ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη υποδομής για επικοινωνιακά συστήματα μικροελεγκτών με στόχο την αποθήκευση δεδομένων σε βάση δεδομένων.

Infrastructure development of microcontroller communication system for data storage in database.

Βλαχόπουλος Οδυσσέας

Εξεταστική Επιτροπή:

Καλαϊτζάκης Κωνσταντίνος, Καθηγητής - Επιβλέπων
Κολοκοτσά Διονυσία, Επίκουρη Καθηγήτρια
Μπάλας Κωνσταντίνος, Αναπληρωτής Καθηγητής

Χανιά 2013

ΠΡΟΛΟΓΟΣ

«Ο κόσμος μόνο όταν τον μοιράζεσαι υπάρχει»
Τάσος Λειβαδίτης

Η παρούσα εργασία δεν θα ήταν δυνατή χωρίς τη στήριξη της οικογένειάς μου, των φίλων και συντρόφων μου όλα αυτά τα χρόνια σπουδών και δουλειάς.

Ιδιαίτερα για τον καιρό εκπόνησης της ίδιας της εργασίας, θα χρωστάω για πάντα στον κολλητό φίλο Ζάκο που με φιλοξένησε στο σπίτι του όσο χρειάστηκε. Τα καλύτερά μας βράδια. Ως τώρα.

Η ίδια η φύση της ενασχόλησης με το παρόν θέμα έχει πολλές πτυχές που αναλύονται παρακάτω. Ελπίζω να προσφέρω, καταθέτοντας ένα πολύ μικρό λιθαράκι, στην πρόοδο, την ευτυχία και τη ζωή.

Χανιά/Αθήνα 2012-13

ΠΕΡΙΕΧΟΜΕΝΑ

3. ΕΦΑΡΜΟΓΕΣ-ΠΑΡΑΓΩΓΗ

3.1 Ελεύθερο/Ανοιχτό Λογισμικό και Υλικό
[Free/Open Source Software-Hardware – Creative Commons License]

3.2 Επιστημονικές εφαρμογές

3.3 Ατομικές εφαρμογές

4. ΣΥΜΠΕΡΑΣΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ 1: κώδικες

ΠΑΡΑΡΤΗΜΑ 2: βιβλιογραφία

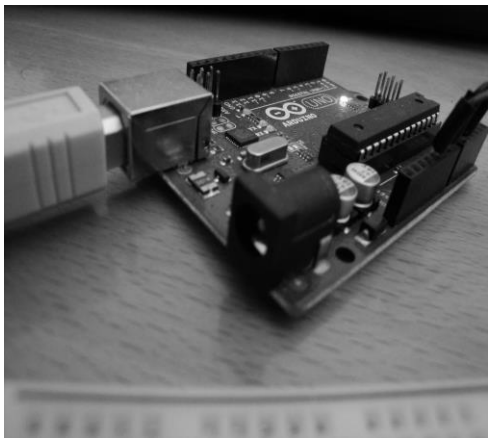
ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία έχει ένα εντυπωσιακά ευρύ φάσμα δυνατοτήτων, μια τεράστια ποικιλία εσωτερικών λειτουργιών, μια ατελείωτη σειρά εφαρμογών σε δεκάδες νευραλγικούς τομείς από την ίδια την πρωτογενή παραγωγή μέχρι την ατομική διασκέδαση.

Ξεκινώντας από τα ίδια τα υλικά της δομής που έχει παραχθεί, τα παραπάνω γίνονται πολύ εμφανή και ιδιαίτερα ενδιαφέροντα.

Σκοπός της παρούσας εργασίας είναι κατά τον ίδιο τον τίτλο να αναπτυχθεί μια δομή με μικροελεγκτές –εν προκειμένω και όπως θα αναλυθεί γιατί αργότερα- με χρήση της ηλεκτρονικής πλατφόρμας Arduino, μέσω της οποίας μπορούν να συλλεχθούν και να αποθηκευτούν δεδομένα από αισθητήρες ή άλλα όργανα, σε Βάση Δεδομένων SQL.

Στο συγκεκριμένο πρόβλημα εξετάζεται ιδιαίτερα μια συγκεκριμένη πτυχή: Η ενιαιοποίηση του συστήματος όσον αφορά την ίδια την επικοινωνία με τη Βάση Δεδομένων των πολλαπλών Arduino ανεξάρτητα με τον τρόπο σύνδεσής τους. Διαφορετικά, και πιο συγκεκριμένα, μεθοδεύεται η απλοποίηση και αυτοματοποίηση από πλευράς προγραμματισμού, όσο το δυνατόν περισσότερο, σειριακής (Serial) και Ethernet σύνδεσης των διάφορων Arduino με το server που χρησιμοποιείται προκειμένου να αποθηκεύουμε στη Βάση Δεδομένων.



Εικόνα 1: Arduino Uno

Ξεκινώντας από το μεγάλο θέμα της εξοικονόμησης ενέργειας σε διάφορα περιβάλλοντα, χρησιμοποιήθηκαν μικροελεγκτές σε πολύ ενδιαφέρουσες διατάξεις όπως αυτές των διάφορων Arduino –εδώ χρησιμοποιήθηκε Arduino Uno- προκειμένου να ελέγχονται με αισθητήρες οι παραμέτροι που μας ενδιαφέρουν όπως θερμοκρασία, υγρασία, φως κλπ, να τις αποθηκεύουμε για επεξεργασία και χρήση και όπως αναλύεται αργότερα μέσα από μια σειρά διεργασίες να πετύχουμε διάφορα αποτελέσματα.

Στην παρούσα εργασία παρουσιάζεται κυρίως ο τρόπος – η τεχνική – το προγραμματιστικό μέρος της εσωτερικής διαδικασίας του δεδομένου προβλήματος.

Οι δυσκολίες που απαντήθηκαν κατά τη διάρκεια της εκπόνησης της εργασίας δεν ήταν αμελητέες. Η αποσφαλμάτωση του κώδικα δυσχεραίνει λόγω εργαλείων και οι βλάβες του hardware πιθανές και δύσκολες στον εντοπισμό. Κατά τη διάρκεια των πειραμάτων, η μία Ethernet Shield απέκτησε κάποιο πρόβλημα που δε φαινόταν και η σύγχυση με πιθανό σφάλμα στον κώδικα είναι μεγάλη.

Πολύ περισσότερο οι λεπτομέρειες χρήσης και κατάλληλων αλλαγών στα εργαλεία, το πλήθος των συνεργατικών διεργασιών απαιτεί πολύ καλή αντίληψη σε βάθος των πρωτοκόλλων, των δικτύων, του πηγαίου κώδικα των εργαλείων και έρευνα σε όλα τα documentation.

Όσον αφορά τα ίδια τα υλικά που χρησιμοποιήθηκαν –Arduino Uno Rev3, Ethernet Shield-, όντας αρκετά περίπλοκα, πρέπει να ελέγχονται με αντικατάσταση αν κάτι δεν λειτουργεί.

Επίσης όσον αφορά τις προγραμματιστικές πλατφόρμες, η μεταξύ τους συμβατότητα, οι συγκεκριμένες εκδόσεις, το ίδιο το λειτουργικό σύστημα, αποτελούν ένα πάζλ που χρήζει διαρκούς προσοχής και ελέγχου. Στη συγκεκριμένη διπλωματική εργασία, μετά από επίπονες προσπάθειες, χρησιμοποιήθηκαν συγκεκριμένα εργαλεία όπως αναλύεται παρακάτω.

Στο 1ο Κεφάλαιο [ΕΙΣΑΓΩΓΗ] παρουσιάζονται τα υλικά που χρησιμοποιήθηκαν, δηλαδή το hardware και το προγραμματιστικό κομμάτι μαζί με τις πλατφόρμες τους, δηλαδή το software.

Στο 2ο Κεφάλαιο [ΥΛΟΠΟΙΗΣΗ ΥΠΟΔΟΜΗΣ] παρουσιάζεται το σύνολο των προγραμμάτων-υπηρεσιών που χρησιμοποιήθηκαν για το στήσιμο της παρούσας εργασίας. Εξετάζεται ο κεντρικός διακομιστής-server WAMP, με τα στοιχεία που το συναποτελούν. WAMP ονομάζεται ο συνδυασμός Windows/ Apache/PHP/MySQL, που είναι η πιο δημοφιλής πλατφόρμα εκτέλεσης ιστοσελίδων.

Ο WAMP Server έχει και το εργαλείο ελεύθερου επίσης λογισμικού, phpMyAdmin, το οποίο χειρίζεται τις Βάσεις Δεδομένων με τη χρήση κάποιου προγράμματος περιήγησης (web browser) στο Διαδίκτυο.

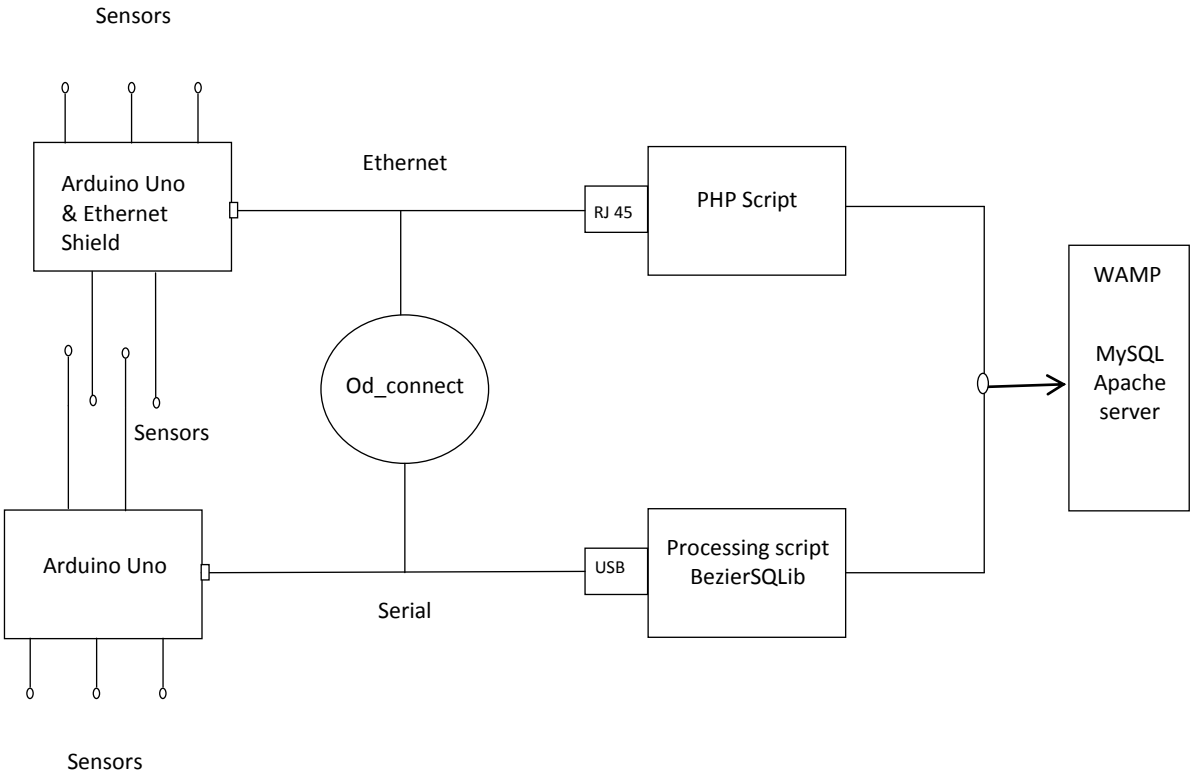
Εδώ χρησιμοποιήθηκε αποκλειστικά ο Mozilla Firefox.

Στη συνέχεια παρουσιάζεται αναλυτικά η ίδια η Βάση Δεδομένων και τα περιβάλλοντα προγραμματισμού των μικροελεγκτών Arduino, τόσο για το εσωτερικό του μικροελεγκτή όσο και για τον υπολογιστή με τον οποίο επικοινωνεί [Processing]. Χρησιμοποιήθηκαν Windows 7 κατά τη διάρκεια της εργασίας για τον κύριο λόγο ότι τα διάφορα OS Linux που θα προτιμούνταν χρήζουν ειδικής μεταχείρησης κάθε ένα διαφορετικό, και τα εργαλεία που χρησιμοποιούνται επίσης, δυσκολεύοντας σχετικά το επιθυμητό αποτέλεσμα για το μέσο χρήστη.

Παρόλα αυτά, τα περισσότερα συστήματα ανάπτυξης και υποστήριξης μικροελεγκτών περιορίζονται στα Windows, κάτι που αναβαθμίζει πολύ το σύνολο των δυνατοτήτων του Arduino αλλά και όλων όσων αναπτύσουν το ελεύθερο λογισμικό και όχι μόνο.

Μαζί με τα παραπάνω παρουσιάζεται η κατασκευασμένη για τις ανάγκες της εργασίας βιβλιοθήκη που εξυπηρετεί από κοινού τη Serial και Ethernet σύνδεση Arduino-Server και απλοποιεί τη διαδικασία ένταξης πολλαπλών Arduino στη όποια μελέτη-εφαρμογή. Ταυτόχρονα στο ίδιο κεφάλαιο παρουσιάζεται αναλυτικά και η διαδικασία-μοντελοποίηση της Σειριακής και Ethernet σύνδεσης ξεχωριστά.

Στο 3ο Κεφάλαιο [ΕΦΑΡΜΟΓΕΣ - ΠΑΡΑΓΩΓΗ] καταγράφεται η πραγματική, η υλική πλευρά της εργασίας, αλλά και η βαθύτερη αξία που εντοπίζεται στον τρόπο και τα μέσα με τα οποία υλοποιήθηκε και είναι προσβάσιμη. Οι δύο αυτές πλευρές όπως θα αναδειχτεί έχουν άμεση συσχέτιση. Τόσο η πλευρά του Hardware (το ίδιο το Arduino δηλαδή) όσο και η πλευρά του Software (οι εφαρμογές-πλατφόρμες προγραμματισμού) αποτελούν κομμάτι της πιο άξιας κληρονομιάς πνευματικής εργασίας, των open source και free Software / Hardware.

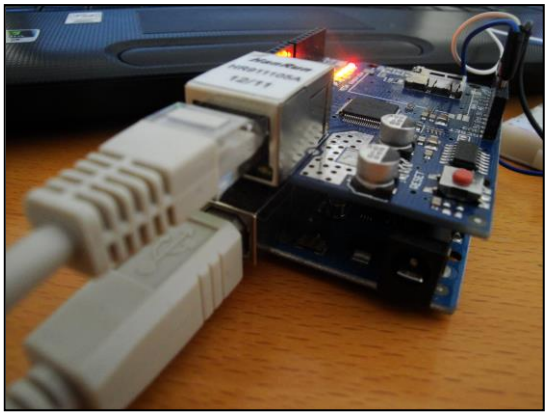


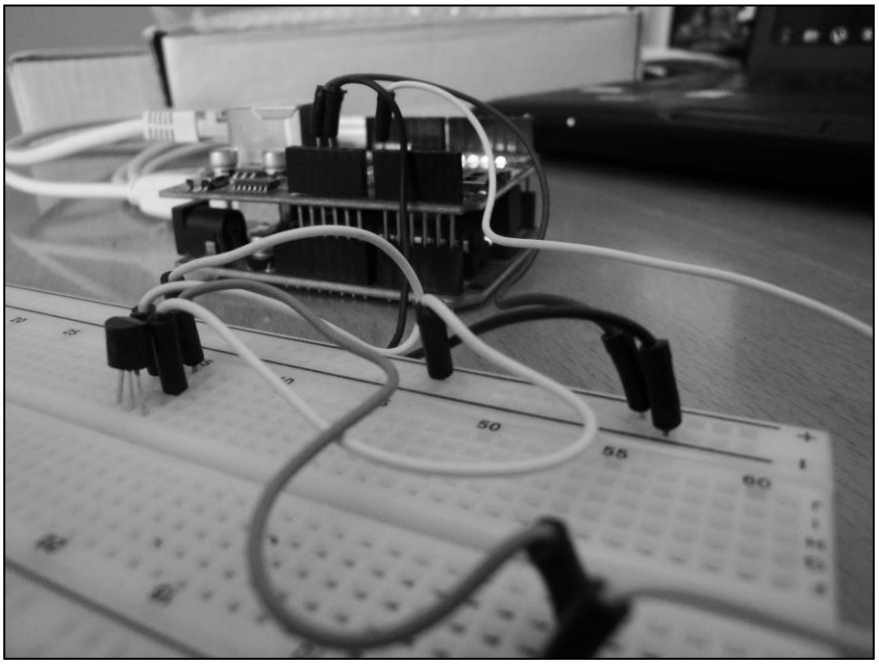
[Σχηματοποίηση της διπλωματικής εργασίας]

Αυτό έχει από μόνο του τεράστια σημασία αλλά έχει και απόλυτη συνάφεια με τις εφαρμογές που στηρίζονται από την ίδια την εργασία και τα εργαλεία που χρησιμοποιήθηκαν, αλλά και την παραγωγή την ίδια, μέχρι τις επιστημονικές εφαρμογές, τις εφαρμογές ασφαλείας, μετρήσεων κλπ. Για τον ίδιο τον άνθρωπο που παράγει με αυτά

τα μέσα υπό οποιεσδήποτε συνθήκες και για όποιο λόγο, και για την ίδια την αξία χρήσης, οι ατομικές εφαρμογές αποτελούν εργασία ή/και διασκέδαση ακόμα και αν αποτελούν κομμάτι μιας συλλογικής προσπάθειας-εργασίας. Το ανοιχτό και το ελεύθερο λογισμικό αποτελεί το μέλλον της ίδιας της δημιουργικότητας και εξέλιξης.

Τέλος, στο 4ο Κεφάλαιο [ΣΥΜΠΕΡΑΣΜΑΤΑ] επιχειρείται μια καταγραφή των συμπερασμάτων που βγήκαν κατά τη διάρκεια της εκπόνησης της εργασίας αυτής αλλά και για τη μελλοντική της χρήση.





