε MySQL

Κάθε φορά που θέλουμε να προσθέσουμε μία είσοδο στον πίνακα (INSERT), αρκεί να προσθέσουμε τον αισθητήρα (sensor), τον αριθμό του αισθητήρα(serial number) και την τιμή της θερμοκρασίας σε βαθμούς κελσίου (temperature in Celsius). Οι τιμές ID και date/time θα αυξάνονται αυτόματα από την mySQL.

Ένα παράδειγμα εισόδου στοιχείου(INSERT):

```
INSERT INTO test.temperature (sensor ,celsius) VALUES ('288884820500006a', '21.65');
```

Ονομάζουμε την βάση δεδομένων που χρησιμοποιούμε στο παράδειγμα μας "test". Ο λόγος είναι ότι μπορούμε να τη βρούμε σε κάθε βάση mySQL. Μπορούμε να την ονομάσουμε όπως επιθυμούμε ωστόσο. Το ίδιο ισχύει και για το όνομα του πίνακα. Απλά πρέπει να χρησιμοποιούμε σταθερά και επανειλημμένα τα ίδια ονόματα σε όλο το project μας.

Ο πίνακας είναι έτοιμος. Όσον αφορά τώρα την PHP πρόκειται να δημιουργήσουμε τρία αρχεία PHP. Ένα για να υποβάλλουμε τα δεδομένα και ένα για να αντλούμε τα δεδομένα έτσι ώστε να έχουμε επίγνωση του τι συμβαίνει. Και τα δυο θα είναι συνδεδεμένα με τη βάση έτσι λοιπόν θα φτιάξουμε και ένα τρίτο το οποίο τα άλλα δύο θα μοιράζονται έτσι ώστε να επικοινωνούν με τη mySQL.

PHP: MySQL Σύνδεση

Αυτό το αρχείο έχει μόνο ένα σκοπό, να εγκαταστήσει μία σύνδεση μεταξύ PHP και mySQL η οποία θα χρησιμοποιείται και για να στέλνουμε και για να δεχόμαστε δεδομένα. Ονομάζουμε το αρχείο αυτό “dbconnect.php” και το αποθηκεύουμε στο directory που επιθυμούμε να έχουμε τα αρχεία της βάσης μας. Στον ίδιο φάκελο θα έχουμε και οτιδήποτε αφορά τη βάση μας. Το αρχείο αυτό περιλαμβάνει τον παρακάτω κώδικα:

```
<?php
$MyUsername = "your username";  // enter your username for mysql
$MyPassword = "your password";  // enter your password for mysql
```

```
$MyHostname = "localhost";       // this is usually "localhost" unless your database
resides on a different server

$dbh = mysql_pconnect($MyHostname , $MyUsername, $MyPassword);
$selected = mysql_select_db("test",$dbh);
```

PHP: Εμφάνιση περιεχομένου πίνακα

Το επόμενο αρχείο μας θα το ονομάσουμε "review_data.php" και θα το αποθηκεύσουμε στο ίδιο directory. Για να ελέγξουμε ότι μέχρι ώρας όλα δουλεύουν σωστά μπορούμε στον browser μας να συνδεθούμε στο http://localhost/review_data.php. Ο πίνακας θα είναι άδειος μέχρι να του προσθέσουμε κάποιο δεδομένο. Ο κώδικας θα είναι ο εξής:

```
<?php
    // Start MySQL Connection
    include('dbconnect.php');
?>

<html>
<head>
    <title>Arduino Temperature Log</title>
    <style type="text/css">
        .table_titles, .table_cells_odd, .table_cells_even {
                padding-right: 20px;
                padding-left: 20px;
                color: #000;
        }
        .table_titles {
            color: #FFF;
            background-color: #666;
        }
        .table_cells_odd {
            background-color: #CCC;
        }
        .table_cells_even {
            background-color: #FAFAFA;
        }
        table {
            border: 2px solid #333;
        }
        body { font-family: "Trebuchet MS", Arial; }
    </style>
</head>

    <body>
        <h1>Arduino Temperature Log</h1>
    <table border="0" cellspacing="0" cellpadding="4">
      <tr>
            <td class="table_titles">ID</td>
            <td class="table_titles">Date and Time</td>
            <td class="table_titles">Sensor Serial</td>
            <td class="table_titles">Temperature in Celsius</td>
        </tr>
<?php
    // Retrieve all records and display them
    $result = mysql_query("SELECT * FROM temperature ORDER BY id ASC");

    // Used for row color toggle
    $oddrow = true;

    // process every record
    while( $row = mysql_fetch_array($result) )
    {
        if ($oddrow)
        {
            $css_class=' class="table_cells_odd"';
        }
        else
        {
            $css_class=' class="table_cells_even"';
        }

        $oddrow = !$oddrow;
```

```
        echo '<tr>';
        echo '   <td'.$css_class.'>'.$row["id"].'</td>';
        echo '   <td'.$css_class.'>'.$row["event"].'</td>';
        echo '   <td'.$css_class.'>'.$row["sensor"].'</td>';
        echo '   <td'.$css_class.'>'.$row["celsius"].'</td>';
        echo '</tr>';
    }
?>
    </table>
    </body>
</html>
```

PHP: Upload Data

Σε αυτό το αρχείο θα αξιοποιήσουμε το <FORM> χρησιμοποιώντας την εντολή GET. Με το GET οι τιμές θα εισάγονται στη βάση δεδομένων και το "review.data.php" θα φορτώνεται μετά από αυτό από τον browser, αν και είναι προαιρετικό και δεν χρειάζεται για το Arduino. Το χρησιμοποιούμε για δοκιμαστικούς λόγους σε αυτή τη φάση αλλά μας χρειάζεται και για την υλοποίηση του project μας. Για να αφαιρέσουμε αυτή τη λειτουργία αρκεί να διαγράψουμε τη γραμμή 12. Το επόμενο αρχείο μας λοιπόν θα το ονομάσουμε "add_data.php" και θα το αποθηκεύσουμε στο ίδιο directory. Ο κώδικας είναι ο εξής:

```
<?php
    // Connect to MySQL
    include("dbconnect.php");

    // Prepare the SQL statement
    $SQL    =    "INSERT    INTO    test.temperature    (sensor    ,celsius)    VALUES
('".$_GET["serial"]."', '".$_GET["temperature"]."')";

    // Execute SQL statement
    mysql_query($SQL);

    // Go to the review_data.php (optional)
    header("Location: review_data.php");
?>
```

Εάν τώρα για παράδειγμα εισάγουμε στον browser μια διεύθυνση όπως http://localhost/add_data.php?serial=288884820500006b&temperature=20.79

μπορούμε να δοκιμάσουμε εάν μπορούν να εισαχθούν δεδομένα στη βάση μας.

Η βάση δεδομένων μας είναι έτοιμη. Το μόνο που μένει είναι να ετοιμάσουμε το Arduino έτσι ώστε να μπορεί να στέλνει δεδομένα στη βάση.

Δοκιμάζοντας τον κώδικα του Arduino

Σε αυτό το παράδειγμα θα γίνει κατανόηση του τρόπου λειτουργίας του συστήματος. Αρχικά δεν θα στέλνουμε στη βάση τιμές των αισθητήρων. Θα συνδεθούμε στον server και θα στέλνουμε κάποια τυχαία δεδομένα κάθε 5 δευτερόλεπτα. Ανανεώνοντας τον Browser κάθε 5 δευτερόλεπτα θα βλέπουμε να έχει προστεθεί άλλη μια γραμμή με δεδομένα στις στήλες που έχουμε ορίσει.

Ο κώδικας που θα κάνουμε upload στο Arduino είναι ο εξής:

```
#include <SPI.h>
#include <Ethernet.h> // Used for Ethernet
// **** ETHERNET SETTING ****
// Ethernet MAC address - must be unique on your network - MAC Reads T4A001 in hex
(unique in your network)
byte mac[] = { 0x54, 0x34, 0x41, 0x30, 0x30, 0x31 }; //For the rest we use DHCP (IP
address and such)
```

```
EthernetClient client;
char server[] = "192.168.1.100"; //
int  interval = 5000; // Wait between dumps

void setup() {

  Serial.begin(9600);
  Ethernet.begin(mac);
  Serial.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
  Serial.print("IP Address        : ");
  Serial.println(Ethernet.localIP());
  Serial.print("Subnet Mask       : ");
  Serial.println(Ethernet.subnetMask());
  Serial.print("Default Gateway IP: ");
  Serial.println(Ethernet.gatewayIP());
  Serial.print("DNS Server IP     : ");
  Serial.println(Ethernet.dnsServerIP());
}

void loop() {
  // if you get a connection, report back via serial:
  if (client.connect(server, 80)) {  Προσπαθούμε να επιτύχουμε σύνδεση
    Serial.println("-> Connected");
    // Make a HTTP request:
    client.print( "GET /testserver/arduino_temperatures/add_data.php?");  GET request
    client.print("serial=");
    client.print( "288884820500006X" );
    client.print("&&");
    client.print("temperature=");
    client.print( "12.3" );
    Τα       δεδομένα       που       στέλνουμε       θα       εχουν       τη       μορφη
serial=288884820500006X&temperature=12.3

    client.println( " HTTP/1.1");
    client.print( "Host: " );    Κλείνουμε το HTTP GET request
    client.println(server)
    client.println( "Connection: close" );
    client.println();
    client.println();
    client.stop();
  }
  else {
    // you didn't get a connection to the server:
    Serial.println("--> connection failed/n");
  }

  delay(interval);
}
```

Η έννοια του localhost αφορά τον browser του υπολογιστή μας. Δεν θα λειτουργήσει στο Arduino αν ορίσουμε τη διεύθυνση που θα λαμβάνει δεδομένα με localhost γιατί θα καλεί τον εαυτό του και όχι τον server μας.

