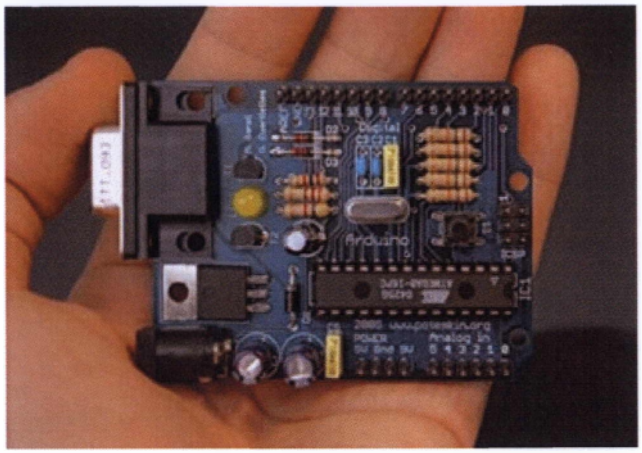
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΑΛΑΜΑΤΑΣ
ΠΑΡΑΡΤΗΜΑ ΣΠΑΡΤΗΣ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΜΕΤΡΗΣΗΣ ΚΑΙ ΚΑΤΑΓΡΑΦΗΣ ΤΗΣ ΕΝΕΡΓΟΥ (RMS) ΤΙΜΗΣ ΗΛΕΚΤΡΙΚΗΣ ΤΑΣΗΣ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

ΕΠΙΒΛΕΠΩΝ: ΛΙΑΠΕΡΔΟΣ ΙΩΑΝΝΗΣ

ΣΠΟΥΔΑΣΤΗΣ: ΤΣΙΡΙΓΩΤΗΣ ΦΩΤΙΟΣ 2005110

ΣΠΑΡΤΗ 2010

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Σκοπός της παρούσας εργασίας υπήρξε η σχεδίαση και η υλοποίηση ενός συστήματος μέτρησης και καταγραφής της ενεργού (RMS) τιμής της ηλεκτρικής τάσης σε πραγματικό χρόνο.

Το κείμενο αποτελείται από δύο βασικά μέρη Το Θεωρητικό και το Πρακτικό. Το πρώτο κρίθηκε απαραίτητο, χωρίς να επεκταθεί σε βάρος του δεύτερου, προκειμένου να δώσει στον μη εξοικειωμένο αναγνώστη τις κύριες θεωρητικές γνώσεις ώστε να γίνει κατανοητό το πώς δουλεύει αυτό το σύστημα.

Στο Πρακτικό μέρος παρουσιάζονται εκτενώς η μεθοδολογία σχεδίασης και κατασκευής.

Θα ήθελα και από τη θέση αυτή να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Ιωάννη Λιαπέρδο για την καθοδήγηση και την πολύτιμη βοήθειά του κατά τη διάρκεια εκπόνησης της εργασίας, καθώς και για τη διάθεση των υποδομών του Εργαστηρίου Ηλεκτρονικής χωρίς τα οποία θα ήταν αδύνατη η πραγματοποίησή της. Επίσης, ευχαριστώ την οικογένειά μου για την πολύπλευρη συμπαράστασή της όχι μόνο κατά τη διάρκεια της εκπόνησης της εργασίας, αλλά για ολόκληρο το διάστημα της φοίτησής μου.

Σπάρτη, του 2011

1. ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

1.1 Ορισμός RMS τάσης

Η τιμή RMS ενός συνόλου δειγμάτων (ή μιας κυματομορφής συνεχούς χρόνου) είναι η τετραγωνική ρίζα του αριθμητικού μέσου όρου (μέσος όρος) των τετραγώνων των τιμών των δειγμάτων (ή το τετράγωνο της συνάρτησης που περιγράφει τη συνεχή κυματομορφή).
Στην περίπτωση ενός συνόλου δειγμάτων η $\{x_1, x_2, \ldots, x_n\}$ η τιμή RMS είναι:

$$x_{\text{rms}} = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \cdots + {x_n}^2}{n}}.$$

1.2 Υπολογισμός RMS τάσης

Ο αντίστοιχος τύπος για μια συνεχή συνάρτηση (ή κυματομορφή) f(t) που ορίζεται στο χρονικό διάστημα $T_1 \leq t \leq T_2$ είναι