Εικόνες 2,3,4 : Arduino Uno & Ethernet Shield με LM35

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

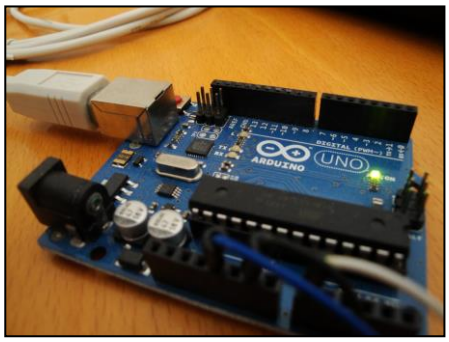
1.1

Υλικά: Μικροελεγκτές, Πλατφόρμα Arduino, Αισθητήρες

Τη ραχοκοκκαλιά της εργασίας αποτελεί ο ίδιος ο μικροελεγκτής. Χρησιμοποιήθηκε η πλακέτα Arduino Uno Revision 3 με μικροελεγκτή ATmega328. Προκειμένου να καταγραφούν οι μετρήσεις από αισθητήρες ή οποιαδήποτε άλλη λειτουργικότητα πρέπει να μεσολαβήσει ο μικροελεγκτής. Είτε για να μετατρέπει την αναλογική έξοδο ενός αναλογικού αισθητήρα σε ψηφιακό και άρα επεξεργάσιμο μέγεθος, είτε να ελέγχει ψηφιακές εισόδους/εξόδους, ο μικροελεγκτής προγραμματίζεται όπως χρειάζεται.

Η τροφοδοσία του Arduino γίνεται με μπαταρία(7-12V), USB(5V) ή AC-to-DC adapter από τον ακροδέκτη-Pin VIN Power (7-12V με όρια ανοχής 6-20V) . Η περιοχή λειτουργίας πρέπει να είναι 7-12 Volts. Τροφοδοσία με λιγότερα από 7V μπορεί να οδηγήσει στην υπολειτουργία της διάταξης με αισθητήρες κλπ από την πλακέτα, ενώ πάνω από 12V μπορεί να οδηγήσει σε υπερθέρμανση του ρυθμιστή τάσης και πιθανή βλάβη του ίδιου του Arduino. Πρέπει να υπολογίζεται κατά περίπτωση η κατανάλωση ρεύματος σε ολόκληρη τη διάταξη που εξυπηρετεί το Arduino και ανάλογα να δίνεται αρκετή τάση προκειμένου να λειτουργούν σωστά.

Το Arduino Uno rev3 βασίζεται στο μικροελεγκτή ATmega328 ο οποίος υποστηρίζει 14 ψηφιακές εισόδους/εξόδους και 6 αναλογικές εισόδους. Παρέχει επίσης μεταξύ άλλων ειδικών χαρακτηριστικών, δύο εξόδους τροφοδοσίας: 5V και 3.3V, pins γείωσης (GND).



Εικόνα 5: Arduino Uno & LM35

Υπάρχουν αρκετά διαφορετικά Arduino boards με διαφορετικούς μικροελεγκτές. Τα πλήρη χαρακτηριστικά και η ακριβής οργάνωση του κάθε Arduino είναι ανοιχτά και προσβάσιμα στο διαδίκτυο για επέκταση λειτουργικότητας και αναπαραγωγή του είτε από μη

επαγγελματίες είτε απλά για εξοικείωση προς χρήση. Για κάθε διαφορετική χρήση το Arduino board πρέπει να προσαρμόζεται αντίστοιχα.

Πίνακας 1: Χαρακτηριστικά Arduino Uno rev3

Μικροελεγκτής	ATMEGA328
Τάση λειτουργίας	5V
Τάση εισόδου(συνιστάται)	7-12V
Όρια τάσης εισόδου	6-20V
Ψηφιακοί ακροδέκτες I/O	14(6 εκ των οποίων PWM έξοδο)
Αναλογικοί ακροδέκτες εισόδου	6
Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη	40mA
Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V	50mA
Μνήμη Flash	32KB(ATMEGA328)
Μνήμη SRAM	2KB(ATMEGA328)
Μνήμη EEPROM	1KB(ATMEGA328)
Ταχύτητα ρολογιού	16MHz

Η πειραματική διάταξη που χρησιμοποιήθηκε ενδεικτικά, είναι η εξής: Το Arduino συνδέεται με τον υπολογιστή είτε σειριακά(USB) είτε μέσω Ethernet (Ethernet Shield) περνώντας τις μετρήσεις αναλογικού αισθητήρα θερμοκρασίας LM35D πάνω σε Mini Breadboard 170 tie point. Η τροφοδοσία της πλακέτας με τον αισθητήρα γίνεται από την την ίδια την πλακέτα του Arduino στα 5V.

Αισθητήρες είναι ηλεκτρονικές συσκευές που μετρούν μια φυσική μεταβλητή, όπως το φως ή η θερμοκρασία και το μετατρέπουν σε τάση. Εξαιτίας αυτής της διαδικασίας μετατροπής μιας μορφής ενέργειας σε μία άλλη οι αισθητήρες αναφέρονται επίσης ως μετατροπείς.

Αισθητήρες μπορούν γενικά να ταξινομηθούν σε δύο κατηγορίες: ψηφιακούς αισθητήρες και αναλογικούς αισθητήρες.

Η έξοδος ενός ψηφιακού αισθητήρα μπορεί να είναι μία από τις δύο πιθανές καταστάσεις . Είτε ON (1) συχνά +5 V, ή OFF (0), 0V. Οι περισσότεροι ψηφιακοί αισθητήρες λειτουργούν με ένα όριο. Αν η εισερχόμενη μέτρηση είναι κάτω από το όριο, ο αισθητήρας θα εμφανίσει τη μία κατάσταση, εάν είναι πάνω από το όριο, ο αισθητήρας θα εμφανίσει την άλλη κατάσταση.

Σε αντίθεση με ένα ψηφιακό αισθητήρα , η έξοδος ενός αναλογικού αισθητήρα μπορεί να λάβει οποιαδήποτε πιθανή τιμή σε μια δεδομένη περιοχή. Αντί να είναι σε θέση να εναλλάσσεται μόνο μεταξύ δύο καταστάσεων όπως ένας ψηφιακός αισθητήρας, ο αναλογικός αισθητήρας μπορεί να εξάγει μια τεράστια ποικιλία τιμών.

1.2

Server, Περιβάλλοντα Προγραμματισμού

Προκειμένου να καταγραφούν οι μετρήσεις σε Βάση Δεδομένων, οργανώθηκε ένα ολοκληρωμένο σύστημα με Εξυπηρετητή/Server και όλες τις διεπαφές που χρειάζονται για τις επιμέρους επικοινωνίες.

Το λογισμικό που χρησιμοποιεί το Arduino ή αλλιώς το IDE [integrated development environment] περιλαμβάνει επεξεργαστή κειμένου με κατάλληλα εργαλεία και διευκολύνσεις προγραμματισμού, μεταγλωτιστή[compiler], μηχανισμό μεταφοράς κώδικα στο Arduino [bootloader] και δυνατότητες διαχείρισης βιβλιοθηκών και αρχείων. Η γλώσσα στην οποία έχει γραφτεί είναι Java και μπορεί να λειτουργήσει σε όλα τα κατάλληλα περιβάλλοντα και λειτουργικά συστήματα.

Η βάση του περιβάλλοντος ανάπτυξης είναι ουσιαστικά η χαμηλότερη γλώσσα C βασιζόμενη στην οποία αναπτύχθηκε η Wiring, η οποία στη συνέχεια και προσαρμοσμένη στις ειδικές ανάγκες της πλακέτας εξελίχθηκε στην Processing. Η Processing λοιπόν αποτελεί τη γλώσσα που χρησιμοποιείται για τους μικροελεγκτές Arduino και την επικοινωνία τους με τον υπολογιστή. Ο Compiler που χρησιμοποιείται είναι ο AVR-GCC με βασική βιβλιοθήκη C την AVR libc.

Συνοπτικά η διαδικασία περιγράφεται ως εξής:

- Η πλευρά του υπολογιστή-δέκτη των μετρήσεων χρησιμοποιεί τον WAMP Server, τον οποίο προσαρμόζουμε στις ανάγκες μας. [Κεφάλαιο 2.1]

- Η Βάση Δεδομένων οργανώνεται όπως χρειάζεται και περιγράφεται αναλυτικά [Κεφάλαιο 2.2] . Με τη βοήθεια της PHP και του phpMyAdmin μεταφέρονται και οργανώνονται αντίστοιχα στο Server και τη Βάση Δεδομένων οι μετρήσεις. [Κεφάλαιο 2.3]

- Ο μικροελεγκτής προγραμματίζεται έτσι ώστε να λαμβάνει μετρήσεις από τους αισθητήρες. [Κεφάλαιο 2.4]

- Στη συνέχεια, ανάλογα με την περίπτωση σύνδεσης με υπολογιστή Σειριακά[USB] ή Ethernet προγραμματίζεται αναλόγως τόσο η πλευρά του υπολογιστή με τη γλώσσα και πλατφόρμα προγραμματισμού Processing είτε κατευθείαν σύνδεση για τη σειριακή περίπτωση. [Κεφάλαιο 2.4]

Γενικά, ως ενδεχόμενο προγραμματισμού για πολλαπλά Arduino, δεν έχει γίνει με ενιαίο τρόπο αλλά βολεύει με επαναλαμβανόμενο κώδικα, μιας και κάθε Arduino έχει άγνωστη και απίθανα ταυτόσημη λειτουργικότητα. Δηλαδή ποια pins θα χρησιμοποιηθούν, με τι τρόπο κλπ.

1.3

Διασύνδεση Υλικού: Serial – Ethernet

Στην παρούσα εργασία εξετάζονται δύο τρόποι διασύνδεσης του υλικού, δηλαδή των Arduino και του υπολογιστή-εξυπηρετητή, τη Σειριακή/Serial σύνδεση που είναι με USB και την Ethernet σύνδεση που χρειάζεται Ethernet Shield στο Arduino με την αντίστοιχη καλωδίωση.

Η Σειριακή σύνδεση είναι ακριβώς αυτό που λέει, σειριακή, με την έννοια ότι μεταδίδει δεδομένα με αλληλουχία bit το ένα μετά το άλλο. Στην εντολή Serial.begin() δίνουμε όρισμα το baud rate ή αλλιώς την ταχύτητα μεταφοράς δεδομένων σε bit/second. Οι δυνατές τιμές baud rate είναι 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ή 115200. Αν κάτι εμφανίζεται στην οθόνη μπερδεμένο και ακατάλληπτο, το πιο πιθανό είναι λάθος στο όρισμα του baud rate.

Στην προκειμένη περίπτωση, επιλέχθηκε baud rate 9600bps δηλαδή περίπου 1000 χαρακτήρες/δευτερόλεπτο.

Η Σειριακή σύνδεση Arduino-Server επιτυγχάνεται μέσω USB από όπου παίρνει και τροφοδοσία [UART ενσωματωμένο hardware[1]]. Υπάρχει η δυνατότητα να χρησιμοποιηθούν εναλλακτικά τα digital pins 0 και 1, αλλιώς RX, TX αντίστοιχα. RX: δέχεται, TX: αποστέλει σεριακά δεδομένα.

Όσον αφορά το Arduino Uno ειδικά, διαφέρει από όλες τις προηγούμενες πλακέτες, υπό την έννοια ότι δεν χρησιμοποιεί το FTDI USB-to-serial chip οδηγού. Αντ 'αυτού, ο Atmega16U2 έχει προγραμματιστεί ως USB-to-serial μετατροπέας [2]. Το ATmega16U2 διοχετεύει τη σειριακή επικοινωνία μέσω USB και εμφανίζεται ως μια εικονική θύρα COM στο λογισμικό του υπολογιστή. Το firmware του 16U2 χρησιμοποιεί τα τυπικά προγράμματα οδήγησης COM USB, και δεν απαιτείται κανένας εξωτερικός οδηγός. Ωστόσο, στα Windows, είναι απαραίτητο ένα αρχείο .inf . Ένα Atmega chip μπορεί να λαμβάνει σειριακά δεδομένα ακόμα και όταν εξυπηρετεί άλλες διεργασίες όσο υπάρχει χώρος στον 64 byte serial buffer του.

Επίσης για το Arduino Uno, η βιβλιοθήκη SoftwareSerial Library επιτρέπει σειριακή επικοινωνία σε οποιοδήποτε digital pin επεκτείνοντας τις σειριακές δυνατότητες [3].

Η σύνδεση με Ethernet είναι τελείως διαφορετική. Για αρχή είναι απαραίτητη η αναβάθμιση του Arduino με Ethernet Shield κατάλληλη για την εκάστοτε πλακέτα. Πρέπει να υπάρχει τροφοδοσία για το Arduino και καλώδιο Cat5/Cat6 Ethernet με είσοδο-έξοδο

[1] http://en.wikipedia.org/wiki/UART

[2] http://arduino.cc/en/Main/arduinoBoardUno

[3] http://www.arduino.cc/en/Reference/SoftwareSerial

τύπου RJ45, που συνδέει το Ethernet Shield με το δίκτυο και τον υπολογιστή στον οποίο λειτουργεί ο WAMP Server.

Το συγκεκριμένο Ethernet Shield που χρησιμοποιήθηκε με το Arduino Uno είναι το μοντέλο 2012 Ethernet W5100 Network Shield For Arduino UNO Mega 2560 1280 328 [model HCARDU0034].

Όπως και οποιοδήποτε Arduino κομμάτι, έτσι και το Shield είναι ανοιχτού λογισμικού και hardware. Για να λειτουργεί υπάρχει η αντίστοιχη βιβλιοθήκη Ethernet Library.

Εικόνα 6: Ethernet Shield

Η τροφοδοσία του Shield γίνεται μέσω της πλακέτας Arduino στα 5V. Μπορεί όμως να γίνει και με τη τροφοδοσία που γίνεται πλέον μέσα από το ίδιο το καλώδιο Ethernet ή αλλιώς την ικανότητα Power Over Ethernet (PoE). Αυτό χρήζει ενός επιπλέον εξαρτήματος και κατάλληλου καλωδίου Cat5/Cat6 Ethernet (RJ45). Είναι προφανές πόσο βοηθητική λειτουργία είναι αυτή προκειμένου να ελαχιστοποιήσουμε καλώδια, τροφοδοσία, πιθανές βλάβες στο δίκτυο τροφοδοσίας και κόστος. Παρέχει 9V το οποίο είναι αρκετό για περίπλοκες διατάξεις συνδεδεμένες με το Arduino.

Το ίδιο Shield έχει δυνατότητα σαν εναλλακτική ή και ταυτόχρονη διαδικασία καταγραφής, να περνάει τις μετρήσεις μέσα σε microSD Card πχ. για επιτόπιο buckup σε περίπτωση που δε λειτουργεί ο server. Για αυτή τη λειτουργία χρησιμοποιείται κατάλληλα η βιβλιοθήκη SD library [4].

Το Ethernet Shield πρέπει να αποκτήσει ένα συγκεκριμένο MAC και IP Address, τα οποία είναι απαραίτητα για τη συνάρτηση που κάνει τη σύνδεση, Ethernet.begin() [5].

Το MAC Address είναι ένα καθολικό μοναδικό προσδιοριστικό στοιχείο/μεταβλητή για κάθε συσκευή. Στο documentation αναφέρεται ότι πλέον κάθε Ethernet Shield αναφέρει ποιά MAC Address να χρησιμοποιηθεί, αλλά αν δεν έχει, μια τυχαία μάλλον είναι λειτουργική. Δεν πρέπει όμως για διαφορετικά Arduino να χρησιμοποιείται η ίδια MAC Address.

Όσον αφορά την IP Address, επίσης χρειάζεται η κατάλληλη διεύθυνση. Η διαδικασία διασύνδεσης και αντιστοιχίας με το Server περιγράφεται συγκεκριμένα προκειμένου να διευκολυνθεί η έρευνα και να αποφευχθούν λάθη:

Καταρχήν χρειάζεται να οριστεί συγκεκριμένη IP για την πλευρά του Server. Ορίζεται η IP στη θύρα Ethernet σε μη αυτόματη απόδοση στο λειτουργικό Windows 7 ως εξής: Πίνακας

[4] http://arduino.cc/en/Reference/SD

[5] http://arduino.cc/en/Reference/EthernetBegin

Ελέγχου -> Δίκτυο και Internet -> Συνδέσεις Δικτύου -> Ιδιότητες Τοπικής Σύνδεσης -> Πρωτόκολλο Internet Έκδοση 4 (TCP/IPv4) -> Ιδιότητες.

Στη συνέχεια, από «Αυτόματη απόδοση διεύθυνσης IP» αλλάζει σε «Χρήση της παρακάτω διεύθυνσης IP» όπου εισάγονται ορίσματα όπως βολεύουν με προσοχή σε διενέξεις: Η «Διεύθυνση IP» ορίζεται πχ. στη συγκεκριμένη εργασία 192.168.3.2 , προσέχοντας να μην είναι κοινό το προτελευταίο όρισμα-subnet(.3.) με οποιοδήποτε άλλο υποδίκτυο μπορεί να βρίσκεται ο υπολογιστής που φιλοξενεί το server. Με αυτόν τον τρόπο ορίζεται η IP του Server. Αυτή η διαδικασία συμβαίνει γιατί οι IPs των Arduino πρέπει να βρίσκονται, να ορίζονται, στο συγκεκριμένο subnet (στο παράδειγμα, το 3) και προφανώς όχι η ίδια IP με του Server. Η «Μάσκα Υποδικτύου» ορίζεται συγκεκριμένα, εδώ κοινώς 255.255.255.0 . Τέλος, η «Προεπιλεγμένη Πύλη» μπορεί να μείνει κενή.

Με αυτές τις απαραίτητες ενέργειες η διασύνδεση του υλικού μας είναι έτοιμη σε πρώτο βαθμό, οδηγώντας στο στάδιο του προγραμματισμού των δύο πλευρών: των Arduino και του Υπολογιστή-Server.