Πως να στέλνουμε πραγματικές τιμές στη βάση

Μέχρι τώρα είδαμε πώς μπορούμε να στείλουμε τιμές που ορίζουμε εμείς στη βάση δεδομένων. Πως μπορούμε όπως να στείλουμε τις μετρήσεις των αισθητήρων που έχουμε συνδέσει στο Arduino; Για αρχή ο αισθητήρας που έχουμε συνδέσει είναι ο DHT11 που μετρά υγρασία και θερμοκρασία. Έρχεται με δικιά του βιβλιοθήκη στην οποία ορίζουμε σε πιο Pin τον έχουμε συνδέσει και ποιός τύπος DHT είναι(υπάρχει και ο DHT22). Στη συνέχεια οι αλλαγές που πρέπει να κάνουμε για να μπορούμε να γράφουμε στη βάση μας τις τιμές θερμοκρασίας που μετράει ο αισθητήρας είναι απλές. Αρκεί να αλλάξουμε λίγες γραμμές στον κώδικα του Arduino.

Αρχικά προσθέτουμε τη βιβλιοθήκη με της απαραίτητε δηλώσεις ως εξής:

```
#include "DHT.h"
#define DHTPIN 22    // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11
```

```
DHT dht(DHTPIN, DHTTYPE);
```

Στο loop() ορίζουμε μεταβλητή t στην οποία σώζουμε της μετρήσεις θερμοκρασίας

```
float t = dht.readTemperature();
```

Στη θέση της τιμής "12.3" αρκεί να κάνουμε αντικατάσταση με τη μεταβλητή t

```
client.print("temperature=");
client.print(t);
```

Ο κώδικας θα μετατραπεί δηλαδή ως εξής:

```
#include <SPI.h>                              Εισάγουμε τις απαραίτητες βιβλιοθήκες
#include <Ethernet.h> // Used for Ethernet
#include "DHT.h"
#define DHTPIN 22
#define DHTTYPE DHT11
#DHT dht(DHTPIN,DHTTYPE);

// **** ETHERNET SETTING ****
// Ethernet MAC address - must be unique on your network - MAC Reads T4A001 in hex
(unique in your network)
byte mac[] = { 0x54, 0x34, 0x41, 0x30, 0x30, 0x31 };Ορισμός  MAC Address, πρέπει να
είναι μοναδική στο δίκτυο  //For the rest we use DHCP (IP address and such)

EthernetClient client; Προσδιορίζουμε τη μεταβλητή Client για να διατηρηθεί η σύνδεσή
μας.
char server[] = "192.168.1.100"; // Η IP διέυθυνση του server μας στον οποίο στέλνουμε
τα δεδομένα
int  interval = 5000; // Wait between dumps

void setup() {

  Serial.begin(9600);
  Ethernet.begin(mac);
  Serial.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-\n");
  Serial.print("IP Address        : ");
  Serial.println(Ethernet.localIP());
  Serial.print("Subnet Mask       : ");
  Serial.println(Ethernet.subnetMask());
  Serial.print("Default Gateway IP: ");
  Serial.println(Ethernet.gatewayIP());
  Serial.print("DNS Server IP     : ");
  Serial.println(Ethernet.dnsServerIP());
}

void loop() {
float t = dht.readTemperature();

  // if you get a connection, report back via serial:
  if (client.connect(server, 80)) {  Προσπαθούμε να επιτύχουμε σύνδεση
    Serial.println("-> Connected");
    // Make a HTTP request:
    client.print( "GET /testserver/arduino_temperatures/add_data.php?");  GET request
    client.print("serial=");    Εδώ ορίζουμε μεταβλητές Serial και temperature με
σταθερές τιμές
    client.print( "288884820500006X" );
    client.print("&&");
    client.print("temperature=");
    client.print( t );
    Τα      δεδομένα      που      στέλνουμε      θα      εχουν      τη      μορφη
serial=288884820500006X&temperature=12.3

    client.println( " HTTP/1.1");
    client.print( "Host: " );    Κλείνουμε το HTTP GET request
    client.println(server)
    client.println( "Connection: close" );
    client.println();
    client.println();
    client.stop();
  }
  else {
    // you didn't get a connection to the server:
    Serial.println("--> connection failed/n");
```

```
  }

  delay(interval);
}
```

Εάν θέλουμε να στέλνουμε δεδομένα από περισσότερους αισθητήρες τότε στον κώδικα του Arduino για παράδειγμα αν θέλουμε να στέλνουμε και υγρασία θα πρέπει να προσθέσουμε τις εξής γραμμές κώδικα στο GET request:

```
client.print("temperature=");
client.print( t );
client.print("&&");
client.print("humidity=");
client.print( h );
```

Όπου h, έχουμε ορίσει στο Loop `float h = dht.readHumidity();`

Ωστόσο χρειάζονται αλλαγές και στην πλευρά της PHP και της mySQL. Όσον αφορά τη mySQL πρέπει να προσθέσουμε μια στήλη στον πίνακα για humidity. Τώρα όσον αφορά την PHP, στο αρχείο add_data.php πρέπει να προσθέσουμε και την υγρασία ως εξής:

```
$SQL = "INSERT INTO test3.sensors ( temperature,humidity) VALUES ('".$_GET["temperature"]."','".$_GET["humidity"]."')";
```

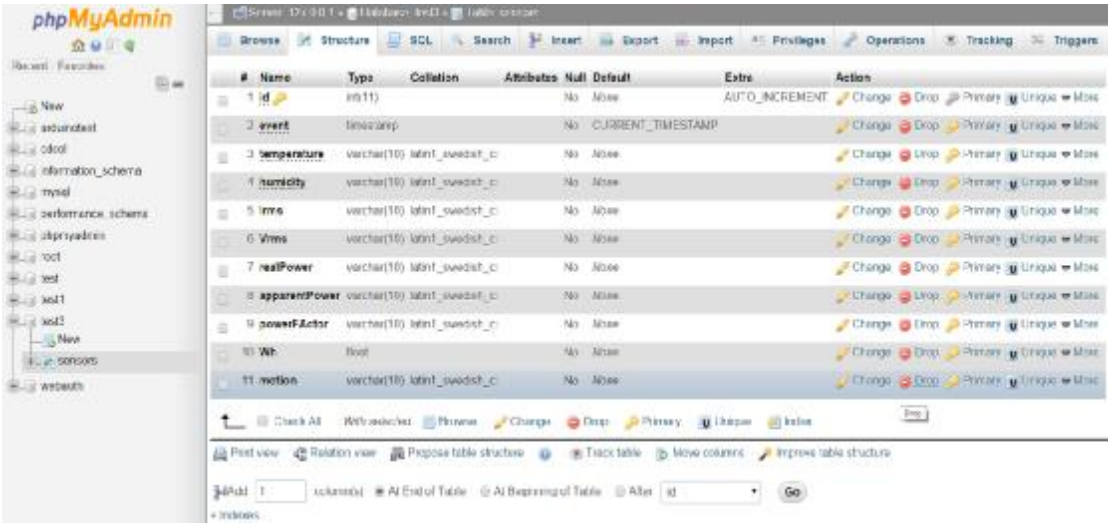
Το dbconnect.php δεν χρειάζεται κάποια αλλαγή. Τέλος τροποποίηση χρειάζεται το review_data.php.