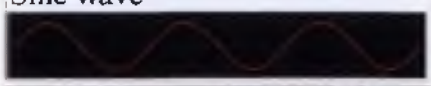
$$f_{\text{rms}} = \sqrt{\frac{1}{T_2 - T_1}\int_{T_1}^{T_2} [f(t)]^2\,dt}.$$

και η αντίστοιχη έκφραση για ολόκληρο το πεδίο του χρόνου είναι

$$f_{\text{rms}} = \lim_{T\to\infty} \sqrt{\frac{1}{2T}\int_{-T}^{T} [f(t)]^2\,dt}.$$

Για μια περιοδική συνάρτηση είναι προφανές ότι οι δύο εκφράσεις ταυτίζονται. Η τιμή RMS ενός συνεχούς σήματος μπορεί να προσεγγισθεί με τη λήψη των RMS μιας σειράς από ισαπέχοντα δείγματα. Στην περίπτωση του στατιστικού RMS μιας τυχαίας διαδικασίας, η αναμενόμενη τιμή χρησιμοποιείται αντί της μέσης τιμής.

Παρακάτω ακολουθεί ένας πίνακας με τις RMS τιμές γνωστών περιοδικών κυματομορφών

Waveform	RMS
Sine wave 	$\frac{a}{\sqrt{2}}$
Square wave 	a

Sawtooth wave	$\frac{a}{\sqrt{3}}$
	

1.3 Λόγοι μέτρησης της RMS τάσης

Ο λόγος ύπαρξης του μεγέθους RMS είναι ο εξής Πολύ συχνά, η μέση τιμή μιας κυματομορφής δεν μας λέει και πολλά πράγματα. Για παράδειγμα, η μέση τιμή μιας ημιτονοειδούς κυματομορφής είναι μηδέν, ασχέτως πλάτους. Αυτό προκαλεί προβλήματα σε πολλούς υπολογισμούς. Για παράδειγμα, η ηλεκτρική τάση από ένα ρευματοδότη είναι ημιτονοειδής. Άρα έχει μηδενική μέση τιμή. Αντικαθιστώντας την τιμή αυτή στις διάφορες εξισώσεις, θα πάρουμε ως αποτέλεσμα ότι η ισχύς που καταναλώνει κάθε ηλεκτρική συσκευή είναι επίσης μηδενική, πράγμα που προφανώς δεν συμβαίνει. Καθίσταται έτσι αναγκαία η δημιουργία ενός νέου μέτρου, του RMS.

1.4 Θεωρίες για κατανόηση του AC

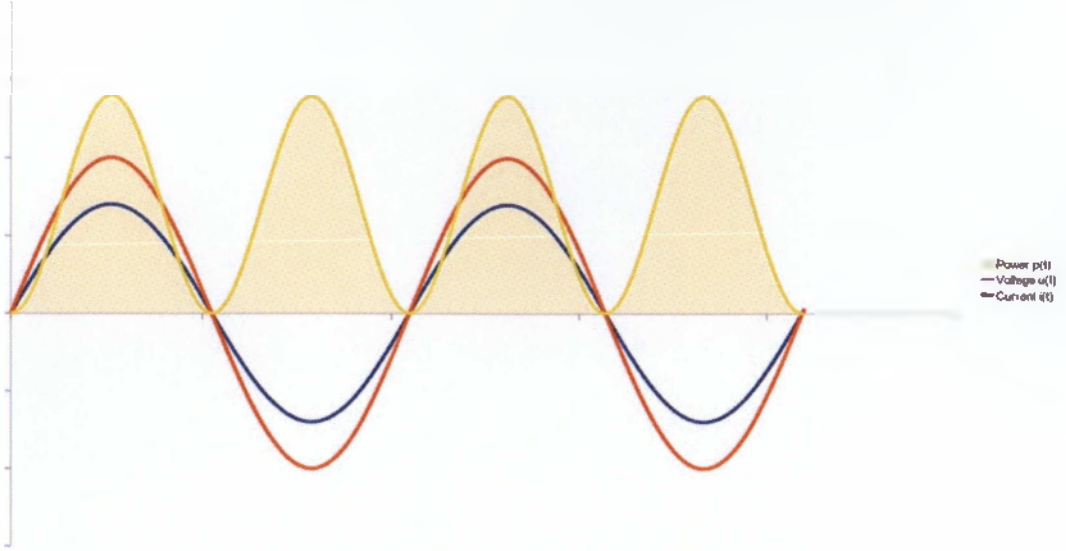
Βασικές Έννοιες

Κατανόηση εναλλασσόμενου ρεύματος

Μια συσκευή παρακολούθησης της κατανάλωσης ενέργειας στο σπίτι μετρά την ενέργεια που χρησιμοποιείται από τις συσκευές που συνδέονται στο οικιακό δίκτυο ηλεκτροδότησης. Για να καταλάβει κανείς πώς γίνεται αυτό, είναι χρήσιμο να γνωρίζουμε πώς οι συσκευές αλληλεπιδρούν με το ρεύμα. Δεν αλληλεπιδρούν όλες οι συσκευές με το ρεύμα με τον ίδιο τρόπο.