ΚΕΦΑΛΑΙΟ 2

ΥΛΟΠΟΙΗΣΗ ΥΠΟΔΟΜΗΣ

2.1

Το σύστημα εξυπηρέτησης

2.1.1 WAMP server

Ο WAMP Server είναι ένα περιβάλλον ανάπτυξης λογισμικού για το διαδίκτυο στο λειτουργικό σύστημα Windows 7. Ονομάζεται έτσι από τα αρχικά των δομικών του στοιχείων που είναι για Windows ο Apache HTTP server, η MySQL και η PHP. Είναι ένα ολοκληρωμένο πακέτο, λειτουργικό αναμεταξύ των στοιχείων που το συναποτελούν και με ένα ακόμα εργαλείο, το PhpMyAdmin, το οποίο επιτρέπει τον πλήρη έλεγχο και εποπτεία των βάσεων που χρειάζονται.

Είναι ένα πολύ βολικό περιβάλλον γιατί εγκαθίσταται ολοκληρωμένο και λειτουργικό, έχει δυνατότητα αλλαγής εκδόσεων των στοιχείων με προσοχή στη μεταξύ τους συμβατότητα, έχει πλήρη πρόσβαση στο server και Logs για όλες τις ενέργειες και προβλήματα που μπορεί να προκύψουν. Ιδιαίτερα φιλικό προς ένα μέσο χρήστη και από εμπειρίας κατατίθεται πως είναι πολύ πιο δύσκολο να στηθούν και να είναι λειτουργικά αναμεταξύ τους τα επιμέρους απαραίτητα στοιχεία, με αποτέλεσμα ο WAMP να αποτελεί πραγματικά μια πολύ ενδιαφέρουσα, σχεδόν απαραίτητη επιλογή.

Για το λειτουργικό Linux, υπάρχει ο αντίστοιχος LAMP.

Για την εργασία χρησιμοποιήθηκε WampServer Version 2.2. [6]

[6] http://www.wampserver.com/en/

2.1.2 Apache server

Ο Apache HTTP server-εξυπηρετητής είναι ένας web server ανοιχτού λογισμικού πολύ διαδεδομένος και με μεγάλες δυνατότητες στο σύνολο των λειτουργικών συστημάτων.

Ως κορμός του WAMP έχει μεγάλη σημασία το στήσιμό του και οι απαραίτητες αλλαγές που έγιναν προκειμένου να λειτουργήσει η παρούσα διπλωματική εργασία. Ο Apache sever που χρησιμοποιήθηκε είναι η έκδοση 2.2.14 .

Η έκδοση αυτή αντικατέστησε εδώ τις νεότερες (2.4.x) προκειμένου να είναι συμβατή με τη συγκεκριμένη έκδοση PHP που μας είναι απαραίτητη καθώς είναι η τελευταία που υποστηρίζει συγκεκριμένη ομάδα εντολών απαραίτητων για την εργασία. Στο κεφάλαιο 2.1.4 αναλύεται εκτενώς η ακριβής κατάσταση.

Όσα χρειάζονται για τη γενική χρήση και τα ενδότερα του Apache μπορούν εύκολα να βρεθούν στην επίσημη ιστοσελίδα του. Εδώ παρατίθονται η πολύ συγκεκριμένη χρήση και οι πολύ συγκεκριμένες αλλαγές στα αρχεία ρυθμίσεων του server.

Ο ίδιος ο εξυπηρετητής βρίσκεται στον υπολογιστή που εγκαθίσταται, δηλαδή για τους προφανείς λόγους στο localhost. Ο «παρών» υπολογιστής μεταφράζεται στην IP address 127.0.0.1 . Η εξοικείωση με τις διαδικτυακές εφαρμογές κλπ είναι απαραίτητη ως βασική γνώση για τη λειτουργία της διπλωματικής εργασίας και για την εφαρμογή της οπουδήποτε. Επί τούτου, αν σε έναν οποιοδήποτε περιηγητή (browser) ενώ λειτουργεί ο Apache εισαχθεί η διεύθυνση «localhost» ή 127.0.0.1 θα εμφανιστεί η αρχική σελίδα με όλες τις προσβάσεις και πληροφορίες του WAMP Server.

Ο Apache έχει ένα κεντρικό αρχείο ρυθμίσεων, το httpd.conf [configuration file] που αποτελείται από οδηγίες [directives] σε απλό αρχείο κειμένου. Μετά την εγκατάσταση ολοκληρωμένου του WAMP το αρχείο αυτό βρίσκεται στο φάκελο, πχ. του C σκληρού δίσκου που έχει μπει, C:\wamp\bin\apache\Apache2.2.14\conf .

Το αρχείο αυτό παραμετροποιεί με πολύ ευκολονόητο τρόπο τον ίδιο τον Apache. Αλλάζουμε το httpd.conf όπως χρειάζεται:

Όπως αναλύονται εκτενώς στο documentation του Apache 2.2, στο module “mod_authz_host” [7] :

Όσον αφορά το κομμάτι των Directories αλλάζουμε ως εξής:

[7] http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

Από **σε**

<Directory />	<Directory />
Options FollowSymLinks	Options FollowSymLinks
AllowOverride **none**	AllowOverride **all**
Order **deny,allow**	Order **allow,deny**
Deny from all	**Allow from all**
</Directory>	</Directory>

- Το “AllowOverride none” σε “all” επιτρέπει από την πλήρη παράβλεψη των αρχείων .htaccess από το server, στην αναγνώριση και εξέτασή τους. Όποτε ο Apache βρίσκει κάποιο αρχείο .htaccess χρειάζεται να γνωρίζει αν και ποιες οδηγίες που ορίζονται μέσα στο αρχείο αυτό μπορούν να παρακάμψουν προηγούμενες οδηγίες λειτουργικότητας (configuration directives). Δεν είναι απαραίτητο να συμβαίνει αυτό, ειδικά όταν υπάρχει άμεση πρόσβαση στο κεντρικό αρχείο configuration του Apache. Η χρήση των .htaccess αρχείων μειώνει αρκετά την απόδοση και ταχύτητα του εξυπηρετητή. Οτιδήποτε μπορεί να χρειάζεται από ένα τέτοιο αρχείο .htaccess μπορεί να εισαχθεί στο Directory block και με καλύτερη απόδοση.

- Η οδηγία “Order” είναι υπεύθυνη για την προεπιλεγμένη κατάσταση πρόσβασης και τη σειρά με την οποία τα “Allow” και “Deny” αξιολογούνται. Όλες οι “Allow” “Deny” οδηγίες θα επεξεργαστούν από το server. Για την ομαλή λειτουργία της εργασίας πρέπει η διαδικασία να είναι “Order allow,deny” [δεν επιτρέπεται κενό ανάμεσα στις λέξεις κλειδιά allow, deny] γιατί πρέπει πρώτα να εξετάζονται όλες οι “allow” οδηγίες και πρέπει τουλάχιστον μία να επιβεβαιώνεται αλλιώς το αίτημα που επεξεργάζεται ο εξυπηρετητής απορρίπτεται. Στη συνέχεια εξετάζονται όλες οι “deny” οδηγίες και αν έστω μία επιβεβαιώνεται το αίτημα απορρίπτεται επίσης καθώς προσπελαύνονται σε ένα Directory section ή .htaccess αρχείο.

- Το “Allow from all” από “Deny from all” επιτρέπει την πρόσβαση σε όλους τους hosts.

Επόμενη αλλαγή στις ρυθμίσεις αφορά τις δυνατότητες πρόσβασης στον server:

Από **σε**

Order **Deny,Allow**	Order **Allow,Deny**
Deny from all	**Allow from all**
Allow from 127.0.0.1	Allow from 127.0.0.1

Τέλος, για τα .htaccess και .htpasswd, αλλάζουν οι ρυθμίσεις προκειμένου να είναι προσβάσιμα από web clients:

Από	**σε**
<FilesMatch "^\.ht">	<FilesMatch "^\.ht">
Order allow,deny	Order allow,deny
Deny from all	**Allow** from all
Satisfy All	Satisfy All
</FilesMatch>	</FilesMatch>

Όπως είναι σαφές, αυτές οι ρυθμίσεις, προαιρετικές σε έναν βαθμό, απαραίτητες για την πρόληψη διενέξεων και λαθών στη ρύθμιση του εξυπηρετητή, είναι πολύ επικίνδυνες για θέματα ασφαλείας και πρέπει γενικά στο διαδίκτυο να αποφεύγονται. Για τη συγκεκριμένη εργασία δεν έχει σημασία το ζήτημα αυτό.

2.1.3 MySQL Database - PhpMyAdmin

Η MySQL αποτελεί το πιο διαδεδομένο σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) ανοιχτού λογισμικού. Είναι το εργαλείο που επιτρέπει τη βέλτιστη οργάνωση και καταγραφή μετρήσεων, δεδομένων κλπ, την εύκολη διαχείρισή τους και ένα τεράστιο πλήθος δυνατοτήτων επεξεργασίας και διαμόρφωσής τους.

Στον WAMP Server η MySQL είναι ενσωματωμένη και διαχειρίσιμη από το συμπληρωματικό της πρόγραμμα PhpMyAdmin. Οι εκδόσεις που χρησιμοποιήθηκαν είναι MySQL 5.5.24 και PhpMyAdmin 3.5.1 .

Το PhpMyAdmin είναι ένα εργαλείο ελεύθερου και ανοιχτού λογισμικού γραμμένο σε PHP προκειμένου να διαχειρίζεται τη MySQL, με τη χρήση κάποιου web browser. Μέσα από αυτή την προγραμματιστική πλατφόρμα φτιάχνονται νέες βάσεις δεδομένων, τροποποιούνται κατά το δοκούν και εκτελούνται MySQL κώδικες.

Όπως και ο Apache έτσι και το PhpMyAdmin έχει αρχείο ρυθμίσεων που χρειάζεται κάποιες αλλαγές προκειμένου να λειτουργήσει με τις συγκεκριμένες εκδόσεις των υπόλοιπων συστατικών στοιχείων του WAMP ιδιαίτερα με την PHP και τον Apache. Το αρχείο ρυθμίσεων βρίσκεται στον σκληρό εγκατάστασης του WAMP, πχ. C, C:\wamp\alias με ονομασία phpmyadmin.conf .

Στο phpmyadmin.conf η αλλαγή προκειμένου να είναι εφικτή η εξωτερική πρόσβαση στο PhpMyAdmin, είναι ως εξής:

Από	**σε**
Require local	**Order Deny,Allow**
	Deny from all
	Allow from all

2.1.4 PHP

Η PHP είναι μια γλώσσα προγραμματισμού ειδικά σχεδιασμένη για server και ανάπτυξη web εφαρμογών, ιστοσελίδων κλπ. Επίσης open γλώσσα πολύ διαδεδομένη και χρηστική σε οποιοδήποτε λειτουργικό σύστημα.

Στην παρούσα εργασία η PHP χρησιμοποιείται σαν μεταβατικό στοιχείο στην Ethernet διασύνδεση προκειμένου να συνδέεται με το server και τη MySQL περνώντας προς μεταγλώτισση την εντολή MySQL με τις μετρήσεις και τα λοιπά χαρακτηριστικά στη Βάση Δεδομένων.

Η PHP ως πολύ διαδεδομένη στο διαδίκτυο έχει και θα έχει ζητήματα ασφάλειας τα οποία ανακαλύπτονται σε ζωντανό χρόνο και εφαρμογές καλοπροαίρετα ή κακοπροαίρετα. Η έκδοση που χρησιμοποιήθηκε δεν είναι η τελευταία για ακριβώς αυτό το λόγο. Μια συγκεκριμένη και πολύ χρήσιμη σειρά μεταβλητών αποτελούν αποδεδειγμένα κίνδυνο ασφάλειας με αποτέλεσμα τη σταδιακή απονέκρωσή τους από την PHP και σήμερα την απενεργοποίησή τους τελείως. Ως εκ τούτου προηγούμενες εκδόσεις τις είχαν ενεργές και είναι προεπιλεγμένα απενεργοποιημένες σήμερα προκειμένου να γίνουν από όσους θέλουν συνειδητές επιλογές. Στις μεταγενέστερες έχουν αφαιρεθεί τελείως [8]. Εδώ χρησιμοποιήθηκε η τελευταία έκδοση PHP 5.3.1 η οποία υποστηρίζει -ανενεργή- την οδηγία “Register_Globals”, αλλιώς PHP registers GET, POST, Cookie, Environment and Built-in μεταβλητές (G, P, C, E & S αντίστοιχα, συχνά προσδιορισμένες ως EGPCS ή GPC).

Η αλλαγή προκειμένου να υποστηρίζονται τα “Register_Globals” γίνεται στο αρχείο ρυθμίσεων της PHP, php.ini στη θέση (πχ. στο σκληρό δίσκο C) C:\wamp\bin\apache\Apache2.2.14\bin όπου από “Off” αλλάζει σε “On”:

[το «;» σημαίνει σχόλιο, αφήνονται εδώ προκειμένου να είναι ξεκάθαρη η θέση και η τοποθέτηση των δημιουργών στο ίδιο το αρχείο ρυθμίσεων της PHP]

```
; Whether or not to register the EGPCS variables as global variables.  You may
; want to turn this off if you don't want to clutter your scripts' global scope
; with user data.
; You should do your best to write your scripts so that they do not require
; register_globals to be on;  Using form variables as globals can easily lead
; to possible security problems, if the code is not very well thought of.
; http://php.net/register-globals
register_globals = On
```

[8] http://php.net/manual/en/security.globals.php

2.2

Η Βάση Δεδομένων

Η Βάση Δεδομένων σχεδιάστηκε προκειμένου να εξυπηρετεί τις ανάγκες της εργασίας και να διακρίνει σαφώς μεταξύ τους τα διαφορετικά Arduino, αισθητήρες κλπ. Η Βάση Δεδομένων προσαρμόζεται κατά τις ανάγκες που εμφανίζονται κάθε φορά και για κάθε διαφορετικό πρόβλημα, οργανώνοντας με σαφήνεια κάθε φορά τα εισαγόμενα δεδομένα.

Η Βάση Δεδομένων που παρατίθεται είναι αποτέλεσμα των δεδομένων λοιπόν που προκύπτουν από τα συγκεκριμένα ζητήματα και με τα συγκεκριμένα μέσα.

Η Βάση Δεδομένων ονομάζεται arduinothesisdb και έχει ένα πίνακα μετρήσεων τον “measurements”.

Κάθε μέτρηση έχει μοναδική αρίθμηση με αυτόματη διαδικασία “auto increment”. Το χαρακτηριστικό που προσδιορίζεται μοναδικά για κάθε μέτρηση αποτελεί και την ιδιαίτερή του ταυτότητα ως μοναδικό και ονομάζεται μη τυχαίως “id” αποτελώντας και το πρωταρχικό κλειδί του πίνακα (primary key) τύπου integer(ακέραιος int).

Κάθε Arduino φροντίζεται να έχει από πλευράς πρωταρχικού σχεδιασμού και προγραμματισμού μοναδικό επίσης όνομα το οποίο καταγράφεται. Το χαρακτηριστικό του πίνακα που το αποθηκεύει είναι το “arduinoID” το οποίο επιλέχθηκε να είναι τύπου varchar() και μεγέθους μέχρι 30 χαρακτήρων (το νούμερο αποτελεί αυθαίρετη επιλογή, μπορεί να είναι ό,τι χρειάζεται).

Όμοια, varchar είναι και τα χαρακτηριστικά “pinNumber” και “whatever”. Τα μεγέθη αντίστοιχα είναι 15 και 20 χαρακτήρες.

Το χαρακτηριστικό “pinNumber” παίρνει τιμές τον αριθμό pin που μας ενδιαφέρει να καταγράψουμε, πχ. μετράει από αισθητήρα. Τα Arduino έχουν αρκετά Pins, αλλά η καταγραφή που χρειάζεται στην παρούσα εργασία είναι τα Analog Pins και τα Digital Pins. Τα Analog Pins είναι σύνολο 6 και έχουν αρίθμηση 0-5. Τα Digital Pins είναι σύνολο 12 και αριθμούνται 2-13. Ανάλογα από πού καταγράφονται δεδομένα, αντίστοιχα στο χαρακτηριστικό “pinNumber” καταγράφεται τι είδους και ποιο είναι το pin. Παραδείγματα: “Analog_Pin_2”, “Digital_Pin_9”. Εξαίρεση αποτελεί η Ethernet σύνδεση που εξαιρεί όπως περιγράφεται παρακάτω τα digital pins 10, 11, 12 και 13.

Το χαρακτηριστικό “whatever” είναι μια ζητούμενη σειρά χαρακτήρων που ορίζεται εξαρχής για κάθε Arduino.

Τέλος, καταγράφεται η ίδια η μέτρηση στη στήλη του πίνακα “reading” η οποία όπως ζητήθηκε είναι float αριθμός. Τα δεκαδικά που μπορεί να έχει είναι προσδιορισμένα στον κώδικα, εδώ η ακρίβεια έχει οριστεί να είναι 2 δεκαδικών ψηφίων.