Στο κομμάτι του πίνακα πρέπει να προσθέσουμε τα απαραίτητα για τη μέτρηση της υγρασίας:

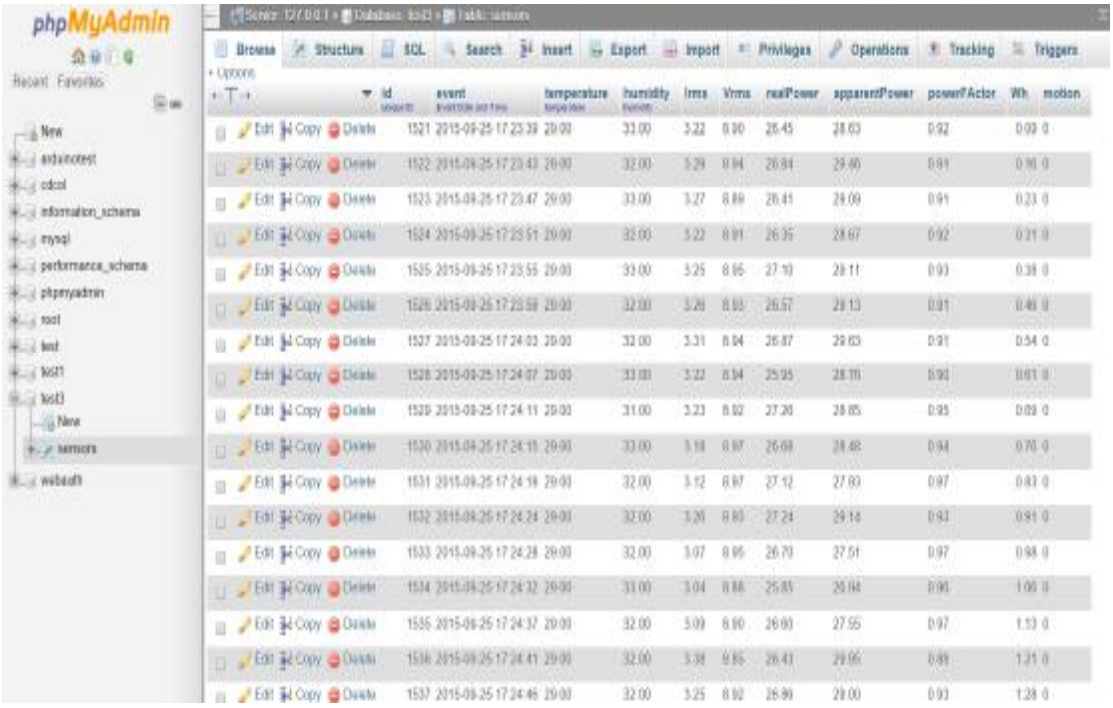
```
<body>
        <h1>House sensor measurements</h1>

    <table border="0" cellspacing="1" cellpadding="4" style="margin-top: 10px;">
      <tr>
            <td class="table_titles">ID</td>
            <td class="table_titles">Date and Time</td>
            <td class="table_titles">Temperature</td>
            <td class="table_titles">Humidity</td>
      </tr>
<?php
    // Retrieve all records and display them
    $result = mysql_query("SELECT * FROM sensors ORDER BY id DESC");

    // Used for row color toggle
    $oddrow = true;

    // process every record
    while( $row = mysql_fetch_array($result) )
    {
        if ($oddrow)
        {
            $css_class=' class="table_cells_odd"';
        }
        else
        {
            $css_class=' class="table_cells_even"';
        }
        $oddrow = !$oddrow;

        echo '<tr>';
        echo '   <td'.$css_class.'>'.$row["id"].'</td>';
        echo '   <td'.$css_class.'>'.$row["event"].'</td>';
        echo '   <td'.$css_class.'>'.$row["temperature"].'°C</td>';
```

```
        echo '    <td'.$css_class.'>'.$row["humidity"].'%</td>';
        echo '</tr>';
    }
?>
    </table>
```

Για περισσότερους αισθητήρες εργαζόμαστε αντίστοιχα. Εν τέλη η βάση δεδομένων του project μας ονομάστηκε test3 και ο πίνακας sensors. Στις παρακάτω φωτογραφίες φαίνεται η δομή της (Εικόνα 4.11), η μορφή του πίνακα στο phpmyadmin (Εικόνα 4.12) και στον browser του υπολογιστή (Εικόνα 4.13) και του κινητού (Εικόνα 4.14) καθώς και οι τελικοί κώδικες.



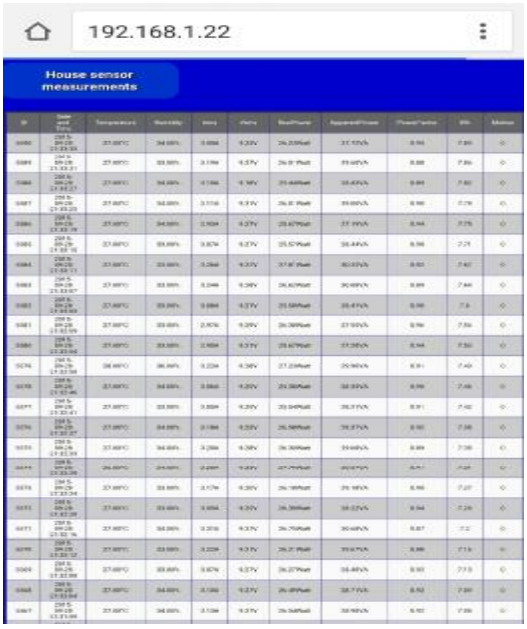
Εικόνα 4.11-Δομή Βάσης Δεδομένων



Εικόνα 4.12-Ο Πίνακας της βάσης δεδομένων

ID	Date and Time	Temperature	Humidity	Irms	Vrms	RealPower	ApparentPower	PowerFactor	Wh	Motion
5575	2015-09-28 21:32:33	27.00°C	34.00%	3.20A	9.20V	26.30Watt	29.68VA	0.89	7.35	0
5574	2015-09-28 21:32:29	26.00°C	34.00%	3.28A	9.33V	27.79Watt	30.57VA	0.91	7.31	0
5573	2015-09-28 21:32:24	27.00°C	33.00%	3.17A	9.20V	26.18Watt	29.18VA	0.90	7.27	0
5572	2015-09-28 21:32:20	27.00°C	33.00%	3.05A	9.25V	26.39Watt	28.22VA	0.94	7.24	0
5571	2015-09-28 21:32:16	27.00°C	34.00%	3.31A	9.27V	26.75Watt	30.68VA	0.87	7.2	0
5570	2015-09-28 21:32:12	27.00°C	33.00%	3.22A	9.26V	26.21Watt	29.67VA	0.88	7.16	0
5569	2015-09-28 21:32:08	27.00°C	33.00%	3.07A	9.27V	26.27Watt	28.40VA	0.92	7.11	0
5568	2015-09-28 21:32:04	27.00°C	34.00%	3.10A	9.23V	26.49Watt	28.71VA	0.92	7.09	0
5567	2015-09-28 21:31:59	27.00°C	34.00%	3.11A	9.27V	26.54Watt	28.98VA	0.92	7.05	0
5566	2015-09-28 21:31:55	28.00°C	33.00%	3.17A	9.25V	26.21Watt	29.29VA	0.89	7.02	0
5565	2015-09-28 21:31:51	27.00°C	33.00%	3.17A	9.20V	26.47Watt	29.18VA	0.91	6.98	0
5564	2015-09-28 21:31:47	27.00°C	32.00%	3.18A	9.24V	26.31Watt	29.41VA	0.89	6.94	0
5563	2015-09-28 21:31:43	27.00°C	33.00%	3.20A	9.18V	26.22Watt	29.38VA	0.89	6.91	0
5562	2015-09-28 21:31:38	28.00°C	35.00%	3.20A	9.24V	26.46Watt	29.54VA	0.90	6.87	0
5561	2015-09-28 21:31:35	27.00°C	33.00%	3.14A	9.24V	26.26Watt	28.98VA	0.91	6.83	0
5560	2015-09-28 21:31:31	27.00°C	34.00%	3.17A	9.15V	26.57Watt	28.95VA	0.92	6.8	0
5559	2015-09-28 21:31:26	27.00°C	33.00%	3.18A	9.24V	26.32Watt	29.38VA	0.90	6.76	0
5558	2015-09-28 21:31:22	27.00°C	33.00%	3.01A	9.24V	26.24Watt	27.79VA	0.94	6.72	0
5557	2015-09-28 21:31:18	27.00°C	34.00%	3.14A	9.22V	26.13Watt	28.96VA	0.91	6.69	0
5556	2015-09-28 21:31:14	27.00°C	33.00%	2.95A	9.16V	24.44Watt	27.01VA	0.90	6.65	0
5555	2015-09-28 21:31:09	27.00°C	33.00%	3.21A	9.12V	27.39Watt	30.08VA	0.91	6.62	0

Εικόνα 4.13-Τα δεδομένα της βάσης στον Browser του υπολογιστη



Εικόνα 4.14-Τα δεδομένα της βάσης στον Browser του κινητού

4.4 Γραφήματα

Στα πλαίσια της πτυχιακής με σκοπό τον καλύτερο έλεγχο και την επίγνωση της κατάστασης του σπιτιού, πέρα από τη βάση δεδομένων η οποία κρατάει τις μετρήσεις των αισθητήρων, ενδεικτικά καλούμε από τη βάση κάποιες από αυτές τις μετρήσεις με σκοπό την απεικόνιση τους σε γραφική παράσταση. Ο λόγος είναι η

καλύτερη κατανόηση των μετρήσεων με την αστικοποίηση τους και η πιο εύκολη αναζήτησή τους με βάση τον χρόνο.