Ωμικά φορτία

Λαμπτήρες πυρακτώσεως, βραστήρες, σίδερα, ηλεκτρικοί θερμοσίφωνες, ηλεκτρικές κουζίνες χρησιμοποιούν όλη την ενέργεια που τους παρέχεται. Πρόκειται για ωμικά φορτία που σημαίνει ότι ισχύει ο Νόμος του Ohm, δηλαδή I = V / R (ρεύμα = τάση / αντίσταση). Έτσι η τάση και το ρεύμα κυματομορφής εξόδου είναι παρόμοιο με το ακόλουθο:

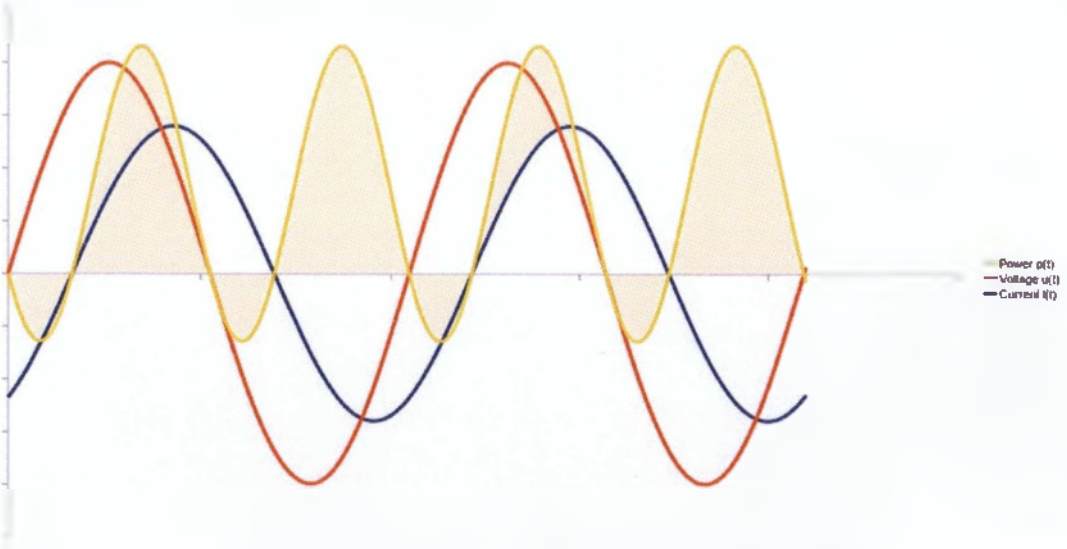


Σχήμα 1.4.1 Κυματομορφή ωμικών φορτίων

Η κίτρινη γραμμή είναι η ισχύς σε μια δεδομένη στιγμή (σε οποιαδήποτε δεδομένη στιγμή ονομάζεται στιγμιαία ισχύς), η οποία είναι ίση με το γινόμενο της τάσης και του ρεύματος σε μια δεδομένη στιγμή. Παρατηρούμε πώς η ισχύς είναι πάντα θετική σε αυτή την περίπτωση. Η θετική κατεύθυνση είναι η ενέργεια που ρέει στο φορτίο.