Ο πίνακας με όλες τις επιθυμητές στήλες σε μια Βάση Δεδομένων αυτούσιος μπορεί να παραχθεί από τον κώδικα SQL παρακάτω:

```
CREATE TABLE IF NOT EXISTS `measurements` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `arduinoID` varchar(30) NOT NULL,
  `pinNumber` varchar(15) NOT NULL,
  `reading` float NOT NULL,
  `whatever` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

- Το χαρακτηριστικό που παρουσιάζεται ως ENGINE InnoDB αποτελεί την προεπιλεγμένη μηχανή αποθήκευσης η οποία είναι πολύ αποτελεσματική εμπεριέχοντας όλους τους απλούς και σύνθετους κανόνες σχεδίασης. [9]
- Όλα τα στοιχεία για την πρόβλεψη λάθους έχουν οριστεί να είναι by default “NOT NULL”. Αυτό σημαίνει ότι κάθε καταχώρηση οφείλει να έχει τιμή για κάθε πεδίο-στήλη της.

[9] http://dev.mysql.com/doc/refman/5.5/en/innodb-storage-engine.html

2.3

PHP και διασύνδεση με τη Βάση Δεδομένων

Η PHP που χρησιμοποιήθηκε καλύπτει ακριβώς τη διασύνδεση των Arduino με τη Βάση Δεδομένων στο server. Καλείται μέσα από τον κώδικα του Arduino για την Ethernet σύνδεση, συνδέεται με τη Βάση Δεδομένων που χρησιμοποιείται για την εφαρμογή και παράγει την sql πρόταση-query με όλες τις τιμές που χρειάζονται.

Το αρχείο ονομάζεται arduinoThesis.php και πρέπει να βρίσκεται στη θέση (πχ. πάλι σκληρός δίσκος εγκατάστασης WAMP, C) C:\wamp\www .

Ο κώδικας είναι ως εξής:

```
<?php
var_dump($_GET);
$readings = $_GET['readingcpp'];
$pinNum = $_GET['pinNumbercpp'];
$ardID = $_GET['arduinoIDcpp'];
$what = $_GET['whatevercpp'];

//Connect to database
$con = mysql_connect("localhost", "root", "");
mysql_select_db("arduinothesisdb", $con);
mysql_query("INSERT INTO measurements(arduinoID, pinNumber, reading, whatever)
VALUES('$ardID', '$pinNum', '$readings', '$what')");
mysql_close($con);
?>
```

- Η εντολή $_GET είναι της οικογένειας των Register Globals. Η χρήση της είναι μια σειρά από μεταβλητές να περνάνε στον τρέχων κώδικα από τις παραμέτρους του URL[10].

  Η $_GET έχει μια ιδιαιτερότητα, δεν αναγνωρίζει το κενό-space στις προτάσεις λόγω reserved characters, οι οποίοι μετά από τη διαδικασία URL encoding αναγνωρίζονται διαφορετικά. Το space αντιστοιχίζεται με %20 [11].

- Η εντολή mysql_connect ανοίγει μια σύνδεση σε κάποιον MySQL server. Εδώ αφορά τη σύνδεση με τον Apache server και τη Βάση Δεδομένων. Οι παράμετροι που

[10] http://php.net/manual/en/reserved.variables.get.php

[11] http://en.wikipedia.org/wiki/Percent-encoding#Percent-encoding_reserved_characters

εισάγονται είναι η θέση του Server, εδώ “localhost”, το όνομα χρήστη Username, εδώ “root”, και ο κωδικός Password, εδώ κενός οπότε “” [12].

- Η εντολή mysql_select_db προσδιορίζει ποια θα είναι η ενεργή Βάση Δεδομένων στο server σχετιζόμενη με τη σύνδεση που έχει ανοίξει με την εντολή mysql_connect. Τα ορίσματά της είναι το όνομα της Βάση Δεδομένων (database_name) και η MySQL σύνδεση (link_identifier). Εδώ το όνομα της Βάση Δεδομένων που χρησιμοποιείται είναι arduinothesisdb και η σύνδεση στο localhost, root [13].

- Η εντολή mysql_query στέλνει μια πρόταση/ερώτηση (query) SQL στην ενεργή Βάση Δεδομένων στη συγκεκριμένη σύνδεση με το server. Δεν υποστηρίζει πολλαπλά queries και το ορίσμά της είναι το ίδιο το query.

  Εδώ το query συντάσσεται απλά [*INSERT INTO measurements(arduinoID, pinNumber, reading, whatever) VALUES('$ardID', '$pinNum', '$readings', '$what')*] και η ερμηνεία του είναι ως εξής: Εισαγωγή στον πίνακα “measurements” στις στήλες-χαρακτηριστικά “arduinoID”, “pinNumber”, “reading”, “whatever” αντίστοιχα τα δεδομένα, τις τιμές που στέλνονται '$ardID', '$pinNum', '$readings', '$what' [14].

- Η εντολή τέλος mysql_close κλείνει τη σύνδεση με το server που έχει ανοίξει και αποτελεί όρισμα της συνάρτησης [15].

[12] http://php.net/manual/en/function.mysql-connect.php
[13] http://php.net/manual/en/function.mysql-select-db.php
[14] http://php.net/manual/en/function.mysql-query.php
[15] http://php.net/manual/en/function.mysql-close.php

2.4

Arduino – Processing περιβάλλοντα

Στο παρόν Κεφάλαιο παραθέτονται οι διαδικασίες, οι κώδικες, οι βιβλιοθήκες που είναι απαραίτητες για τη λειτουργία ολόκληρης της εργασίας. Αποτελεί το πιο κεντρικό της κομμάτι, τον κορμό, και έχει τα σκέλη-υποκεφάλαια όπως αναλύονται.

Το γενικό σκεπτικό είναι η όσο περισσότερο ομογενοποίηση των διαδιακασιών σύνδεσης Serial - Ethernet μεταξύ Arduino και server.

Για το σκοπό αυτό, σε κάθε Arduino φορτώνεται το ίδιο πρόγραμμα με μια διαφοροποίηση ανάλογα με τον τρόπο που θα συνδεθεί.

Όπως φαίνεται από τον κώδικα παρακάτω προκειμένου να ενιαιοποιηθούν οι διαδικασίες σύνδεσης προγραμματιστικά, το μόνο που αλλάζει κατά περίπτωση είναι αν χρειάζεται MAC address. Αν έχει MAC μόνο μηδενικά, τότε επιλέγεται η σειριακή σύνδεση και καλεί από τη βιβλιοθήκη Od_connect.h τη συνάρτηση που είναι υπεύθυνη για την ενεργοποίηση της σειριακής καταγραφής, ενώ στην άλλη περίπτωση τη διαδικασία που εκκινεί την Ethernet σύνδεση με την με τις Ethernet.begin() και con.connect_ethernet(). Οι συναρτήσεις αυτές περιγράφονται κατά περίπτωση στην παράθεση και επεξήγηση της βιβλιοθήκης.

Στην αρχή του sketch(το αρχείο με τον κώδικα για το Arduino) ενσωματώνονται τρεις βιβλιοθήκες, η Od_connect που δημιουργήθηκε για τις ανάγκες της παρούσας εργασίας, η SPI και η Ethernet. Οι τελευταίες δύο παρουσιάζονται για διευκόλυνση στο κεφάλαιο 2.4.1 όπου πλέον εξοικειώνεται ο αναγνώστης με τις βιβλιοθήκες.

Τα προγράμματα-περιβάλλοντα προγραμματισμού που χρησιμοποιήθηκαν είναι τα εξής: Processing 2.0.3 και Arduino-1.0.5 σε WINDOWS 7. Οι εκδόσεις 64bit έχουν θέματα συμβατότητας με τη βιβλιοθήκη BezierSQLib η οποία αναλύεται αργότερα, οπότε προτιμήθηκαν 32bit. Η παραδοχή 32 bit αποτελεί και την πιο γενική περίπτωση μιας και συστήματα 64bit δεν συγκρούονται με 32bit εφαρμογές κλπ, ενώ το αντίστροφο μπορεί να αποτελέσει πρόβλημα.

Η βιβλιοθήκη καλείται και φτιάχνεται ένα καινούργιο στιγμιότυπό της, ένα instance που ορίζεται ως *Od_connect con(arduinoName, whatever, ip, mac, serverName)* με ορίσματα όλα όσα μπορεί να χρειαστούν και χρειάζονται για την βιβλιοθήκη και το πέρασμα των μετρήσεων στη Βάση Δεδομένων. Επίσης ορίζονται το μοναδικό όνομα του κάθε Arduino και το String που ζητήθηκε “whatever”, τα οποία χρειάζεται από εδώ να καταγραφούν μόνο για την Ethernet σύνδεση.

Κάθε Arduino πρόγραμμα έχει δύο σκέλη, το setup και το loop.

Η συνάρτηση setup() καλείται μία μόνο φορά στο πρόγραμμα και είναι το κατάλληλο πεδίο που μπορούν να οριστούν αρχικές ρυθμίσεις όπως τη χρήση των pins και την αρχικοποίηση βιβλιοθηκών. Εδώ αρχικοποιείται πάντα η Serial διαδικασία και αν η σύνδεση είναι Ethernet, η Ethernet διαδικασία.

Η συνάρτηση loop() καλείται συνεχώς και αποτελεί τον πυρήνα του προγράμματος. Εδώ ανάλογα με τη σύνδεση καλούνται οι αντίστοιχες συναρτήσεις από τη βιβλιοθήκη Od_connect.

Τόσο η συνάρτηση setup() όσο η loop() είναι απαραίτητο να υφίστανται ακόμα και αν δεν έχουν περιεχόμενο.

Το sketch Arduino_mult_sensors_odconnect_THIS.ino είναι απαραίτητο για όλα τα Arduino και παρουσιάζεται σε ψευδοκώδικα ως εξής:

Define mac, ip, Servername, arduinoName, whatever;
Od_connect con(arduinoName, whatever, ip, mac, serverName);
if MAC[0] == 0 { con.connect_serial();}
else if MAC[0] != 0 {con.connect_ethernet();}

Ολόκληρο το sketch παραπέμπεται στο Παράρτημα 2.1.

2.4.1
Βιβλιοθήκη OD_connect για Serial και Ethernet διασύνδεση

Για τη βέλτιστη δόμηση της εργασίας, την ομογενοποίηση κατά το δυνατόν των μεθόδων διασύνδεσης υλικού και την καλύτερη οργάνωση του κώδικα για την παρούσα εργασία αλλά και μελλοντικές επεκτάσεις, πχ. σε ασύρματες συνδέσεις, δημιουργήθηκε μια βιβλιοθήκη για το Arduino που εξυπηρετεί όλες τις βασικές λειτουργίες. Βιβλιοθήκη είναι επί της ουσίας ένα σύνολο από εντολές σχεδιασμένο έτσι ώστε να εκτελούν μια συγκεκριμένη λειτουργία.

Οι βιβλιοθήκες χρησιμοποιούνται προκειμένου να απλοποιήσουν τη χρήση και οργάνωση του κώδικα, να ενισχύσουν την ευκολία ανάγνωσης του κώδικα και να αποκεντρωθεί η λογική, δηλαδή να καλείται στο πρόγραμμα library.action() αντί για ολόκληρο τον κώδικα που υλοποιείται μέσα τη βιβλιοθήκη. Οι βιβλιοθήκες είναι προσβάσιμες από όλα τα προγράμματα αφού εγκατασταθούν.

Όσον αφορά το Arduino board όταν καλείται κάποια βιβλιοθήκη φορτώνεται (γίνεται upload) μαζί με το βασικό κώδικα (sketch) οπότε αυξάνεται ο όγκος πληροφορίας που πρέπει να αποθηκευτεί στη μνήμη του μικροεπεξεργαστή στην πλακέτα. Σε κάποια ογκώδη εφαρμογή πρέπει να διαχειριστεί κανείς τους πόρους αναλόγως.

Κάθε βιβλιοθήκη απαρτίζεται πάντα από τουλάχιστον δύο αρχεία, ένα header file (με κατάληξη .h) και ένα source file (με κατάληξη .cpp). Το header περιέχει όλους τους ορισμούς για τη βιβλιοθήκη, δηλαδή μια καταγραφή όλων όσων περιέχονται σε αυτήν, ενώ το source τον ίδιο τον κώδικα, την ίδια τη λειτουργία, δηλαδή την υπολοίηση των δηλωμένων συναρτήσεων και μεταβλητών από το header αρχείο.

Σημείωση: Μπορεί να υπάρχουν πάνω από ένα .h και .cpp αρχεία, αλλά πρέπει πάντα να υπάρχει ένα .h που έχει το ίδιο όνομα με τη βιβλιοθήκη και αυτό είναι που γίνεται *#include* στην αρχή ενός sketch.

Η βιβλιοθήκη για την εργασία ονομάζεται OD_connect και τα αντίστοιχα αρχεία είναι τα OD_connect.h και OD_connect.cpp. Τα αρχεία αυτά ως φάκελος βιβλιοθήκης πρέπει να τοποθετηθούν στο directory που βρίσκεται το πρόγραμμα-διεπαφή του Arduino, στον υποφάκελο “libraries” σε φάκελο με ονομασία OD_connect. Εκεί βρίσκονται και οι υπόλοιπες πρόσθετες βιβλιοθήκες για το Arduino.

Τόσο στο OD_connect.h όσο και στο OD_connect.cpp πρέπει να δηλώνονται οι λοιπές βιβλιοθήκες που χρησιμοποιούνται με την ακριβή τους θέση. Για παράδειγμα η βιβλιοθήκη Ethernet.h συμπεριλαμβάνεται και καλείται και στα δύο αρχεία με ολόκληρη τη διεύθυνσή της με την εξής πρόταση: *#include "C:\...\arduino-1.0.5\libraries\Ethernet\Ethernet.h".*

Στο source αρχείο πρέπει επίσης να καλείται και το header αρχείο με include. Δηλαδή, *#include "Od_connect.h"*.

Στο .h αρχείο ορίζεται η κλάση Od_connect η οποία έχει ένα public και ένα private μέρος. Το public κομμάτι είναι δημόσιο και προσβάσιμο από όποιον χρησιμοποιεί την ίδια την κλαση, για οποιαδήποτε λειτουργία, ενώ το private είναι προσβάσιμο μόνο μέσα από την ίδια την κλάση.

Ο γενικός κανόνας είναι πως όσα περισσότερα μπορούν να οριστούν στο private κομμάτι τόσο το καλύτερο. Για τον προγραμματισμό Arduino αυτό συνήθως –όπως και εδώ- σημαίνει ότι όλες οι μεταβλητές δηλώνονται private και όλες οι συναρτήσεις public.

Στην προκείμενη πρίπτωση στο public μέρος ορίζεται ο (πάντα) ομόνυμος constructor (ο constructor είναι υπεύθυνος για τη δημιουργία ενός στιγμιοτύπου –instance- της κλάσης) Od_connect με ορίσματα *String arduinoName, String whatever, uint8_t ip[], uint8_t mac[], uint8_t serverName[].* Όπως είναι ήδη διατυπωμένο δημόσιες ορίζονται οι συναρτήσεις connect_serial() και connect_ethernet().

Στο private μέρος ορίζονται οι μεταβλητές *String _arduinoName, String _whatever, uint8_t _mac[6], uint8_t _ip[4], uint8_t _serverName[4]*.

Η δομή του .h αρχείου έχει συγκεκριμένη αρχή και τέλος:

```
// These 2 lines prevent the code from being included into the program binary multiple times
#ifndef Od_connect_h
#define Od_connect_h

…

#endif  //ends the #ifndef preprocessor directive
```

Όπως επεξηγείται και από τα σχόλια, οι πρώτες δύο γραμμές χρειάζονται για την αποτροπή προβλημάτων σε περίπτωση που καλεστεί πάνω από μία φορά η βιβλιοθήκη και η τελευταία κλείνει την οδηγία αυτή.

Σημείωση: Για το Arduino API (*application programming interface*) είναι απαραίτητο να χρησιμοποιείται η βιβλιοθήκη Arduino.h που περιέχει τις απαραίτητες δηλώσεις. Αυτή αντικατέστησε από την έκδοση Arduino 1.0 και ύστερα την προηγούμενη έκδοση με όνομα WProgram.h . Για να προβλεφθούν αν χρειάζεται ασυμβατότητες μεταξύ των πιθανών εκδοχών, η αρχή του .h αρχείου μπορεί να γίνει ως εξής:

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
```

Ολόκληρος ο κώδικας του Od_connect.h παραπέμπεται στο Παράρτημα 2.2.

2.4.2
Ethernet επικοινωνία

Το αρχείο της βιβλιοθήκης Od_connect, Od_connect.cpp, περιέχει τις υλοποιήσεις ολόκληρης της Ethernet σύνδεσης και την υλοποίηση από πλευράς Arduino της Serial.

Η ανάλυση της κάθε διαδικασίας ξεκινώντας από την Ethernet θα ολοκληρώσει και την παρουσίαση της βιβλιοθήκης από το προηγούμενο υποκεφάλαιο.

Στα δύο αρχεία της βιβλιοθήκης Od_connect ενσωματώνεται η Ethernet library. Μαζί της, περιλαμβάνεται και αναφέρεται ήδη από το sketch του Arduino, η βιβλιοθήκη SPI.

- Ethernet library:

  Η βιβλιοθήκη αυτή, μαζί με ένα Ethernet Shield επιτρέπει στο Arduino board να συνδέεται με το διαδίκτυο. Έχει μία ακόμη βασική λειτουργία, η οποία για τη συγκεκριμένη εφαρμογή στη διπλωματική εργασία είναι η ενδιαφέρουσα.

  Η βιβλιοθήκη λειτουργεί είτε ως server-εξυπηρετητής δεχόμενος εισερχόμενες διασυνδέσεις είτε ως client-πελάτης δημιουργώντας εξερχόμενες διασυνδέσεις. Η βιβλιοθήκη μπορεί να υποστηρίζει μέχρι τέσσερις διασυνδέσεις είτε εξερχόμενες είτε εισερχόμενες είτε συνδυασμό των παραπάνω.

  Στην παρούσα εργασία, στην περίπτωση Ethernet, όπου και καλείται η βιβλιοθήκη χρησιμοποιείται η client δυνατότητα, αφού αυτό χρειάζεται. Το Arduino Shield στέλνει μια συγκεκριμένη πρόταση στο server που το εξυπηρετεί. Η κάθε πρόταση έχει όλα τα στοιχεία που χρειάζεται να εισαχθούν στη Βάση Δεδομένων στο server.

Για το σκοπό αυτό χρησιμοποιείται η κλάση EthernetClient() η οποία δημιουργεί έναν client ο οποίος μπορεί να συνδεθεί σε συγκεκριμένο IP και port. Εδώ συνδέεται με το IP του server και port-θύρα την 80.

Αφού λοιπόν δημιουργηθεί ένα καινούργιο στιγμιότυπο της EthernetClient και δημιουργηθεί ο client, συνδέεται. Ο κώδικας είναι απλός:

```
EthernetClient _client;
_client.connect( _serverName, 80 )
```

Η συνάρτηση client.connect() επιστρέφει true αν η σύνδεση είναι επιτυχής αλλιώς false.