Αφού δείξαμε σε προηγούμενη ενότητα πώς φτιάχνουμε μια βάση δεδομένων και πως μπορούμε να αποθηκεύσουμε μετρήσεις από το Arduino, ας δούμε πως μπορούμε να απεικονίσουμε αυτά τα δεδομένα. Αρχικά να αναφέρουμε πως για την επίτευξη αυτού του στόχου χρησιμοποιήθηκε η βιβλιοθήκη amCharts η οποία όπως είπαμε και σε προηγούμενη ενότητα μας παρέχει έτοιμες δομές γραφικών παραστάσεων με τη βοήθεια JavaScript.

Όπως στην περίπτωση της βάσης δεδομένων, έτσι και εδώ, πρέπει να φτιάξουμε ένα αρχείο PHP το οποίο να παίρνει τα δεδομένα από τη βάση και στη συνέχεια αρχεία HTML να μας αντλούν τα δεδομένα αυτά και να τα απεικονίζουν σε γραφικές παραστάσεις.

Το αρχείο PHP θα το ονομάσουμε data_all.php και θα περιέχει τον εξής κώδικα:

```php
<?php
// Connect to MySQL
$link = mysql_connect( 'localhost', 'root', '' );
if ( !$link ) {
  die( 'Could not connect: ' . mysql_error() );
}

// Select the data base
$db = mysql_select_db( 'test3', $link );
if ( !$db ) {
  die ( 'Error selecting database \'test3\' : ' . mysql_error() );
}

// Fetch the data
$query = "
  SELECT *
  FROM sensors
  ORDER BY event DESC   LIMIT 24  ";
$result = mysql_query( $query );

// All good?
if ( !$result ) {
  // Nope
  $message  = 'Invalid query: ' . mysql_error() . "\n";
  $message .= 'Whole query: ' . $query;
  die( $message );
}

// Print out rows
$prefix = '';
echo "[\n";
while ( $row = mysql_fetch_assoc( $result ) ) {
  echo $prefix . " {\n";
  echo '  "event": "' . $row['event'] . '",' . "\n";
   echo '  "temperature": "' . $row['temperature'] . '",' . "\n";
  echo '  "humidity": ' . $row['humidity'] . ',' . "\n";
  echo '  "Irms": "' . $row['Irms'] . '",' . "\n";
   echo '  "Vrms": "' . $row['Vrms'] . '",' . "\n";
   echo '  "realPower": "' . $row['realPower'] . '",' . "\n";
      echo '  "Wh": "' . $row['Wh'] . '",' . "\n";
  echo " }";
  $prefix = ",\n";
}
echo "\n]";

// Close the connection
mysql_close($link);
?>
```

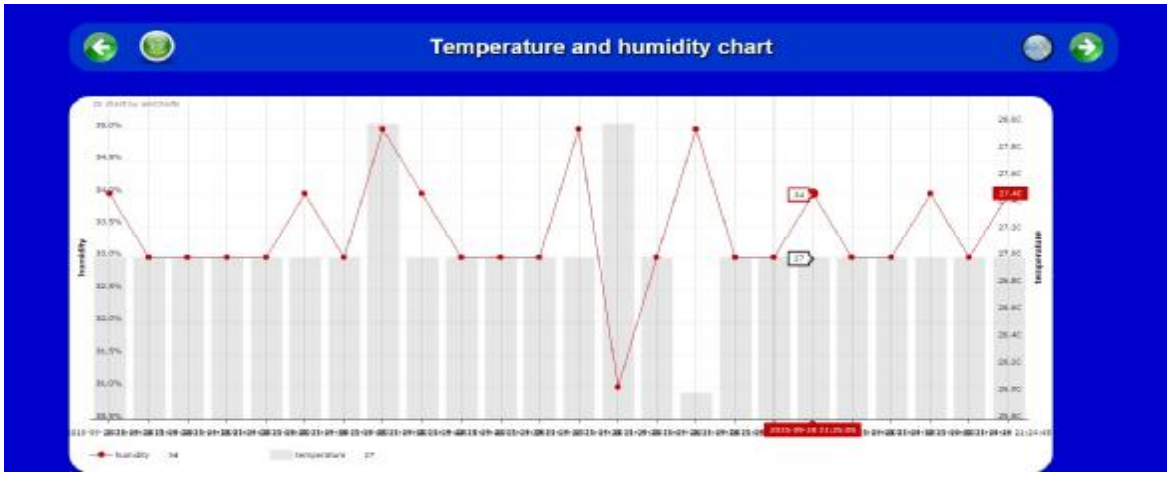
Στη συνέχεια για κάθε μία μεταβλητή όπως temperature,humidity κλπ, θα υπάρχει το αντίστοιχο αρχείο HTML που θα συνδέεται με το αρχείο PHP που φτιάξαμε και επικοινωνώντας με τη βάση, σε ένα browser θα μπορεί να μας εμφανίσει γράφημα των μετρήσεων. Όπως καταλαβαίνουμε δεν χρειάζεται κάποια τροποποίηση στον κώδικα του Arduino

Για παράδειγμα όσον αφορά την υγρασία και τη θερμοκρασία η HTML θα είναι:

```
<!DOCTYPE html>
<html>
<style>
h1  {       width:1400;
            margin-top:30px;
            margin-right:85px;
            margin-bottom:40px;
            margin-left:85px;
            font-weight: bold;
            font-style:normal;
            text-decoration: none;
            background-color:#0033CC;
            color:white;
            text-shadow: 2px 2px 4px #000000;
            border-radius: 30px;
            padding-top:20px ;
            padding-bottom:20px ;
            font-family: Arial;
            text-allign:center;
        }
body {
            color:#000000;
            background-color:#0000CC;
            text-align: center;
        }
</style>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>amCharts examples</title>
        </head>
        <body>
        <h1><a href="Irms.html">
        <img src="images/back.png" height="50" width="50" align="left" style="margin:-6px 0px 0px
20px;" />
        </a><a href="../home.html">
        <img src="images/home.png" height="70" width="70" align="left" style="margin:-18px 0px 0px
20px;" />
        </a>Temperature and humidity chart
        <a href="Wh.html">
        <img src="images/backr.png" height="50" width="50" align="right" style="margin:-6px 20px
0px 0px;" />
        </a>
        <a href="../review_data.php">
        <img src="images/mysql.png" height="50" width="50" align="right" style="margin:-6px 20px -
6px 0px;" />
        </a></h1>
        <link rel="stylesheet" href="style.css" type="text/css">
        <script src="../amcharts/amcharts.js" type="text/javascript"></script>
        <script src="../amcharts/serial.js" type="text/javascript"></script>
        <script>
AmCharts.loadJSON = function(url) {
  // create the request
  if (window.XMLHttpRequest) {
    // IE7+, Firefox, Chrome, Opera, Safari
    var request = new XMLHttpRequest();
  } else {
    // code for IE6, IE5
    var request = new ActiveXObject('Microsoft.XMLHTTP');
  }
  // load it
  // the last "false" parameter ensures that our code will wait before the
  // data is loaded
  request.open('GET', url, false);
  request.send();
  // parse adn return the output
  return eval(request.responseText);
};
  </script>
    <div id="chartdiv" style="width: 1400px; height: 600px; background-color:white;margin: 0px
200px 85px 85px;border-radius: 30px; "></div>
        <script>
            var chart;
            AmCharts.ready(function () {
            // SERIAL CHART
            var chartData = AmCharts.loadJSON('data_all.php');
                // SERIAL CHART
                chart = new AmCharts.AmSerialChart();
                chart.dataProvider = chartData;
  chart.categoryField = "event";
  chart.dataDateFormat = "JJ:NN:SS";
                chart.marginTop = 0;
                // AXES
                // category axis
```

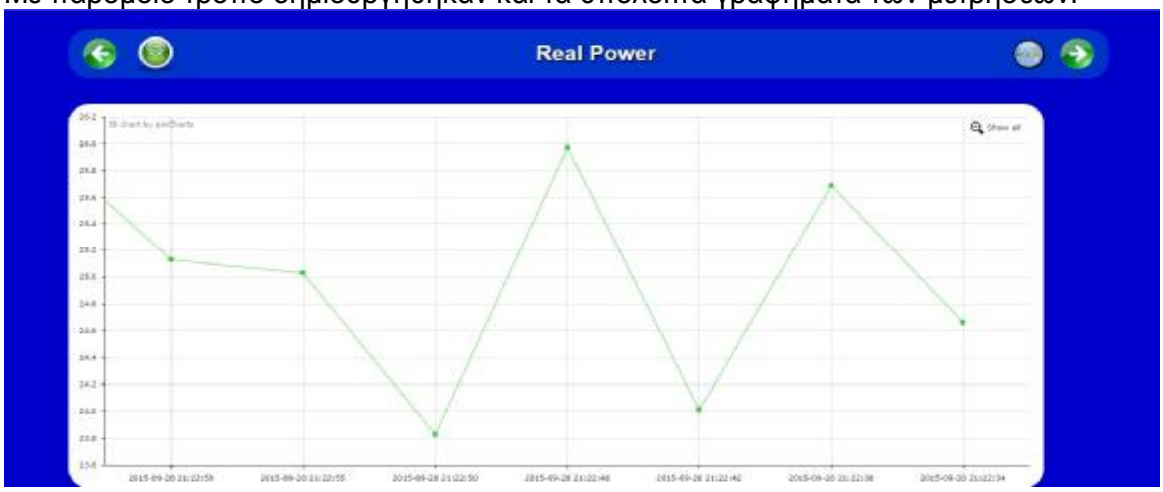
```
                chart.categoryAxis.dateFormats = true;
                // as we have data of different units, we create two different value axes
                // Duration value axis
                var durationAxis = new AmCharts.ValueAxis();
                durationAxis.title = "humidity";
                durationAxis.gridAlpha = 0.05;
                durationAxis.axisAlpha = 0;
                durationAxis.inside = true;
                // the following line makes this value axis to convert values to duration
                // it tells the axis what duration unit it should use. mm - minute, hh - hour...
                chart.addValueAxis(durationAxis);
                // Distance value axis
                var distanceAxis = new AmCharts.ValueAxis();
                distanceAxis.title = "temperature";
                distanceAxis.gridAlpha = 0;
                distanceAxis.position = "right";
                distanceAxis.inside = true;
                distanceAxis.unit = "C";
                distanceAxis.axisAlpha = 0;
                chart.addValueAxis(distanceAxis);
                // GRAPHS
                // duration graph
                var durationGraph = new AmCharts.AmGraph();
                durationGraph.title = "humidity";
                durationGraph.valueField = "humidity";
                durationGraph.type = "line";
                durationGraph.valueAxis = durationAxis; // indicate which axis should be used
                durationGraph.lineColor = "#CC0000";
                durationGraph.balloonText = "[[value]]";
                durationGraph.lineThickness = 1;
                durationGraph.legendValueText = "[[value]]";
                durationGraph.bullet = "round";
                durationGraph.bulletBorderColor = "#CC0000";
                durationGraph.bulletBorderAlpha = 1;
                durationGraph.bulletBorderThickness = 1;
                chart.addGraph(durationGraph);
                // distance graph
                var distanceGraph = new AmCharts.AmGraph();
                distanceGraph.valueField = "temperature";
                distanceGraph.title = "temperature";
                distanceGraph.type = "column";
                distanceGraph.fillAlphas = 0.1;
                distanceGraph.valueAxis = distanceAxis; // indicate which axis should be used
                durationAxis.unit = "%";
                distanceGraph.lineColor = "#000000";
                distanceGraph.lineAlpha = 0;
                chart.addGraph(distanceGraph);
                // CURSOR
                var chartCursor = new AmCharts.ChartCursor();
                chartCursor.zoomable = false;
                chartCursor.categoryBalloonDateFormat = "JJ";
                chartCursor.cursorAlpha = 0;
                chartCursor.valueLineEnabled = true;
                chartCursor.valueLineBalloonEnabled = true;
                chartCursor.valueLineAxis = distanceAxis;
                chart.addChartCursor(chartCursor);
                // LEGEND
                var legend = new AmCharts.AmLegend();
                legend.bulletType = "round";
                legend.equalWidths = false;
                legend.valueWidth = 120;
                legend.color = "#000000";
                legend.useGraphSettings = true;
                chart.addLegend(legend);
                // WRITE
                chart.write("chartdiv");
            });
        </script>
    </body>
</html>
```

Στις επόμενες εικόνες ακολουθούν τα γραφήματα που προκύπτουν από τις μετρήσεις των αισθητήρων θερμοκρασίας και υγρασίας (Εικόνα 4.15), από τη μέτρηση πραγματικής ισχύος (Εικόνα 4.16), τη μέτρηση Wh(Watt/h) (Εικόνα 4.17), από τον αισθητήρα ρεύματος (Εικόνα 4.18), και από τη μέτρηση της τάσης(Εικόνα 4.19).

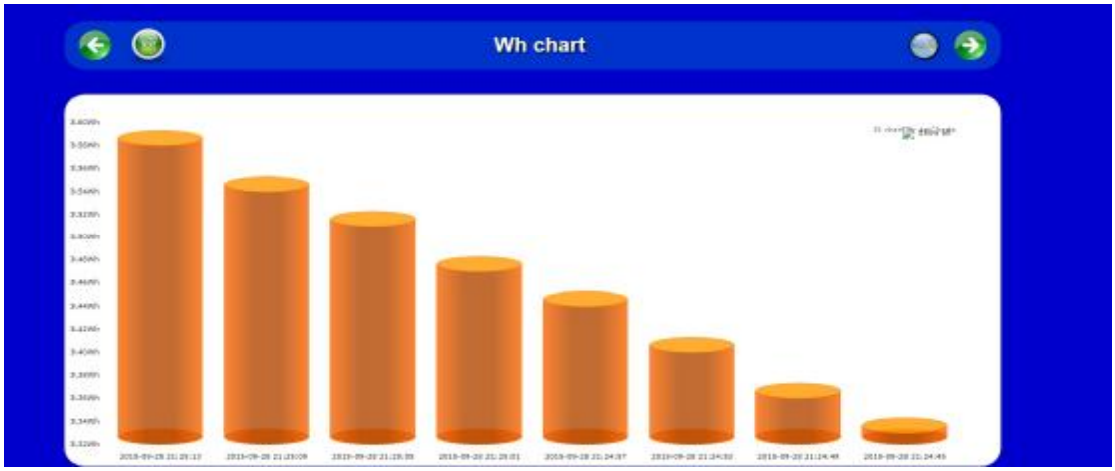


Εικόνα 4.15-Γράφημα θερμοκρασίας και υγρασίας

Με παρόμοιο τρόπο δημιουργήθηκαν και τα υπόλοιπα γραφήματα των μετρήσεων.