Οι θύρες προσδιορίζουν με μοναδικό τρόπο διαφορετικές εφαρμογές ή διαδικασίες που τρέχουν σε έναν υπολογιστή και έτσι τους δίνει τη δυνατότητα να μοιραστούν μια ενιαία φυσική σύνδεση με ένα δίκτυο μεταγωγής πακέτων, όπως το Διαδίκτυο.

Η θύρα 80 είναι προεπιλογή για το HTTP (Hypertext Transfer Protocol).

Ένας πελάτης HTTP ξεκινά μια αίτηση μέσω του πρωτοκόλλου Transmission Control Protocol (TCP) σε μια συγκεκριμένη θύρα στο διακομιστή (συνήθως θύρα 80). Ένα HTTP server εκτελεί ακρόαση στη θύρα αυτή και περιμένει για το αίτημα ενός πελάτη.

Το HTTP ορίζει μεθόδους για να υποδεικνύουν την επιθυμητή δράση που πρέπει να εκτελεστεί. Στην προκειμένη εργασία χρησιμοποιείται για το HTTP 1.1 η μέθοδος GET. Αιτήματα που χρησιμοποιούν τη μέθοδο GET μπορούν μόνο να ανακτούν δεδομένα. Η κατάλληλη για τις ανάγκες τις εργασίας GET πρόταση-αίτημα φτιάχνεται στη συνάρτηση connect_ethernet().

Αυτή η πρόταση «εκτυπώνεται», γίνεται print στο server στον οποίο είναι συνδεδεμένος ο client με την συνάρτηση client.print() και client.println().

Για παράδειγμα από το Apache access log βλέπουμε την πρόταση
192.168.3.3 - - [30/Oct/2013:22:21:42 +0200] "GET /arduinoThesis.php?readingcpp=20.51&arduinoIDcpp=Ethernet_1&pinNumbercpp=Analog_Pin_0&whatevercpp=Whatever HTTP/1.1" 200 191 .

Αυτό το string στέλνεται στην “ιστοσελίδα” *arduinoThesis.php*, όπου ξεχωρίζονται από το php πρόγραμμα τα δεδομένα προς εισαγωγή στη βάση δεδομένων.

Συνοψίζοντας, η διαδικασία Ethernet ξεκινάει από τον αισθητήρα που συνδέεται με το Ethernet Shield. Το shield συνδέεται με το Arduino (SPI βιβλιοθήκη όπως αναλύεται από κάτω) και με τον υπολογιστή-server. Στην πλευρά του υπολογιστή, το αίτημα που έχει αποσταλεί αποκωδικοποιείται και τα σωστά δεδομένα τέλος εισάγονται στη Βάση Δεδομένων.

Στη διαδικασία Ethernet αλλά σχετιζόμενη και με το επόμενο κεφάλαιο, παρουσιάζεται και η βιβλιοθήκη SPI.

- <u>SPI library:</u>

  Serial Peripheral Interface

  Αποτελεί ένα πρωτόκολλο για σύγχρονη επικοινωνία με σειριακά δεδομένα μεταξύ μικροελεγκτή και περιφερειακών συσκευών, αλλά και μεταξύ μικροελεγκτών. Το πρωτόκολλο SPI δίνει τη δυνατότητα της ταυτόχρονης αποστολής και λήψης, αντίθετα με άλλα όπως το I2C που δεν το επιτρέπουν αυτό. Παρόλο που δεν χρησιμοποιείται άμεσα η βιβλιοθήκη, καλείται μέσα στην Ethernet.

Αυτό συμβαίνει γιατί εξυπηρετείται η επικοινωνία Arduino και Ethernet Shield η οποία γίνεται μέσω SPI (μέσα από το ICSP header [16]). Το Ethernet Shield βασίζεται στο chip Wiznet W5100 με 16K Buffer [17]. Για αυτό το λόγο, η ανάγκη ενσωμάτωσης της SPI library βρίσκεται στα αρχεία της Ethernet library -> utilities -> W5100.h και .cpp .

Συγκεκριμένα για το Arduino Uno χρησιμοποιούνται για το δίαυλο SPI bus το pin 10 ως SS και τα digital pins 11, 12 και 13. Ως εκ τούτου στο Arduino δεν πρέπει να χρησιμοποιούνται αυτά τα pins για οτιδήποτε άλλο.

Η βιβλιοθήκη SPI πρέπει να περιλαμβάνεται ως *#include <SPI.h>* πριν από την Ethernet.

Ολόκληρος ο κώδικας του Od_connect.cpp παραπέμπεται στο Παράρτημα 2.3.

[16] http://en.wikipedia.org/wiki/In_Circuit_Serial_Programming_%28ICSP%29

[17] www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=5&cate2=7&cate3=26&pid=1011

2.4.3
Σειριακή επικοινωνία

Η Σειριακή διαδικασία εισάγει ένα ακόμα εργαλείο προγραμματισμού από πλευράς υπολογιστή, την πλατφόρμα και γλώσσα Processing. Για να καταγράψουμε με σειριακό τρόπο δεδομένα από το Arduino στη Βάση Δεδομένων η συνοπτική παρουσίαση είναι ως εξής:

- Φορτώνεται το Arduino sketch κατάλληλα αλλαγμένο για serial σύνδεση, δηλαδή για το MAC address όλα τα στοιχεία 0. Δεν πρέπει να είναι ανοιγμένη η σειριακή οθόνη γιατί δεν μπορεί να λειτουργεί το πρόγραμμα Processing στην ίδια COM θύρα.
- Καλείται από το Arduino sketch η βιβλιοθήκη Od_connect, συγκεκριμένα η συνάρτηση connect_serial().
- Όσο «τρέχει» το Processing script τόσο θα καταγράφονται στη Βάση Δεδομένων μετρήσεις.

Η λειτουργία του Serial μέρους βασίζεται στην αποστολή και αποκωδικοποίηση bytes.

Από τη βιβλιοθήκη Od_connect για τη σειριακή σύνδεση στη συνάρτηση connect_serial(), αποστέλλεται πρώτος ο χαρακτήρας H (capital H, ASCII 72), αποτελώντας τον Header της αλληλουχίας bytes που θα αποσταλούν. Το ίδιο δεσμεύει ένα byte.

Μετά το Header αποστέλονται οι μετρήσεις από τα Digital Pins (2-13), τα οποία βρίσκονται σε ένα integer. Κάθε integer έχει μέγεθος 2 bytes. Για κάθε Digital Pin χρειάζομαστε 1 bit. Οπότε για όλα τα Digital Pins αρκούν 2 bytes (1 byte = 8 bits) τα οποία καταχωρούνται σε low – high order, ή αλλιώς από δεξιά προς τα αριστερά [high order bit θεωρείται το msb τελευταίο αριστερά].

Παράδειγμα: Έστω ότι τα 2-6 Digital Pins είναι high (1), τα 7,8,9 low (0), τα 10-11 high και 12,13 low. Αυτό από τον κώδικα

```
for(int i=2; i <= 13; i++)
{
        bitWrite(values, bit, digitalRead(i));  // set the bit to 0 or 1 depending
                                                // on value of the given pin
        bit = bit + 1;        // increment to the next bit
}
```

[Ορισμός: bitWrite(byteValue, bitIndex, bitValue)]

μεταφράζεται σε μια αλληλουχία bits καταχωρημένη σαν integer των 2 bytes. Δηλαδή αποστέλεται ως bits 0000001100011111 σε 2 bytes. Η συνάρτηση bitWrite() γράφει κάθε Digital Pin, διαβάζοντας με τη digitalRead(), την τιμή του pin (high/low - 1/0).

Στη συνέχεια χρειάζεται ένας integer (που στη συνέχεια μετατρέπεται σε float, δες Σημείωση) για κάθε Analog Pin. Άρα χρειάζονται 6 integers * 2 bytes = 12 bytes ακόμα για τις αναλογικές μετρήσεις.

Συνολικά αποστέλονται 15 bytes. 1 για την επικεφαλίδα Η, 2 για όλα τα Digital Pins, 2 για κάθε Analog Pin. Χρησιμοποιώντας κάθε φορά τις εντολές

Serial.write(lowByte(values)); // send the low byte
Serial.write(highByte(values)); // send the high byte

αποστέλλεται όπως πρέπει κάθε ζευγάρι bytes και αποκωδικοποιείται αντίστοιχα στην πλευρά του υπολογιστή στο Processing πρόγραμμα.

Πίνακας 2: Η συνολική ακολουθία bytes που αναγνωρίζει ο υπολογιστής

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14
Header H	Digital Pins 2-13		Analog Pin 0		Analog Pin 1		Analog Pin 2		Analog Pin 3		Analog Pin 4		Analog Pin 5	

Σημείωση: [Η εργασία ζητάει να καταγράφεται float μέτρηση με πχ.2 δεκαδικά ψηφία ακρίβεια] Η αλλαγή σε float ενώ αποστέλλεται integer, με ακρίβεια δύο δεκαδικών ψηφίων, γίνεται με τον εξής τρόπο: Όταν διαβάζεται η μέτρηση του αισθητήρα με την analogRead() και εισάγεται στον τύπο που υπολογίζει το μέγεθος, υπολογίζεται σαν float [Οι float αριθμοί έχουν νόημα υπολογισμού μέχρι το πολύ 7ο δεκαδικό ψηφίο, γιατί στη συνέχεια χάνουν ακρίβεια]. Αυτός ο αριθμός αποτέλεσμα float πολλαπλασιάζεται με 100 και αποστέλλεται σαν integer. Δηλαδή το πλέον πολλαπλασιασμένο ακέραιο μέρος. Στη συνέχεια τον επεξεργάζεται το Processing script, το οποίο αφού τον διαιρέσει με 100 και τον βάλει σε float μεταβλητή, καταχωρεί στη Βάση Δεδομένων.

PROCESSING

Αφού φορτωθεί το Arduino sketch, η καταγραφή ξεκινάει με την εκκίνηση του Processing προγράμματος και συνεχίζει όσο λειτουργεί αυτό.

Στο Processing καλούνται δύο βιβλιοθήκες απαραίτητες για τις δύο βασικές λειτουργίες, τη διασύνδεση με το Arduino board και με τη Βάση Δεδομένων. Οι βιβλιοθήκες αυτές είναι καταχωρημένες στο διαδίκτυο και εισάγονται ως φάκελοι όπως περιγράφεται στο επίσημο κεφάλαιο “How to install a contributed library” [18].

- Processing.serial

  import processing.serial.;*

  Η Serial βιβλιοθήκη εξυπηρετεί την ανάγνωση και εγγραφή δεδομένων προς και από εξωτερικές συσκευές ένα byte τη φορά. Επιτρέπει την αποστολή και τη λήψη δεδομένων. Η βιβλιοθήκη έχει την ευελιξία να επικοινωνεί με μικροελεγκτές και να τους χρησιμοποιεί ως είσοδο ή έξοδο σε προγράμματα Processing. Η σειριακή θύρα είναι εννέα pins I/O που μπορεί να προσομοιωθεί μέσω USB.

  Στη συνέχεια αυτή η βιβλιοθήκη χρησιμοποιείται προκειμένου να κατασκευαστεί ένα καινούργιο στιγμιότυπο Serial το οποίο ουσιαστικά αποτελεί και τη σειριακή μας σύνδεση. Αυτό είναι εφικτό αφού πρώτα οριστεί η θύρα και ο ρυθμός μετάδοσης με την οποία έχει συνδεθεί το Arduino:

  portName = Serial.list()[portIndex];
  myPort = new Serial(this, portName, 9600); // From serial.java , parent typically use "this", rate 9600 is the default, name of the port (COM1 is the default)

  Από αυτή τη βιβλιοθήκη γίνεται χρήση ακόμα δύο συναρτήσεων, της .available() και της .read().

  - .available():
    Από το πηγαίο πρόγραμμα java της βιβλιοθήκης Serial.java-> *public int available() {return (bufferLast - bufferIndex);}* όπου επιστρέφει το μέγεθος σε bytes των δεδομένων που αποστέλλονται. Εδώ όπως εξηγήθηκε θα πρέπει να είναι κάθε φορά 15.

  - .read()
    Επιστρέφει έναν αριθμό μεταξύ 0 και 255 για το επόμενο byte που είναι σε αναμονή στο buffer. Επιστρέφει -1 αν δεν υπάρχει byte, αν και αυτό θα

[18] http://wiki.processing.org/w/How_to_Install_a_Contributed_Library

πρέπει να αποφεύγεται ελέγχοντας με την *.available()* για να δούμε εάν τα δεδομένα είναι διαθέσιμα.

Το processing script δουλεύει ως εξής:

Αφού περιμένει στο buffer να υπάρχουν τουλάχιστον 15 bytes [αυτό πρέπει να αλλάζει ανάλογα με το πόσα και ποια pins έχουμε εν ενεργεία. Εδώ υποτίθεται λειτουργία όλων των pins, analog και digital].

Αν ο πρώτος χαρακτήρας είναι η επικεφαλίδα 'Η', στη συνέχεια καλεί τη συνάρτηση που ονομάζεται *readArduinoInt()* να διαβάσει δύο bytes και να τους μετατρέψει πίσω σε έναν ακέραιο αριθμό, κάνοντας τη συμπληρωματική μαθηματική πράξη που είχε εκτελεστεί από το Arduino για να πάρει τα επιμέρους bits που αντιπροσωπεύουν τα Digital Pins. Οι έξι ακέραιοι αντιπροσωπεύουν τις αναλογικές τιμές από τα αντίστοιχα Analog Pins.

Η συνέχεια είναι η διαδικασία καταχώρησης των δεδομένων στη Βάση Δεδομένων από το Processing πρόγραμμα.

- BezierSQLib 0.2.0

  Για να επιτευχθεί η σύνδεση MySQL και Processing είναι απαραίτητη η βιβλιοθήκη BezierSQLib. Αποτελεί ένα άρτια πετυχημένο εγχείρημα του Florian Jenett για Processing 2+ που υποστηρίζει MySQL, PostgreSQL και SQLite.

  Πρέπει να αρχικοποιηθεί η βιβλιοθήκη και να φτιαχτεί μια σύνδεση με τη Βάση Δεδομένων, δηλαδή ένα στιγμιότυπο της MySQL κλάσης της βιβλιοθήκης. Για να γίνει αυτό από το documentation της βιβλιοθήκης ορίζεται ως εξής μέσω του constructor:

  ```
  MySQL msql;
  …
  String database = "arduinothesisdb";
  String user     = "root";
  String pass     = "";
  msql = new MySQL( this, "localhost", database, user, pass );
  ```

  Η συνάρτηση ορίζεται ως εξής:

  ```
  public MySQL(processing.core.PApplet _papplet,
                  java.lang.String _server,
                  java.lang.String _database,
                  java.lang.String _user,
                  java.lang.String _pass)
  ```

Τα ορίσματα στην καινούργια σύνδεση είναι

_papplet: Συνήθως εισάγεται απλά το “this”
_server: Ο server στον οποίο βρίσκεται η Βάση Δεδομένων, “localhost”
_database: Το όνομα της , “arduinothesisdb”
_user: Το όνομα χρήστη της Βάσης Δεδομένων, “root”
_pass: Κωδικός του χρήστη στη Βάση Δεδομένων [εδώ χωρίς κωδικό, οπότε κενό]

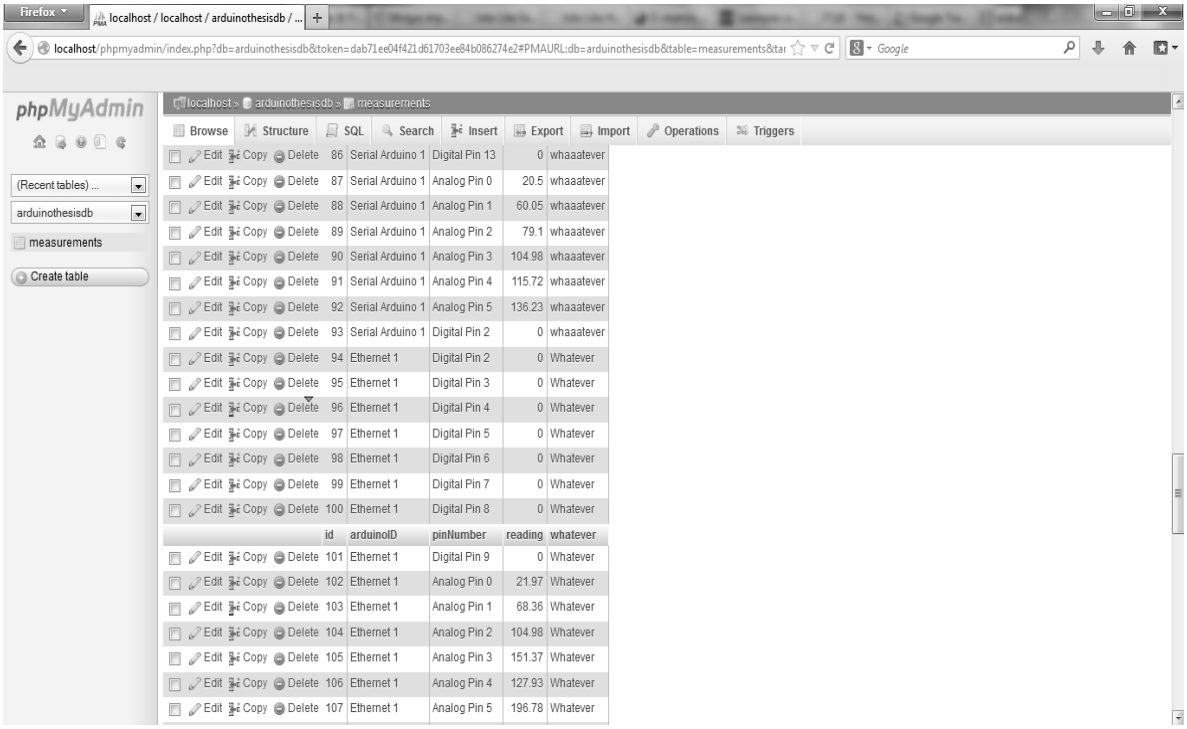
Το επόμενο βήμα είναι ο έλεγχος αν έχει γίνει η σύνδεση με τις παραμέτρους με την boolean συνάρτηση .connect(), η οποία επιστρέφει true/false ανάλογα με το αν η σύνδεση έχει επιτευχθεί ή όχι.