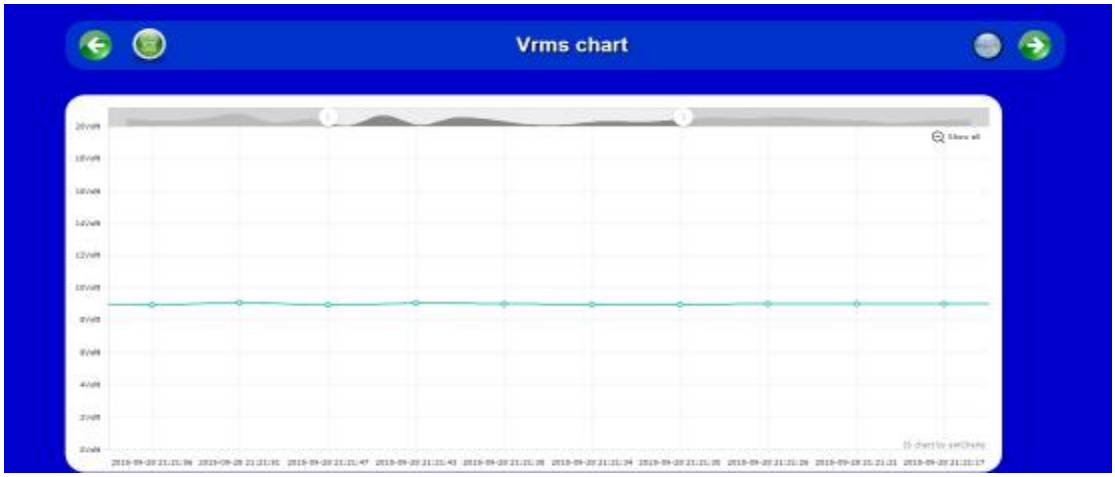


Εικόνα 4.16-Γράφημα Πραγματικής ισχύος



Εικόνα 4.17-Γράφημα Wh

Εικόνα 4.18-Γράφημα ρεύματος



Εικόνα 4.19-Γράφημα Τάσης

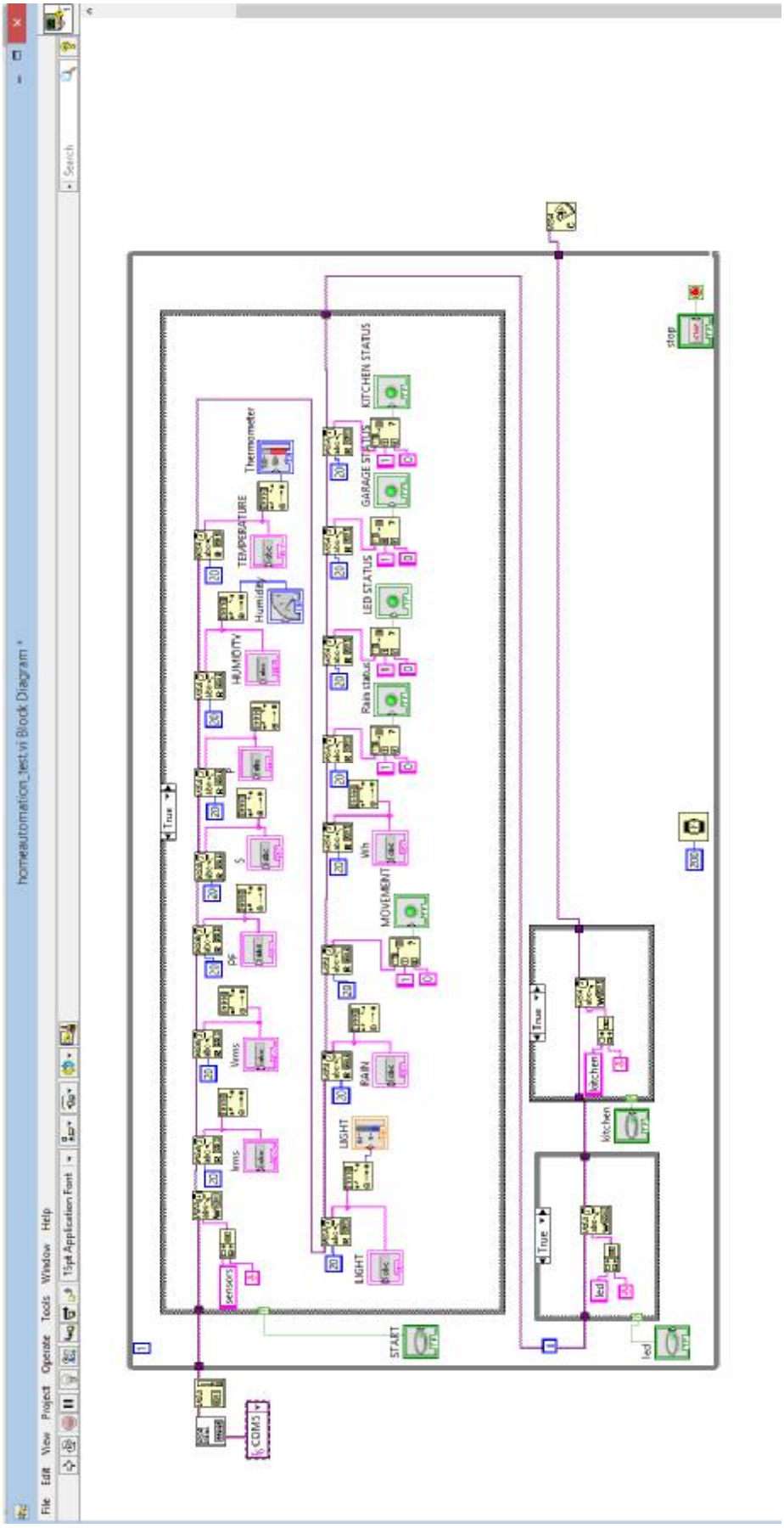
4.5 Προγραμματισμός LabVIEW

Για ένα επιπλέον τρόπο ελέγχου του συστήματός μας πέρα από την ιστοσελίδα, την οθόνη, τα buttons και τις φωνητικές εντολές, χρησιμοποιήσαμε το περιβάλλον προγραμματισμού σε γραφικό περιβάλλον LabVIEW. Παρακάτω φαίνεται το περιβάλλον στο οποίο ο χρήστης μπορεί να δει μετρήσεις των αισθητήρων του συστήματος καθώς επίσης έχει και τη δυνατότητα να επέμβει κάνοντας χειρισμούς με σκοπό να ενεργοποιήσει ή να απενεργοποιήσει κάποιες συσκευές (Εικόνα 4.20). Μπορεί να ελέγχει συγκεκριμένα δύο φωτιστικά, να δει μετρήσεις ενέργειας, θερμοκρασίας και υγρασίας και έχει ένδειξη σε περίπτωση που ανιχνευτεί κίνηση ή βροχή. Επίσης το LabVIEW διαθέτει και web interface μέσω πρωτοκόλλου HTTP, μεταφέροντας το περιβάλλον του στο διαδίκτυο. Η λειτουργία αυτή υποστηρίζεται μόνο από τον Internet Explorer ή διαφορετικά για ελάχιστους άλλους browsers μπορούμε να εισάγουμε το κατάλληλο plug-in (npapi). Ο χρήστης έχει τη δυνατότητα να συνδεθεί στο περιβάλλον του LabVIEW από οποιονδήποτε άλλο υπολογιστή βρίσκεται στο ίδιο δίκτυο έχοντας τη δυνατότητα απομακρυσμένου ελέγχου του συστήματος.

Εικόνα 4.20-Front Panel LabVIEW

Για φόντο τοποθετήθηκε μια τυχαία κάτοψη σπιτιού με σκοπό να έχουμε μια καλύτερη εικόνα των χειρισμών. Ακολουθεί εικόνα με το web interface του συστήματος μας μέσω LabVIEW στον Internet Explorer (Εικόνα 4.21).

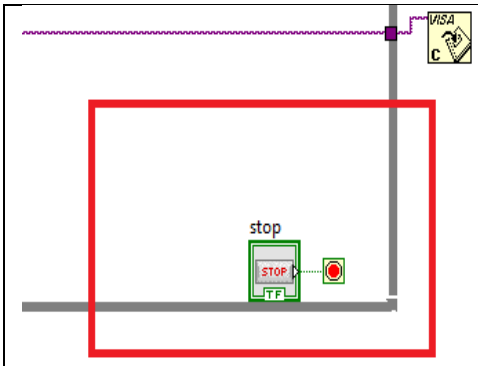
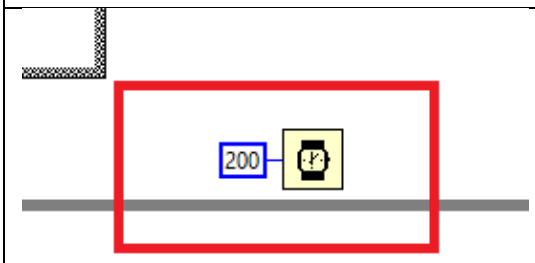
Εικόνα 4.21-Έλεγχος LabVIEW από Internet Explorer

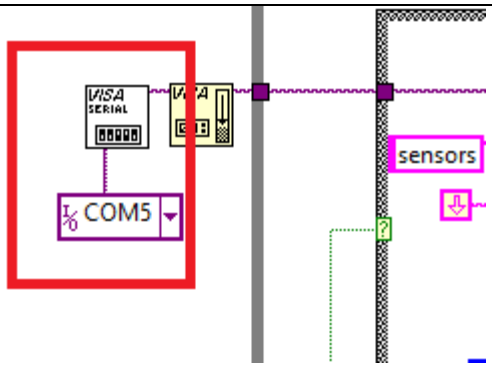
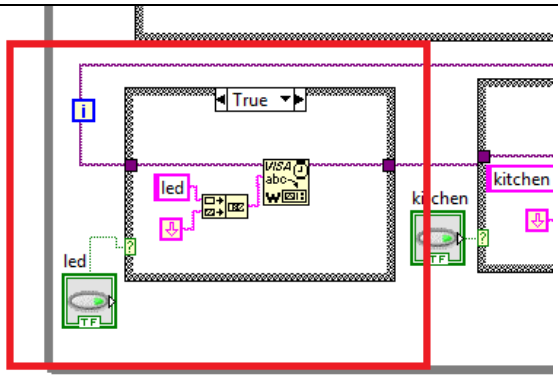


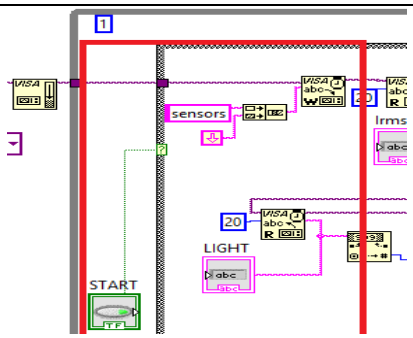
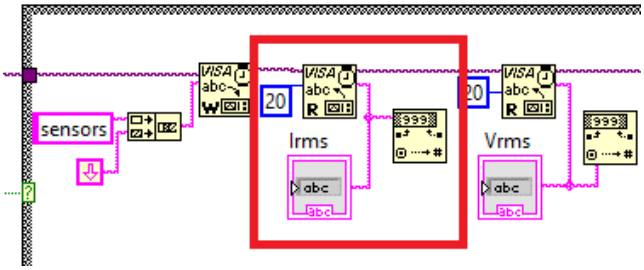
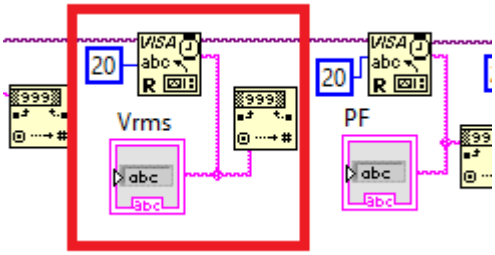
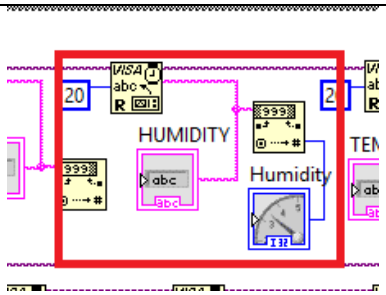
Εικόνα 4.22-Block Διάγραμμα συστήματος

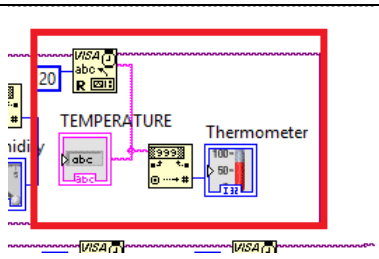
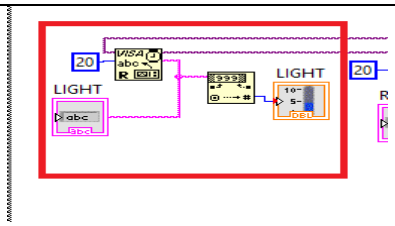
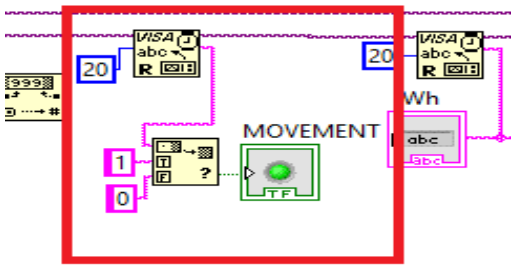
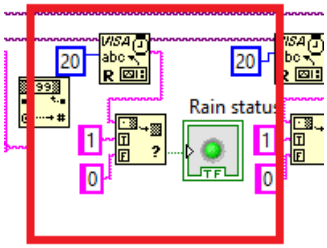
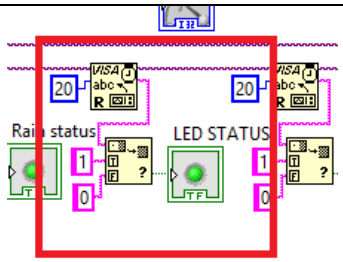
Στην προηγούμενη εικόνα (Εικόνα 4.22) μπορούμε να δούμε το πρόγραμμα που υλοποιήθηκε στο περιβάλλον του LabVIEW. Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο στο LabVIEW ο κώδικας αποτελείται από blocks, κάθε ένα από τα οποία έχει τη δική του λειτουργία. Όλος ο κώδικας τρέχει μέσα σε μία while loop μέσα στην οποία υπάρχουν συνθήκες true/false που ελέγχονται από buttons (στον κώδικα του Arduino εκφράζονται με συνθήκες if...else). Στον συγκεκριμένο κώδικα μας ενδιαφέρει να ελέγξουμε κάποια led που έχουν συνδεθεί στο Arduino και να δούμε σε indicators ή σε κάποιο γραφικό στοιχείο τις μετρήσεις των αισθητήρων μας. Η λογική που χρησιμοποιούμε, όπως θα δείξουμε πιο αναλυτικά παρακάτω για κάθε σημείο του κώδικα, είναι στην ουσία να στέλνουμε ένα αίτημα στο Arduino με τη μορφή string για κάθε δεδομένο που θέλουμε να προβάλουμε ή για κάθε led που θέλουμε να ελέγξουμε, και στη συνέχεια περιμένουμε από το Arduino να εκτελέσει το αίτημα αυτό και να στείλει το αντίστοιχο μήνυμα πίσω στο LabVIEW, το οποίο με τη σειρά του θα διαβάσει τα δεδομένα αυτά και θα τα απεικονίσει με όποιο τρόπο του έχουμε ορίσει εμείς. Το Arduino μπορεί να επικοινωνεί με το LabVIEW είτε σειριακά είτε μέσω της θύρας Ethernet. Στο σύστημά που υλοποιήσαμε επιλέξαμε την πρώτη περίπτωση.

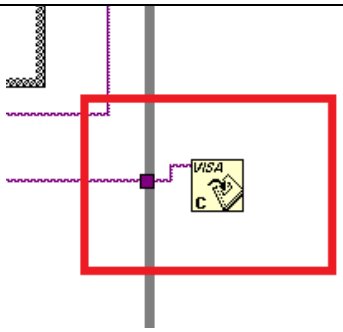
Ας δούμε αναλυτικά κάθε κομμάτι του κώδικα. Το γκρι πλαίσιο γύρω από το μεγαλύτερο μέρος του κώδικα είναι η while loop.

	Στο κάτω μέρος της while loop βάζουμε stop button για να μπορούμε να τη σταματήσουμε. Όσο εκτελείται είναι αδύνατο να φορτώσουμε άλλο κώδικα στο Arduino και δεν μπορούμε να ανοίξουμε το serial monitor. Πρέπει πρώτα να πατήσουμε το Stop.
	Επίσης βάζουμε ένα χρόνο καθυστέρησης σε ms έτσι ώστε να μειώσουμε την πιθανότητα σφαλμάτων από τη συνεχή ανάγνωση δεδομένων και του διαφορετικού χρονισμού των ρολογιών του υπολογιστή και του Arduino.

	Στην αρχή του προγράμματος βάζουμε τα απαραίτητα δεδομένα έτσι ώστε να επιτύχουμε τη σειριακή επικοινωνία με το Arduino. Έτσι ορίζουμε την COM5 στην οποία ακούει το Arduino μέσω του VISA Serial. Το VISA Flush I/O Buffer είναι ένας μετρητής δεδομένων ο οποίος χρησιμοποιήθηκε πειραματικά για να ελέγχουμε πόσα bit δεδομένων λαμβάνουμε με κάθε VISA Read. Δεν είναι απαραίτητο για τη σύνδεση μας.
	Το γκρι πλαίσιο αποτελεί μια συνθήκη true/false. Η συνθήκη αυτή ελέγχεται από το button "led". Όταν πατήσουμε το button "led" η συνθήκη γίνεται True και τότε το LabVIEW θα στείλει μέσω της VISA Write στο Arduino το string "led" και το Arduino με τη σειρά του θα εκτελέσει τις εντολές που του έχουμε ορίσει, στη συγκεκριμένη περίπτωση θα ανάψει το led στο pin 35 του Arduino Mega.Σε διαφορετική περίπτωση η συνθήκη παραμένει False.
	Αν προσέξουμε στο block διάγραμμα του προγράμματος υπάρχει μια μεγάλη συνθήκη true/false. Που ελέγχεται από το button "start".

	Πατώντας το "start" όπως είπαμε η συνθήκη γίνεται true. Τότε το LabVIEW μέσω της εντολής VISA Write γράφει το string "sensors" στη σειριακή θύρα του Arduino.
	Μετά το VISA Write αναμένεται από το Arduino μια απάντηση σε string. Αυτό το αναλαμβάνει η εντολή VISA Read. Εφόσον διαβάσει την τιμή του ρεύματος στη συγκεκριμένη περίπτωση, τη στέλνει σε έναν Indicator στον οποίο βλέπουμε στο Front Panel. Το Decimal String to Number δε το χρησιμοποιούμε. Χρειάζεται όμως στην περίπτωση που θέλουμε να απεικονίσουμε την μετρούμενη τιμή σε κάποιο γραφικό.
	Ομοίως, όπως στην περίπτωση του ρεύματος, προγραμματίσαμε το LabVIEW για να υπολογίσουμε και να έχουμε ένδειξη σε indicator μετρήσεις τάσης, πραγματικής και άεργου ισχύος, συντελεστή ισχύος, Wh καθώς και μετρήσεις του αισθητήρα βροχής.
	Στην περίπτωση της υγρασίας, πέρα από τον απλό indicator έχουμε προσθέσει στην έξοδο του Decimal String to Number ένα μετρητή Gauge έτσι ώστε να έχουμε καλύτερη απεικόνιση των μετρήσεων.

	Αντίθετα στην περίπτωση της θερμοκρασίας αντί για gauge έχουμε προσθέσει ένα θερμόμετρο.
	Για τη μέτρηση του φωτός έχουμε τοποθετήσει ένα μετρητή μπάρας.
	Στην πιθανότητα ανίχνευσης κίνησης, από την πλευρά του LabVIEW,έχουμε προγραμματίσει να ενεργοποιείται μια φωτεινή ένδειξη. Εάν ανιχνευθεί κίνηση το Arduino στέλνει ένα string "1" ενώ σε αντίθετη περίπτωση στέλνει "0". Ύστερα από το VISA Read το "0" ή το "1" περνά από το Mach True/False String το οποίο ανάλογα με το δεδομένο που διαβάστηκε επιλέγει αν θα ενεργοποιήσει την ένδειξη ή όχι.
	Με τον ίδιο τρόπο όπως προηγουμένως ελέγχουμε εάν υπάρχει ή όχι βροχή. Στην πρώτη περίπτωση το Arduino στέλνει string "1" και ανάβει η ένδειξη και αντίστροφα. ομοίως λειτουργούμε και όταν θέλουμε να ελέγχουμε εάν ένα led είναι αναμμένο ή μια συσκευή ενεργοποιημένη.
	Έλεγχος κατάστασης λειτουργίας led.