Τέλος καλείται η .execute() που πάιρνει όρισμα την πρόταση που πρέπει να σταλεί στη MySQL, η οποία είναι όπως φαίνεται παρακάτω:

```
if ( msql.connect() ) {

    msql.execute( "INSERT INTO measurements (arduinoID, pinNumber, reading, whatever) VALUES ('"+arduinoName+"','"+pin+"',"+digiValue+", '"+whatever+"');" ); }
```

Ολόκληρος ο κώδικας παραπέμπεται στο Παράρτημα 2.4.

Εν τέλει, στη Βάση Δεδομένων αποθηκεύονται τα ζητούμενα δεδομένα όπως φαίνεται στην εικόνα παρακάτω σε μια τυχαία καταγραφή:



Εικόνα 7: Ενδεικτικές μετρήσεις στη Βάση Δεδομένων Σειριακά και Ethernet

ΚΕΦΑΛΑΙΟ 3

ΕΦΑΡΜΟΓΕΣ-ΠΑΡΑΓΩΓΗ

Είναι πάρα πολύ σημαντικό πως η παρούσα διπλωματική έγινε εξολοκλήρου με Ελεύθερο λογισμικό και λογισμικό ανοιχτού κώδικα και τα υλικά Arduino και το Ethernet Shield που επίσης αποτελούν τμήμα της ίδιας λογικής σε επίπεδο hardware.

Η ίδια η παραγωγική και εφαρμοστική αξία τους αποκτάει μια τελείως διαφορετική διάσταση με αυτόν τον τρόπο. Πλέον εισάγεται ένα ποιοτικό και ποσοτικό μοντέλο ανάπτυξης και χρήσης λογισμικού και υλικού που οριοθετείται πολύ δύσκολα, είναι αενάως αναπτυσσόμενο, πολύ φθηνό και προσβάσιμο, πολύ εύχρηστο και διαποτισμένο από άκρη σε άκρη με την ελευθερία, την ανώτατη δημιουργική αξία και προϋπόθεση.

Πλέον, μετά την διευρυμένη, σχεδόν καθολική αποδοχή της παραπάνω προσπάθειας στο λογισμικό και το υλικό, οποιοδήποτε είδος εργασίας δύναται να είναι ελεύθερο, με αποτέλεσμα ο ορισμός του ελεύθερου λογισμικού να έχει επεκταθεί στον ορισμό των ελεύθερων πολιτισμικών εργασιών που μπορεί να εφαρμοστεί σε όλα τα είδη εργασίας.

3.1

Ελεύθερο/Ανοιχτό Λογισμικό και Υλικό [Free/Open Source Software-Hardware - Creative Commons License]

ΤΟ ΛΟΓΙΣΜΙΚΟ

Στον χώρο της πληροφορικής και των ηλεκτρονικών υπολογιστών, με τον όρο **λογισμικό ανοικτού κώδικα** (αγγλ.: *Open Source Software*, OSS) εννοείται λογισμικό του οποίου ο πηγαίος κώδικας διατίθεται με κάποιον τρόπο ελεύθερα σε όσους ζητούν να τον εξετάσουν, ακόμα και να τον τροποποιήσουν ή αξιοποιήσουν σε άλλες εφαρμογές. Κατά καιρούς έχουν εμφανιστεί αρκετές διαφορετικές άδειες χρήσης σχεδιασμένες να συνοδεύουν λογισμικό ανοιχτού κώδικα.

Παραφράζοντας σχετικά την τοποθέτηση της ελληνικής κοινότητας Ελεύθερου λογισμικού θα γίνει σαφές γιατί αποτελεί ένα πολύ σημαντικό στοιχείο αυτής της εργασίας.

Το λογισμικό ανοικτού κώδικα δεν σημαίνει απαραιτήτως δωρεάν λογισμικό, ούτε ελεύθερο λογισμικό σύμφωνα με τον ευρύ ορισμό, αλλά αναφέρεται μόνο στο γεγονός πως επιτρέπεται σε κάθε χρήστη να εξετάσει και να χρησιμοποιήσει τη γνώση και τις δυνατότητες που προσφέρει ο παρεχόμενος πηγαίος κώδικας. Στην πράξη, τα περισσότερα προγράμματα ανοιχτού κώδικα παρέχονται δωρεάν και μπορούν να χαρακτηριστούν ελεύθερα.

Από την άλλη πλευρά αξίζει να εξεταστεί ο ορισμός, η πραγματική διάσταση του Ελεύθερου Λογισμικού, που δίνεται από το Ίδρυμα Ελεύθερου Λογισμικού και είναι γενικά αποδεκτός.

Το Ελεύθερο λογισμικό είναι ζήτημα ελευθερίας, όχι κόστους. Για να κατανοηθεί ο όρος αυτός θα πρέπει να ερμηνευτεί η λέξη free όπως ο ελεύθερος λόγος (free speech) και όχι όπως πχ. η δωρεάν μπύρα (free beer). Το ελεύθερο λογισμικό δεν έχει σχέση δωρεές ή παραχωρήσεις, διότι αυτοί υπονοούν ότι το θέμα είναι το κόστος και όχι η ελευθερία.

Το Ελεύθερο λογισμικό παρέχει στους χρήστες την ελευθερία να εκτελούν, αντιγράφουν, διανέμουν, μελετούν, τροποποιούν και βελτιώνουν το Ελεύθερο λογισμικό. Για την ακρίβεια, αναφέρεται σε τέσσερις βασικές ελευθερίες:

- Την ελευθερία να εκτελείται το πρόγραμμα για οποιονδήποτε σκοπό (ελευθερία 0).
- Την ελευθερία να μελετάται ο τρόπος λειτουργίας του προγράμματος και να προσαρμόζεται στις ανάγκες του χρήστη (ελευθερία 1). Η πρόσβαση στον πηγαίο κώδικα είναι προϋπόθεση για να ισχύει κάτι τέτοιο.

- Την ελευθερία να αναδιανέμονται αντίγραφα του προγράμματος. (ελευθερία 2).
- Την ελευθερία να βελτιώσει κανείς το πρόγραμμα και να δημοσιεύσει τις βελτιώσεις που έχει κάνει στο ευρύ κοινό, ώστε να επωφεληθεί ολόκληρη η κοινότητα (ελευθερία 3). Η πρόσβαση στον πηγαίο κώδικα είναι προφανής προϋπόθεση για να ισχύει κάτι τέτοιο.

Ένα πρόγραμμα θεωρείται ελεύθερο λογισμικό όταν οι χρήστες του έχουν όλες τις παραπάνω ελευθερίες. Επομένως, οποιοσδήποτε είναι ελεύθερος να αναδιανέμει αντίγραφα, με ή χωρίς τροποποιήσεις, δωρεάν ή χρεώνοντας για την διανομή, στον οποιονδήποτε και οπουδήποτε. Η ελευθερία για όλα τα παραπάνω σημαίνει (μεταξύ άλλων) πως δεν χρειάζεται εξουσιοδότηση ή πληρωμή σε κάποιον ώστε να ληφθεί η ανάλογη άδεια.

Η ελευθερία αυτή σημαίνει επίσης να γίνονται τροποποιήσεις και να χρησιμοποιούνται ιδιωτικά στην δουλειά ή για διασκέδαση, χωρίς να χρειάζεται αναφορά οπουδήποτε.

Η ελευθερία της χρήσης ενός προγράμματος σημαίνει πως δίδεται η ελευθερία σε κάθε άτομο ή επιχείρηση να το χρησιμοποιήσει σε κάθε είδους υπολογιστικό σύστημα, για κάθε είδος εργασίας χωρίς να είναι υποχρεωμένο να επικοινωνήσει εκ των προτέρων με τον προγραμματιστή ή με κάποια άλλη οντότητα. Σε αυτή την ελευθερία, είναι η άποψη *του χρήστη* που έχει σημασία, και όχι η άποψη *του κατασκευαστή*. Ο χρήστης είναι ελεύθερος να εκτελεί το πρόγραμμα για τους δικούς του λόγους, και αν το διανέμει σε οποιονδήποτε άλλο άνθρωπο, τότε και εκείνος είναι ελεύθερος να το εκτελεί για τους δικούς του λόγους, δίχως να έχει κανείς το δικαίωμα να του υποβάλει κυρώσεις.

Για να ισχύουν πρακτικά οι τέσσερις βασικές ελευθερίες, και να μπορεί οποιοσδήποτε να δημοσιεύει βελτιωμένες εκδόσεις, θα πρέπει να έχει πρόσβαση στον πηγαίο κώδικα του προγράμματος. Επομένως, η πρόσβαση στον πηγαίο κώδικα είναι απαραίτητη προϋπόθεση στο ελεύθερο λογισμικό.

Ωστόσο, μερικοί κανόνες που αφορούν τον τρόπο με τον οποίο διανέμεται το ελεύθερο λογισμικό είναι αποδεκτοί όταν δεν συγκρούονται με τις τέσσερις βασικές ελευθερίες. Για παράδειγμα, στην άδεια αντιγραφής (copyleft) όταν αναδιανέμεται κάποιο πρόγραμμα, δεν μπορεί κανείς να προσθέσει περιορισμούς ώστε να μειώσει από τον χρήστη τις τέσσερις βασικές ελευθερίες. Αυτός ο κανόνας δεν συγκρούεται με τις βασικές ελευθερίες, αλλά τις προστατεύει. Αυτή η άδεια αντιγραφής είναι η γνωστή GNU General Public License (GPL).

Ελεύθερο λογισμικό δεν σημαίνει μη-εμπορικό. Ένα ελεύθερο πρόγραμμα θα πρέπει να είναι διαθέσιμο για εμπορική χρήση, εμπορική ανάπτυξη ή εμπορική διανομή. Η εμπορική ανάπτυξη του ελευθέρου λογισμικού δεν είναι ασυνήθιστη. Αντιθέτως, τέτοιου είδους ελεύθερο λογισμικό είναι σημαντικό να υπάρχει.

Στη συγκεκριμένη εργασία λοιπόν τα εργαλεία που χρησιμοποιήθηκαν έχουν όλα αντίστοιχους όρους υπόστασης.

- Η πλατφόρμα Processing αποτελεί εξ ολοκλήρου ελεύθερο λογισμικό, με την άδεια-πλήρη προστασία των ελευθεριών που προαναφέρθηκαν, της GNU General Public License [19].

- Το πλήρες λογισμικό του Arduino είναι επίσης ελεύθερο υπό την άδεια GNU General Public License [20].

- Η PHP αποτελεί ανοιχτού τύπου λογισμικό με κάποια δικαιώματα κατοχυρωμένα. Μόνο η PHP 3 είχε GNU Licence αλλά στις επόμενες εκδόσεις επανήλθε σε κάποιους περιορισμούς [21].

- Ο Apache έχει δική του άδεια, η οποία το καθιστά σχεδόν ταυτόσημο με ελεύθερο λογισμικό, δηλαδή πιο μακριά από απλά ανοιχτό λογισμικό. Η Apache licence είναι συμβατή και συνδυάζεται με την GNU Licence [22].

- Η MySQL αποτελεί ένα από πιο πετυχημένα εγχειρήματα ελεύθερου λογισμικού. Επίσης ο πηγαίος κώδικας υπό την GNU GPL [23].

- Ο phpMyAdmin είναι επίσης ελεύθερο λογισμικό υπό την GNU GPLicense [24].

- Τέλος ο Firefox αποτελεί σαν τον Apache ένα υβρίδιο με κάποια κλειστά σημεία που επίσης προσιδιάζει πολύ περισσότερο σε ελεύθερο λογισμικό και είναι συμβατό με την GNU GPL [25].

[19] http://wiki.processing.org/w/FAQ
[20] http://arduino.cc/en/Main/FAQ
[21] http://www.php.net/license/
[22] http://www.apache.org/licenses/
[23] http://www.mysql.com/about/legal/licensing/index.html
[24] http://www.mysql.com/about/legal/licensing/index.html
[25] http://www.mozilla.org/en-US/foundation/licensing/

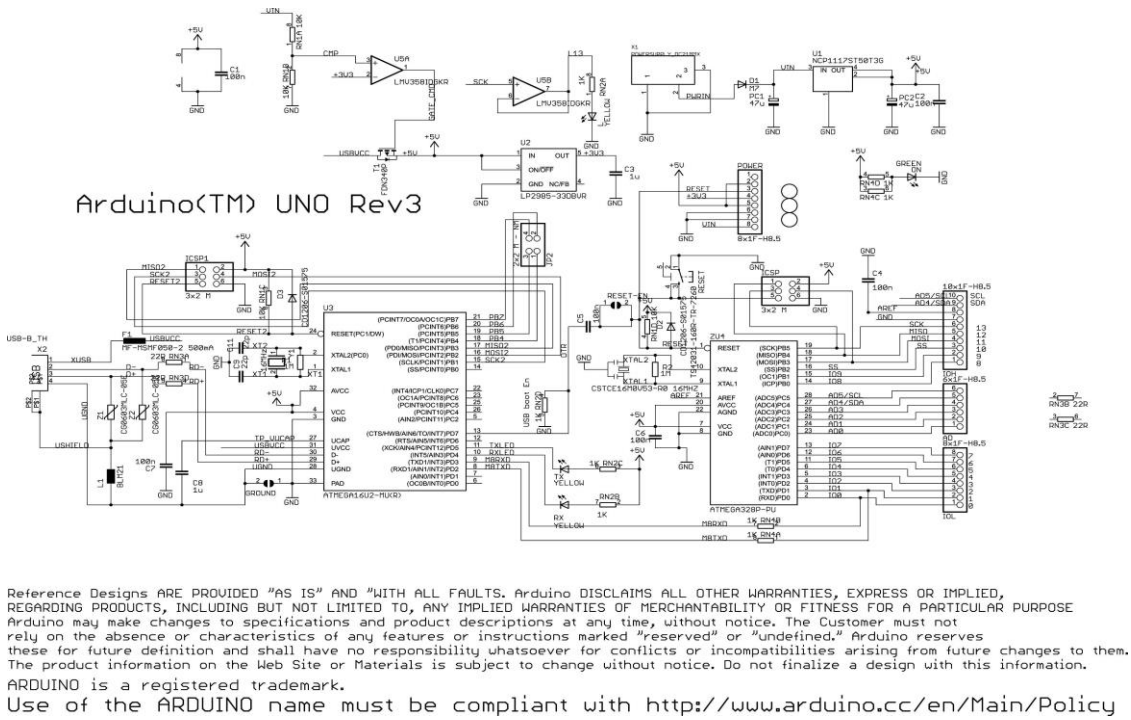
ΤΟ ΥΛΙΚΟ – HARDWARE

Η πλατφόρμα Arduino είναι η ίδια αρκετά χρήσιμη για εργασίες με μικροελεγκτές, αλλά αυτό από μόνο του δεν είναι αρκετό για να προωθήσει τη δημοτικότητα και την ευρεία υιοθέτηση της πλατφόρμας. Αντί να κλείνεται το hardware σχεδιασμό του και το περιβάλλον ανάπτυξης, το σύνολο του Arduino είναι βαθιά ριζωμένο στην αναδυόμενη πρακτική του open-source hardware.

Το open-source hardware δεσμεύει ένα κατανεμημένο μοντέλο ανάπτυξης υλικού με συνεισφέροντες γενικά από διαφορετικά μέρη του κόσμου . Αντί για κλειστά συστήματα, τα έργα ανοικτού κώδικα αναπτύσσουν την ατομική ελευθερία με την ανοιχτή πρόσβαση στα αρχεία προέλευσης του σχεδιασμού, προκειμένου να γίνουν βελτιώσεις, και να επιστρέψουν με τις βελτιώσεις αυτές σε μια μεγαλύτερη κοινότητα.

Το Arduino ενσωματώνει ριζικά αυτή η προσδοκία για διαφάνεια στο σχεδιασμό - αρχιτεκτονική, τη συνεργασία και τη φιλοσοφία. Αυτή η δημιουργική ανατροφοδότηση εξασφαλίζει ότι κάθε εμπνευσμένη καινοτομία που προέρχεται από την πλατφόρμα Arduino θα συναντηθεί με ακόμα πιο ευφάνταστες χρήσεις για ακόμη πιο νέα πράγματα.

Αντίστοιχα συμβαίνει και με το ίδιο το υλικό, την πλακέτα και το μικροελεγκτή. Τα σχέδια του Arduino και του Ethernet Shield είναι ανοιχτά και κάτω από την άδεια Creative Commons Attribution-ShareAlike 3.0 License [26]. Αναπαράγεται και προσαρμόζεται ελεύθερα με βάση τα σχέδιά του σε EAGLE files [27] , και Schematic [28] :



Εικόνα 8: Arduino Uno rev3 Reference Designs

[26] http://creativecommons.org/licenses/by-sa/3.0/

[27] arduino.cc/en/uploads/Main/arduino_Uno_Rev3-02-TH.zip

[28] http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

3.2

Επιστημονικές εφαρμογές

Οι μικροελεγκτές σε διατάξεις όπως το Arduino board έχουν αναρίθμητες εφαρμογές. Η χρηστική αξία ενός τέτοιου εργαλείου δεν θα ήταν υπερβολή να πει κανείς ότι περιορίζεται από δύο μόνο παράγοντες: Τα φυσικά του χαρακτηριστικά και τη φαντασία του μηχανικού ή χομπίστα ηλεκτρονικού ή καλλιτέχνη κλπ που το χρησιμοποιεί.

Όσον αφορά την ίδια την επιστήμη, το μικρότερο παράδειγμα, η παρούσα διπλωματική εκπονείται με τέτοιο στόχο.

Οι αισθητήρες, το δίκτυο μικροελεγκτών με διατάξεις μετρήσεων και ελέγχου έχουν απεριόριστες εφαρμογές. Η μελέτη των καιρικών φαινομένων σε περιοχές, ο έλεγχος των φυσικών και τεχνητών συνθηκών περιβάλλοντος σε κλειστούς χώρους όπως εργαστήρια, η δευκόλυνση στη γεωπονία, στα θερμοκήπια και άρα στην πρωτογενή ακόμα παραγωγή.

Οι δυνατότητες του Arduino ξεπερνάνε τη μικρή κλίμακα των ατομικών εφαρμογών, των μικρών εργαστηριακών πειραμάτων.

Ο αυτοματισμός σε βασικές αστικές και βιομηχανικές μονάδες είναι στη σημερινή περίοδο ένα τομέας που ανθεί. Εφαρμογές ασφάλειας, εξοικονόμησης ενέργειας, λεπτομερούς ελέγχου πλέον προβλέπονται στην ίδια μελέτη με πχ. τα υδραυλικά συστήματα.

Κεντρικό παράδειγμα εφαρμογής Arduino μικροελεγκτών αποτελούν τα λεγόμενα «έξυπνα σπίτια». Αισθητήρες φωτός υγρασίας και θερμοκρασίας μπορούν να εξοικονομούν ενέργεια ελέγχοντας τη θέρμανση, το κλιματισμό, το φωτισμό.

Μια ακόμη έξυπνη χρήση αποτελεί η εφαρμογή του στην οδήγηση. Μετρώντας την κατανάλωση καυσίμου σε ζωντανό χρόνο και με συγκεκριμένες ενδείξεις εύκολα μπορεί να σχεδιαστεί ένα ολολκηρωμένο σύστημα που να βοηθάει τον οδηγό να μάθει να αποφεύγει λειτουργίες που καταναλώνουν περισσότερα καύσιμα κάνοντας οικονομία.

Οι εφαρμογές και οι πιθανές διατάξεις Arduino, αισθητήρων και άλλων οργάνων στην έρευνα, στην παραγωγή, στον έλεγχο, στην ασφάλεια κλπ θα μπορούσαν να καταγράφονται για χιλιάδες σελίδες μέσα σε λίγα μόνο χρόνια πρακτικής.

Το κόστος, η ευκολία σχεδιασμού, και κυρίως η ελευθερία χρήσης και μεταβολής των αντικειμένων και του κώδικα είναι ασυναγώνιστη πηγή έμπνευσης και υλοποίησης.

Παρόλα αυτά, οι παραπάνω εφαρμογές σε μέγεθος και βάθος φαντασίας, πέραν δηλαδή από παραγωγική και επιστημονική αξία χρήσης, ωχριούν μπροστά στο επόμενο υπο-κεφάλαιο.

3.3

Ατομικές εφαρμογές

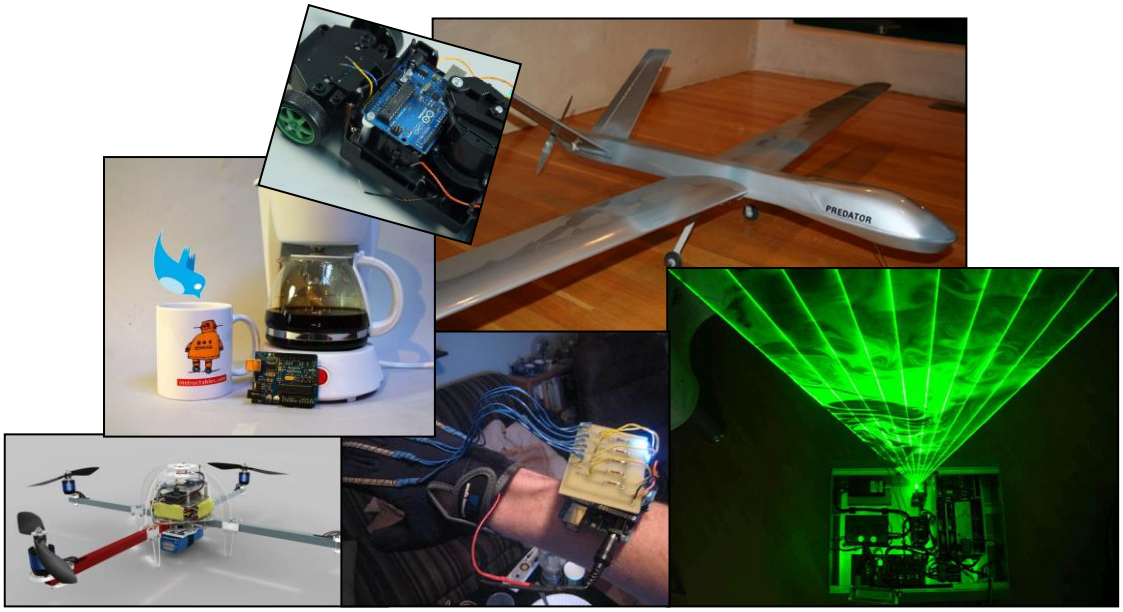
Αντίθετα με άλλες πλατφόρμες και με ό,τι αποτελούσε μια κοινώς αποδεκτή πραγματικότητα πως οι ηλεκτρονικές διατάξεις, πολύ περισσότερο οι μικροεπεξεργαστές και μικροελεγκτές, αφορούν μια πολύ περιορισμένη και αρκετά καταρτισμένη κοινότητα με ειδικά χαρακτηριστικά, το Arduino τα άλλαξε όλα σε τεράστιο βαθμό.

Η ταυτόχρονη ανάπτυξη του διαδικτύου δεν είναι τυχαίος παράγοντας. Οδήγησε στην παραγωγικοποίηση και εμπορική διάχυση με ταχείς ρυθμούς πλακέτων Arduino, η εξάπλωση των οποίων με τη σειρά της ανάδρασε με εκατομύρια ατομικές αφαρμογές. Αυτή η συνεχόμενη αναίρεση του προηγούμενου, αυτή η αέναη ανάπτυξη όπως προαναφέρθηκε έχει τις ρίζες της στην ελευθερία.

Αυτή η ελευθερία έχει εντείνει εκπληκτικά την ατομική δημιουργικότητα με εφαρμογές σε όλους τους τομείς.

Η βελτίωση της καθημερινής ζωής και του σπιτιού, οι ελεγχόμενες μικρές καλλιέργιες, οι αυτοματισμοί σε παιχνίδια, η ανάπτυξη καινούργιων παιχνιδιών, εκατοντάδες μουσικά όργανα και μουσικές εφαρμογές, καλλιτεχνικές δημιουργίες, οργάνωση χώρων με φωτισμό, θεατρικές παραστάσεις, όλα μπορεί πραγματικά κάποιος να τα ανιχνεύσει στο διαδίκτυο, όπου τα μοιράζονται οι δημιουργοί. Η θέση του Arduino είναι κεντρική και βοηθητική συνάμα. Εκτοξεύει τις δυνατότητες της φαντασίας, εξοικιώνει ανθρώπους με τα ηλεκτρονικά συστήματα νέας γενιάς, τον αυτοματισμό, γεννάει ιδέες.

Ασύρματα οχήματα με σερβομηχανισμούς, ρουχισμός, εκπαιδευτικά μηχανήματα, όλα είναι πλέον εύκολα προς κατασκευή σε ιδιωτικό χώρο χωρίς ιδιαίτερες υποδομές.



Εικόνα 9: Ενδεικτικές ατομικές εφαρμογές

ΚΕΦΑΛΑΙΟ 4

ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την ενασχόληση με την εργασία αυτή μπορεί κάποιος να βγάλει πολλά επί μέρους συμπεράσματα. Η αξία χρήσης ενός δικτύου μικροελεγκτών με αισθητήρες και όργανα που καταχωρούν σε Βάση Δεδομένων είναι τεράστια.

Δύσκολα περιβάλλοντα, χρονοβόρα πειράματα και περιορισμένοι πόροι είναι το εύφορο έδαφος για τη χρήση της παρούσας εργασίας.

Η διευκόλυνση των μετρήσεων, η αυτοματοποίηση και η ενιαιοποίηση διαδικασιών είναι τα στοιχήματά της επίσης.

Ας επιτραπεί σαν κεντρικό συμπέρασμα όμως κάτι βαθύτερο από τις εφαρμογές τις ίδιες.

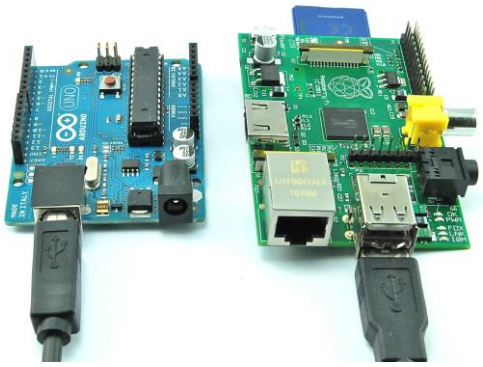
Το Arduino είναι καταδικασμένο να αντικατασταθεί σύντομα. Η εντατική ανάπτυξη και πρακτική πάνω του θα αναιρέσει κάποια στιγμή ακόμα και τον κεντρικό του πυρήνα.

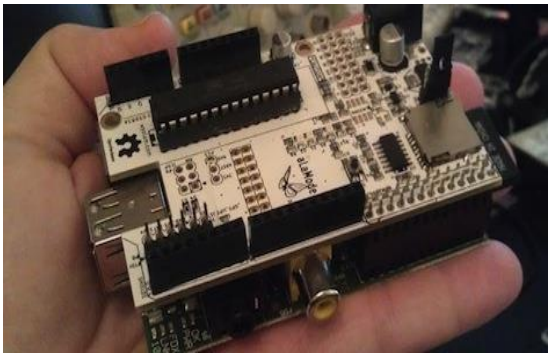
Αυτή είναι η μαγεία της ελεύθερης ανάπτυξης, της δημιουργίας.

Ένα παράδειγμα:

Το Arduino είναι κατά γενική ομολογία ικανό εργαλείο για ηλεκτρονικά project. Υπάρχει ένα άλλο εργαλείο πολύ πρόσφατο και επίσης με διαρκή ανάπτυξη που βρίσκεται στην ίδια κατηγορία με αρκετές όμως διαφορές, και για σχετικά διαφορετική χρήση, το πολύ χαμηλού κόστους Raspberry Pi. Το board αυτό αποτελεί έναν ολοκληρωμένο, λειτουργικό μικρό υπολογιστή. Οι εφαρμογές που έχει είναι επίσης ασυγκράτητες, και πολλοί έχουν αναρωτηθεί γιατί Arduino και όχι Raspberry Pi ή το αντίστροφο. Δεν χρειάζεται να μπει κανένας πλέον σε λεπτομέρειες και πολύ περισσότερο να πάρει θέση. Υπάρχει η λύση σε όλα τα προβλήματα και αυτή είναι η ενσωμάτωση του ενός στο άλλο. Πιο συγκεκριμένα παρόλο που το Raspberry Pi δεν είναι ανοιχτό hardware, επειδή το Arduino είναι, ήδη κυκλοφορούν ελεύθερα οι εξείς μέθοδοι:

- Το project AlaMode τοποθετεί διαμορφωμένο κλώνο Arduino πάνω από το Pi, δίνοντας άμεση πρόσβαση σε όλες τις συνήθεις λειτουργίες Arduino [29].
- Το project του 'Dr. Monk' επιτρέπει τη σειριακή σύνδεση με USB μεταξύ Arduino και Raspberry Pi με μια διεπαφή σε Python [30].





Οπότε οι δυνατότητες καθενός συμπυκνώνονται σε ένα εργαλείο ικανό για εκατομμύρια εφαρμογές.

[29] http://hackaday.com/2012/07/23/the-proper-way-to-put-an-arduino-in-a-raspberry-pi/

[30] http://www.doctormonk.com/2012/04/raspberry-pi-and-arduino.html

ΠΑΡΑΡΤΗΜΑ 1 [κώδικες]

2.1 Κώδικας Arduino

```
/*
  Arduino_mult_sensors_odconnect_THIS.ino
  Created by OD, June, 2013.
  Released into the public domain.
*/
#include <SPI.h>
#include <Ethernet.h>
#include <Od_connect.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Used for ETHERNET
//byte mac[]={0,0,0,0,0,0}; //Used for SERIAL

byte ip[] = { 192,168,3,3 };
byte serverName[] = { 192,168,3,2 };
//Strings "arduinoName" and "whatever" are used in Ethernet only entries
String arduinoName = "Ethernet%201"; // For every Arduino we need to change this
String whatever = "Whatever";

Od_connect con(arduinoName, whatever, ip, mac, serverName);

void setup()
{
  if (mac[0] != 0){
    Ethernet.begin(mac, ip);
    delay(1000); // a second for the ethernet hardware to initialise
  }
  Serial.begin(9600);
}
void loop()
{
  if (mac[0] != 0){
    con.connect_ethernet();
  }
  else{
```

```
    con.connect_serial();
  }
  //Every Arduino that has this script loaded,
  //will make a new instance, so no "MULTIPLE" code is needed
}
```

2.2 Od_connect.h

```
/*
  Od_connect.h - Library for connecting.
  Created by OD, June, 2013.
  Released into the public domain.
*/
#ifndef Od_connect_h  // These 2 lines prevent the code from being included into the
program binary multiple times
#define Od_connect_h

#include "Arduino.h"

#include "C:\Users\BrotherInArms\Desktop\IP SENSORS\arduino-
1.0.5\libraries\Ethernet\Ethernet.h"

class Od_connect
{
//everything under here is public, accessible outside of the class
  public:
                    Od_connect(String arduinoName, String whatever, uint8_t ip[], uint8_t
mac[], uint8_t serverName[]);

                    void connect_serial();
                    void connect_ethernet();

//everything under here is private, only accessible inside the class
  private:
    String _arduinoName;
                    String _whatever;
                    uint8_t _mac[6];
                    uint8_t _ip[4];
                    uint8_t _serverName[4];

};

#endif  //ends the #ifndef preprocessor directive
```

2.3 Od_connect.cpp

```
/*
  Od_connect.cpp - Library for connecting.
  Created by OD, June, 2013.
  Released into the public domain.
*/

#include "Arduino.h"
#include "Od_connect.h"

#include "C:\Users\BrotherInArms\Desktop\IP SENSORS\arduino-
1.0.5\libraries\Ethernet\Ethernet.h"

const char HEADER = 'H';  // a single character header to indicate the start of a message

String nameAndIPString, stringOne, stringTwo, stringThree, stringFour ;

// Constructor ////////////////////////////////////////////
// Function that handles the creation and setup of instances
Od_connect::Od_connect(String arduinoName, String whatever, uint8_t ip[4], uint8_t
mac[6], uint8_t serverName[4])
{
                    // initialize this instance's variables
                    _arduinoName= arduinoName;
                    _whatever= whatever;

                    for(int i=0; i<4; i++){
                    _ip[i]=ip[i];
                    }
                    for(int i=0; i<6; i++){
                    _mac[i]=mac[i];
                    }
                    for(int i=0; i<4; i++){
                    _serverName[i]=serverName[i];
                    }

}

void Od_connect::connect_serial()
{
```

```
                Serial.write(HEADER); // send the header
                //DIGITAL PINS 2-13
                // put the bit values of the pins into an integer
                int values = 0;
                int bit = 0;
                //DIGITAL PINS
                for(int i=2; i <= 13; i++)  //We suppose that there is no ethernet shield on
when using serial connection.
                {
                        bitWrite(values, bit, digitalRead(i));  // set the bit to 0 or 1 depending

                // on value of the given pin
                        bit = bit + 1;         // increment to the next bit
                }
                //send the two bytes that comprise an integer value with all the bits from the
digi pins to the serial port
                Serial.write(lowByte(values));  // send the low byte
                Serial.write(highByte(values)); // send the high byte


                //ANALOG PINS 0-5
                for(int i=0; i < 6; i++)
                {
                        float floatValues = ( analogRead(i) * 500.00) / 1024.00;
                        //FLOAT DIRTY TRICK
                        values=floatValues*100; // Dirty trick to send int and then divide so
we get float .2digits
                        //send the two bytes that comprise an integer value to the serial port
for every one analog pin
                        Serial.write(lowByte(values));  // send the low byte
                        Serial.write(highByte(values)); // send the high byte

                }
                delay(1000); //send every second
}


// function to send through ethernet the sensor data and store them in database
void Od_connect::connect_ethernet()
{
                EthernetClient client;  //New instance of the class EthernetClient
```

```
		float floatValues;
		int val = 0;
		if(client.connect( _serverName, 80 )) // If it is connected with the server IP and the HTTP port 80
	{
				//DIGITAL PINS
				for(int i=2; i <= 9; i++)   // Digital pins 10,11,12,13 are used by the ethernet shield (SPI). Not to be used otherwise
				{
					val = digitalRead(i);
					String iii = String(i); // int to string
					String digipin = "Digital%20Pin%20"+ iii;

					Serial.println("connected...");
					client.print("GET /arduinoThesis.php?readingcpp=");  // Start making HTTP request
					client.print(floatValues);
					client.print("&arduinoIDcpp=");
					client.print(_arduinoName);
					client.print("&pinNumbercpp=");
					client.print(digipin);
					client.print("&whatevercpp=");
					client.print(_whatever);
					client.println(" HTTP/1.1");
					client.println("Host: localhost");
					client.println();
					delay(1000);
				}

				//ANALOG PINS 0-5
				for(int j=0; j < 6; j++)
				{
					floatValues = ( analogRead(j) * 500.00) / 1024.00;
					String ii = String(j); // int to string
					String pin = "Analog%20Pin%20"+ ii;

					Serial.println("connected...");
					client.print("GET /arduinoThesis.php?readingcpp=");
					client.print(floatValues);
					client.print("&arduinoIDcpp=");
```

```
                                        client.print(_arduinoName);
                                        client.print("&pinNumbercpp=");
                                        client.print(pin);
                                        client.print("&whatevercpp=");
                                        client.print(_whatever);
                                        client.println(" HTTP/1.1");
                                        client.println("Host: localhost");
                                        client.println();
                                        delay(1000);
                                }
                        }
                else if (!client.connected()) {
   Serial.println("connection failed");
                Serial.println();
   Serial.println("disconnecting.");
   client.stop();
   for(;;)
    ;
   }
}
```