	Με το VISA close που τοποθετούμε έξω από τη while loop κλείνουμε τη σύνδεση με τη συνδεδεμένη συσκευή, δηλαδή στην περίπτωση μας με το Arduino.

Κεφάλαιο 5: Συμπεράσματα και Μελλοντικές Επεκτάσεις

Το project αυτό με βοήθησε να μάθω αρκετά πράγματα από τον κόσμο του Arduino. Γνώρισα και δούλεψα πάνω σε βάσεις δεδομένων και γλώσσες προγραμματισμού διαδικτύου, αισθητήρες, μετρήσεις, γραφικές παραστάσεις και χρησιμοποίησα αρκετά προγράμματα όπως το XAMPP και το LabVIEW.

Επίσης απέκτησα βασικές γνώσεις πάνω στα έξυπνα σπίτια και στους αυτοματισμούς.

Το τελικό αποτέλεσμα όπως εξηγήθηκε στην πορεία της πτυχιακής, είναι ένα σύστημα ελέγχου κάποιων συσκευών τοπικά μέσω buttons ή με τη χρήση lcd οθόνης, καθώς και απομακρυσμένα με τη βοήθεια της ιστοσελίδας που φτιάξαμε και με τη χρήση του LabVIEW. Οι αισθητήρες που χρησιμοποιήσαμε μας δίνουν τη δυνατότητα να δημιουργήσουμε διάφορα αυτοματοποιημένα σενάρια ανάλογα με τις ανάγκες μας. Τέλος η βάση δεδομένων και τα γραφήματα από τις μετρήσεις των χρησιμοποιούμενων αισθητήρων βοηθούν τον χρήστη να έχει μια εικόνα για την κατανάλωση ενέργειας των συσκευών του σπιτιού του, την ύπαρξη ή όχι κινητικότητας στο εσωτερικό του σπιτιού όσο αυτός απουσιάζει, την διακύμανση της θερμοκρασίας και της υγρασίας κλπ.

Όσον αφορά το κομμάτι του Arduino, μετά από αρκετές δοκιμές και αρκετές ώρες διαβάσματος και προγραμματισμού, έχω να πω πως είναι ένα σοβαρό εργαλείο υλοποίησης σεναρίων αυτοματισμού για αρχάριους αλλά και για επαγγελματίες. Οι διάφορες πλακέτες που κυκλοφορούν μπορούν να δώσουν ζωή σε πάρα πολλά project και σε πολλές ιδέες. Ο μόνος περιορισμός είναι τα χρήματα που μπορεί να διαθέσει κάποιος και η φαντασία του.

Στα πλαίσια του δικού μας project θα ήταν συνετό να γινόταν χρήση περισσότερων πλακετών Arduino, έτσι ώστε να μοιράζονται οι διαδικασίες και να αυξηθεί ο χρόνος απόκρισης του όλου συστήματος. Χρησιμοποιώντας αρκετούς αισθητήρες και σε συνδυασμό με τα σενάρια αυτοματισμού, τη βάση και την ιστοσελίδα, επειδή η μνήμη του Arduino είναι σχετικά μικρή και όλος ο κώδικας τρέχει σε ένα loop καθώς επίσης ανάλογα με τις επιμέρους διεργασίες προσθέτουμε κι εμείς κάποιες καθυστερήσεις, ο κώδικας γίνεται όλο και πιο αργός. Όπως και να' χει όμως, βελτιώσεις στον κώδικα μπορούν να γίνουν ανάλογα με την εξοικείωση που έχει ο κάθε ένας στον προγραμματισμό, αλλά και σε συνδυασμό με ένα δίκτυο Arduino όπου θα διαχωρίζεται το σύστημα σε επιμέρους διεργασίες και τα δεδομένα θα αποστέλλονται και θα ελέγχονται από ένα κεντρικό Arduino ή έναν υπολογιστή, το σύστημα θα μπορούσε να γίνει περισσότερο λειτουργικό.

Επίσης ας μην ξεχνάμε πως ένα τέτοιο σύστημα μπορεί να εφαρμοστεί σε οποιοδήποτε σπίτι, χωρίς προϋπάρχουσα υποδομή και με χαμηλό σχετικά κόστος. Στα θετικά επίσης να τονίσω και πάλι πως μπορούμε να ορίζουμε εμείς τα σενάρια αυτοματισμών που επιθυμούμε ανάλογα με τις ανάγκες μας. Και μάλιστα τι πιο ωραίο από μια κατασκευή ή ένα σύστημα DIY(Do It Yourself).

Τέλος, θα ήθελα να αναφέρω κάποια επιπλέον σενάρια που πρόκειται να εφαρμοστούν μελλοντικά στο σύστημα μας.

- Αρχικά όπως ανέγερα και προηγουμένως θα προθεθούν περισσότερες πλακέτες Arduino με σκοπό τον διαμοιρασμό των διαδικασιών.
- Θα μπορούσαμε να προσθέσουμε περισσότερους αισθητήρες για κάθε δωμάτιο του σπιτιού και να υλοποιήσουμε για κάθε δωμάτιο και από ένα ή περισσότερα σενάρια. Έστω δηλαδή πως είχε κάθε δωμάτιο τον δικό του αισθητήρα θερμοκρασίας και ανάλογα με τις ανάγκες του κάθε χώρου να υπήρχε και ο αντίστοιχος αυτοματισμός θερμοκρασίας.
- Θα ήταν δυνατό να εισάγουμε αισθητήρες για σύστημα πυροπροστασίας αλλά και πλημμύρας.
- Επίσης θα μπορούσαμε να αναπτύξουμε το σενάριο συναγερμού του αισθητήρα κίνησης ούτως ώστε να μας στέλνει το σύστημα μήνυμα, mail ή να μας καλεί στο κινητό μας με ένα ηχογραφημένο μήνυμα όταν λείπουμε από το σπίτι και αναληφθεί κάποια κίνηση.
- Επιπλέον για να αυξήσουμε την ασφάλεια του σπιτιού, αντί να χρησιμοποιούμε κλειδιά, θα μπορούσε να γίνει ένα σύστημα εισόδου με κωδικό pin, με μήνυμα από το κινητό μας ή με κάποιο μυστικό χτύπημα στην πόρτα.

Βιβλιογραφία

Βιβλια:

Π.Παπάζογλου και Σπ. Π.Λιωνής, Ανπαπτυξη εφαρμογών με το Arduino, Εκδόσεις Τζιόλα

L.ULLMAN, Εισαγωγή στις PHP 6 & MySQL 5 με εικόνες, Εκδόσεις Κλειδάριθμος

Διαδίκτυο:

https://el.wikipedia.org/wiki/JavaScript

https://el.wikipedia.org/wiki/Apache_HTTP_%CE%B5%CE%BE%CF%85%CF%80%CE%B7%CF%81%CE%B5%CF%84%CE%B7%CF%84%CE%AE%CF%82

https://el.wikipedia.org/wiki/XAMPP

https://www.arduino.cc/en/Main/ArduinoBoardMega2560#overview

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

https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/

http://www.greeceandroid.gr/dev/639-smart-home-android-arduino-6

http://bildr.org/2012/11/photoresistor-arduino/

http://grobotronics.com/photo-resistor-ldr-5mm.html

http://openenergymonitor.org/emon/node/58

https://learn.adafruit.com/photocells/overview

https://www.openhacks.com/uploadsproductos/rain_sensor_module.pdf

http://www.tweaking4all.com/hardware/arduino/arduino-ethernet-data-push/

https://startingelectronics.org/tutorials/arduino/ethernet-shield-web-server-tutorial/SD-card-IO/

http://www.instructables.com/howto/arduino/

http://fritzing.org/home/

https://en.wikipedia.org/wiki/Fritzing

http://www.amcharts.com/

Παραρτημα

Κώδικες

Όλοι οι Κώδικες βρίσκονται στο επισυναπτόμενο δίσκο CD. Οι κώδικες που αφορούν το Arduino μπορούν να διαβαστούν με το Arduino IDE που μπορούμε να κατεβάσουμε ελεύθερα από την επίσημη σελίδα http://arduino.cc. Γενικά όλα τα αρχεία οι html και php , για να διαβαστεί και να επεξεργαστεί ο κώδικάς τους μπορεί να χρησιμοποιηθεί το notepad++ το οποίο διατίθεται και αυτό δωρεάν από την επίσημη σελίδα του https://www.notepad-plus-plus.org.