2.4 Processing

```
/*
  serial_Processing_mult_sensors_odconnect_THIS.pde : Processing script
  Created by OD, June, 2013.
  Released into the public domain.
*/

import de.bezier.data.sql.*;
//Documentation for SQL-Processing package
//http://bezier.de/processing/libs/sql/documentation/de/bezier/data/sql/package-summary.html

import processing.serial.*;
Serial myPort;       // Create object from Serial class
// Serial myPort2 // MULTIPLE

int portIndex = 0;  // select the COM port, 0 is the first port
//int nextPortIndex = 2 // COM port of the next arduino MULTIPLE

float valFloat;
String portName;
//String nextPortName; // MULTIPLE

char HEADER = 'H';
MySQL msql;

void setup()
{
  size(200, 200);
  // Open whatever serial port is connected to Arduino.
  portName = Serial.list()[portIndex];
  //nextPortName = Serial.list()[nextPortIndex];  // MULTIPLE

  myPort = new Serial(this, portName, 9600); // From serial.java , parent typically use "this", rate 9600 is the default, name of the port (COM1 is the default)
  //myPort2 = new Serial(this, nextPortIndex, 9600); // MULTIPLE

}

void draw()
```

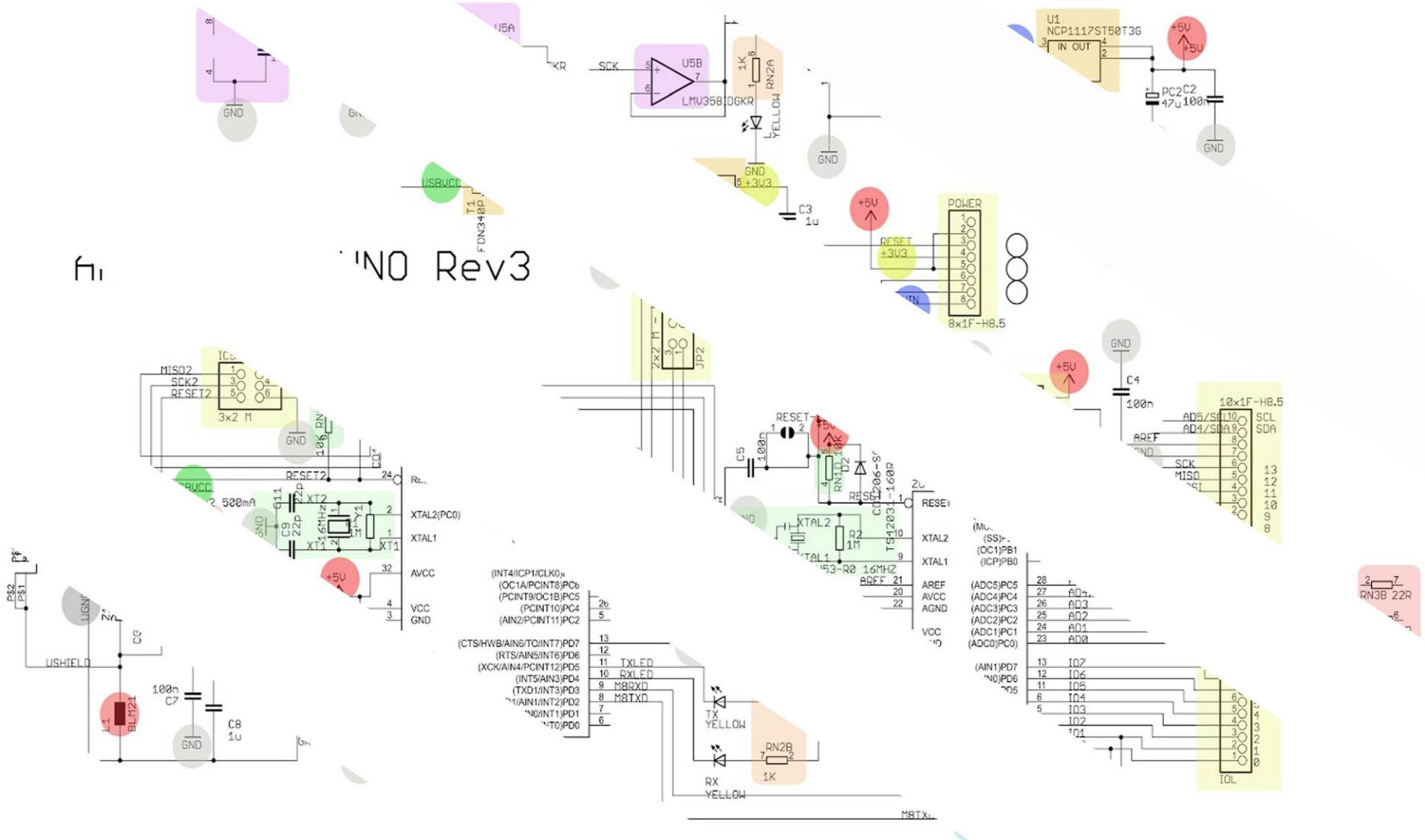
```
{
float val;

  if ( myPort.available() >= 15)  // wait for the entire message to arrive
  //(1 byte for the header, 2 bytes for the digital pin values, and 12 bytes for the six analog integers.bytes)
  //Source of .available : Serial.java of Processing: public int available() {return (bufferLast - bufferIndex);}
  {
    if( myPort.read() == HEADER) // if it is the header H
    {
      val = readArduinoInt();

      //Initialise DEBEZIER LIBRARY
      String database = "arduinothesisdb";
      String user     = "root";
      String pass     = "";
      msql = new MySQL( this, "localhost", database, user, pass );

      String arduinoName = "Serial Arduino 1";
      String whatever = "whaaatever";

      // enter in DB the digital values
      for(int j=2, bit=1; j <= 13; j++){
        String pin = "Digital Pin "+ j;
        int intVal=int(val);
        int digiValue = (intVal & bit); // 1 AND 1 = 1 , otherwise 0

        bit = bit * 2; //shift the bit to the next higher binary place

        if ( msql.connect() ) {
        msql.execute( "INSERT INTO measurements (arduinoID, pinNumber, reading, whatever) VALUES ('"+arduinoName+"','"+pin+"',"+digiValue+", '"+whatever+"');" );

        }
      }
      println();

      // print and enter in DB the six analog values
      for(int i=0; i < 6; i ++){
```

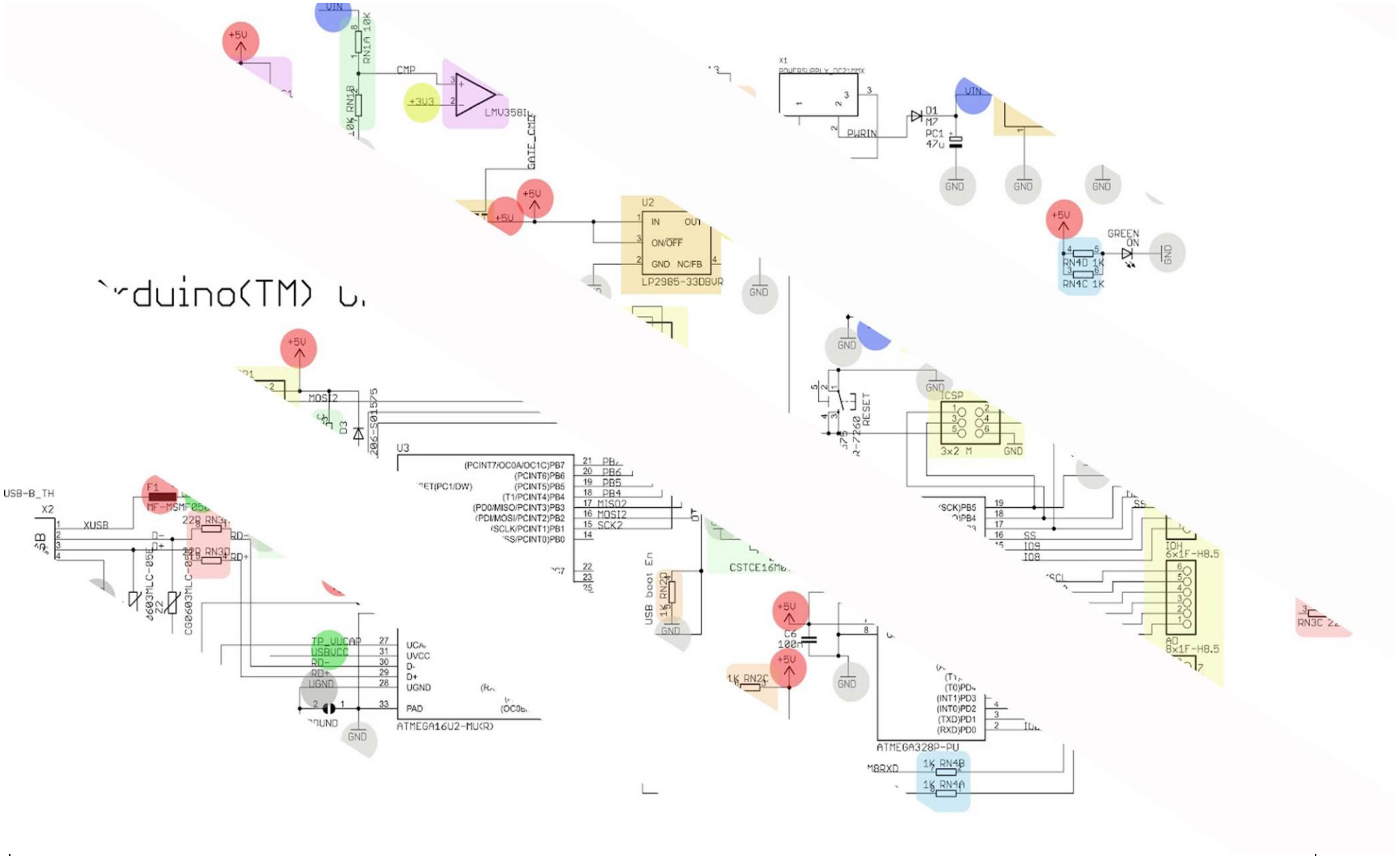
```
      val = readArduinoInt(); //The sensor output value as int
      valFloat = val/100; //The sensor output value as float
      String pin = "Analog Pin "+ i;

      if ( msql.connect() ) {
        msql.execute( "INSERT INTO measurements (arduinoID, pinNumber, reading,
whatever) VALUES ('"+arduinoName+"','"+pin+"',"+valFloat+", '"+whatever+"');" );
        }
     }
     println("----");
   }
  }
  /* // MULTIPLE
  if ( myPort2.available() >= 15)  // wait for the entire message to arrive
  {
    if( myPort2.read() == HEADER) // is this the header
    {   //ALL SAME AS ABOVE FOR THE SECOND ARDUINO ETC FOR 3RD ...
   // CHANGE INNER FUNCTION readArduinoInt() to readArduinoInt2()
    }
  }
 */
}

// return integer value from bytes received from serial port (in low,high order)
// On the Arduino Uno (and other ATMega based boards) an int stores a 16-bit (2-byte) value
// http://arduino.cc/en/Reference/int
int readArduinoInt()
{
  int val;      // Data received from the serial port
  val = myPort.read();            // read the least significant byte
  val =  myPort.read() * 256 + val; // add the most significant byte
  return val;
}
/* // MULTIPLE
int readArduinoInt2()
{
  int val;      // Data received from the serial port
  val = myPort2.read();           // read the least significant byte
  val =  myPort2.read() * 256 + val; // add the most significant byte
  return val;
}*/
```

ΠΑΡΑΡΤΗΜΑ 2 [βιβλιογραφία]

- Apache Documentation, http://httpd.apache.org/docs/
- Arduino Documentation, http://arduino.cc/en/Reference/HomePage
- ATMEL, 2009, documentation ATMEGA328P.pdf
- Banzi Massimo, 2008, O'Reilly, Getting Started with Arduino
- Banzi Massimo, 2006, Interaction Design Lab, Arduino Workshop
- BezierSQLib 0.2.0 documentation, http://bezier.de/processing/libs/sql/documentation/index.html
- C++ Documentation, http://www.cplusplus.com/doc/
- Elmasri R. / S.B. Navathe, 1998, (μετάφραση Μ. Χατζόπουλος), Εκδόσεις Δίαυλος, Θεμελιώδεις Αρχές Συστημάτων Βάσεων Δεδομένων
- Evans Brian, 2007, Arduino programming notebook
- Evans Brian, Apress - Beginning Arduino Programming
- Fabian Winkler, 2007, Envision Art, Arduino Workshop
- Free Software, Free Society: Selected Essays of Richard M. Stallman, 2002, GNU Press, http://www.gnu.org/philosophy/fsfs/rms-essays.pdf
- Greenberg, I., 2007, Processing: Creative Coding and Computational Art
- Igoe Tom, 2007, O'Reilly, Making Things Talk
- Lang T. T., 2000, Ηλεκτρονικά Συστήματα Μετρήσεων, Εκδόσεις Τζιόλα
- Lessig Lawrence, 2004, Free Culture, The penguin press, New York
- Margolis Michael, 2011, O'Reilly, Arduino Cookbook 2nd Edition
- Mellis D. / Banzi M. / Cuartielles D. / Igoe T. ,2007, Arduino: An Open Electronics Prototyping Platform
- MySQL Documentation, http://dev.mysql.com/doc/
- Noble Joshua, 2009, O'Reilly, Programming Interactivity
- oomlout.com , Arduino Experimenter's Guide
- Oxer J. / Blemings H. , 2009, Apress - Beginning Arduino Programming

- Oxer J. / Blemings H. , 2009, Apress - Practical Arduino Cool Projects for Open Source Hardware
- PHP Documentation, http://www.php.net/manual/en/
- phpMyAdmin Documentation, http://www.phpmyadmin.net/home_page/docs.php
- Processing Documentation, http://processing.org/reference/
- Ramakrishnan R. / J. Gehrke, 2002, Συστήματα Διαχείρισης Βάσεων Δεδομένων, (μετάφραση Δ. Δέρβος, Α. Ευαγγελίδης) Εκδόσεις Τζιόλα
- Tanenbaum Andrew S., 2011, Δίκτυα Υπολογιστών, Εκδόσεις Κλειδάριθμος
- Texas Instruments, 2000, LM35 Precision Centigrade Temperature Sensors, documentation
- Tod E. Kurt, 2006, Spooky Projects, Introduction to Microcontrollers with Arduino, Class 4
- Tod E. Kurt, 2007, Bionic Arduino - Introduction to Microcontrollers with Arduino , Class 1,2,3,4
- Walrand Jean, 1997, Δίκτυα Επικοινωνιών, (μετάφραση Μ. Αναγνώστου) εκδόσεις Παπασωτηρίου
- WAMP, http://www.wampserver.com/en/
- Warren J.D. / Adams J. / Molle H. , 2011, Apress - Arduino Robotics
- Williams Sam, 2002, O'Reilly, Free as in Freedom: Richard Stallman's Crusade for Free Software, http://oreilly.com/openbook/freedom/
- www.ArduinoFun.com, Build Your Own Arduino
- Δολιανίτης Σπυρίδων, 2007, ΤΕΙ Κρήτης-Τμ.Ηλεκτρονικής, Προγραμματισμός και εγκατάσταση δικτύου αυτοματισμών LonWorks σε θερμοκήπια
- Ζαφειρόπουλος Αναστάσιος, 2010, ΕΜΠ, Ημερίδα ΚΑΠΕ, Διαχείριση Δικτύων Αισθητήρων χαμηλής ενεργειακής κατανάλωσης
- Καλαϊτζάκη Κ. / Κουτρούλη Ε., 2010, "Ηλεκτρικές Μετρήσεις και Αισθητήρες"


NO Rev3
U1
NCP1117ST50T3G
IN OUT
+5V
GND
C2
100n
C4
100n
U5B
LMV358IDGKR
SCK
1K
RN2A
YELLOW
GND
+3V3
C3
1u
USBVCC
T1
FDN340P
POWER
RESET
+3V3
8x1F-H8.5
JP2
2x2 M
MISO2
SCK2
RESET2
3x2 M
500mA
C11
22p
C9
22p
16MHZ
XT2
XT1
XTAL2(PC0)
XTAL1
AVCC
VCC
GND
C5
100n
RESET
10K
RN1D
TS4203-160R
TS4203-1-160R
XTAL2
R2
1M
16MHZ
AREF
AVCC
AGND
VCC
10x1F-H8.5
SCL
SDA
AREF
SCK
MISO
(INT4/ICP1/CLK0)
(OC1A/PCINT8)PC6
(PCINT9/OC1B)PC5
(PCINT10)PC4
(AIN2/PCINT11)PC2
(CTS/HWB/AIN6/TO/INT7)PD7
(RTS/AIN5/INT6)PD6
(XCK/AIN4/PCINT12)PD5
(INT5/AIN3)PD4
(TXD1/INT3)PD3
(INT0)PD0
TXLED
RXLED
M8RXD
M8TXD
TX
YELLOW
RX
YELLOW
RN2B
1K
(SS)
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0
(AIN1)PD7
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IOL
RN3B 22R
USHIELD
BLM21
L1
100n
C7
C8
1u
GND


+5V
VIN
RN1A 10K
RN1B 10K
CMP
+3V3
LMV358I
GATE_CMP
U2
IN
OUT
ON/OFF
GND NC/FB
LP2985-33DBVR
X1
PWRIN
D1
M7
PC1
47u
GND
GREEN
ON
RN4D 1K
RN4C 1K
MOSI2
D3
U3
(PCINT7/OC0A/OC1C)PB7
(PCINT6)PB6
(PCINT5)PB5
(T1/PCINT4)PB4
(PD0/MISO/PCINT3)PB3
(PDI/MOSI/PCINT2)PB2
(SCLK/PCINT1)PB1
(SS/PCINT0)PB0
MISO2
MOSI2
SCK2
USB boot En
CSTCE16M
RESET
ICSP
3x2 M
SS
IOS
IOB
IOH
6x1F-H8.5
AD
8x1F-H8.5
USB-B_TH
F1
MF-MSMF050
XUSB
220 RN3A
220 RN3D
RD-
RD+
TP_UCAP
USBVCC
UCAP
UVCC
D-
D+
UGND
PAD
ATMEGA16U2-MUR
C6
100n
1K RN2C
ATMEGA328P-PU
M8RXD
1K RN4B
1K RN4A