Εν μέρει άεργα φορτία

Ωστόσο, συσκευές όπως ψυγεία και πλυντήρια, χρησιμοποιούν μέρος από την ενέργεια που τους παρέχεται. Αυτά έχουν επαγωγικά ή χωρητικά δομικά στοιχεία εκτός από την αντίσταση. Ένα μερικώς επαγωγικό φορτίο δίνει τάση και ρεύμα κυματομορφής εξόδου παρόμοιο με το ακόλουθο:



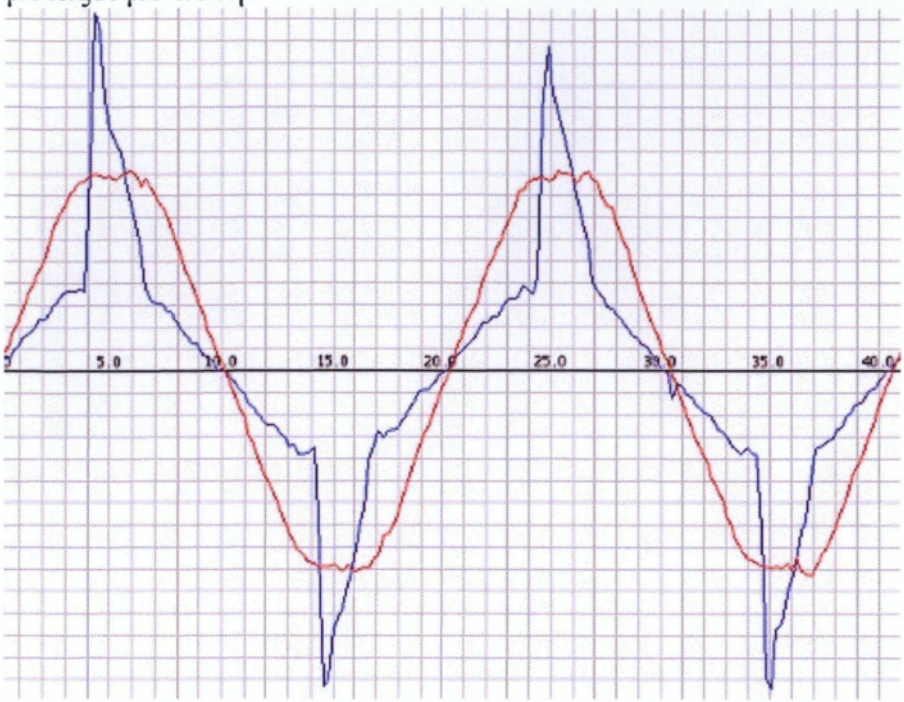
Σχήμα 1.4.2 Κυματομορφή μη ωμικών φορτίων

Παρατηρούμε πώς η κίτρινη γραμμή είναι αρνητική για ένα χρονικό διάστημα. Το θετικό κομμάτι είναι η ενέργεια που ρέει στο φορτίο και το αρνητικό κομμάτι είναι η ενέργεια που ρέει στην πηγή.
Το άλλο πράγμα που πρέπει να σκεφτούμε είναι ότι οι κυματομορφές της τάσης και του ρεύματος έχουν μετατοπιστεί. Είναι σαν να φορτίζουμε ένα αρκετά μεγάλο πυκνωτή με μια αντίσταση σε σειρά. Η τάση της πηγής αυξάνεται και μιας και είναι υψηλότερη από την τάση του πυκνωτή, το ρεύμα εισέρχεται στον πυκνωτή (η θετική κατεύθυνση στο γράφημα) και έτσι αυξάνεται η τάση του πυκνωτή. Όταν η τάση της πηγής πέφτει, η τάση στα άκρα του πυκνωτή είναι μεγαλύτερη από την τάση της πηγής και το ρεύμα αρχίζει να ρέει προς την κατεύθυνση της πηγής (αρνητική κατεύθυνση στο γράφημα). Αυτό προκαλεί στην κυματομορφή ρεύματος να εμφανίζεται σαν να έχει μετακινηθεί, όπως στο γράφημα (Η αλλαγή είναι γνωστή ως μετατόπιση φάσης).

Πραγματική ισχύς, άεργος ισχύς και φαινομενική ισχύς
Κοιτάζοντας τις δύο γραφικές παραστάσεις σε συχνότητα ρεύματος η κυματομορφή της ισχύος κυμαίνεται 50 με 60 φορές το δευτερόλεπτο.
Πραγματική ισχύς (Real power) συχνά ορίζεται ως η ενέργεια που χρησιμοποιείται από μια συσκευή για να παράγει χρήσιμο έργο. Κοιτάζοντας το διάγραμμα πιο πάνω τα θετικά τμήματα είναι η ισχύς του φορτίου και τα αρνητικά τμήματα είναι η ισχύς της πηγής. Έτσι η ισχύς του φορτίου μείον την ισχύ της πηγής είναι η πραγματική ισχύς.
Άεργος ισχύς (Reactive power) είναι ένας τρόπος ελέγχου της ισχύος που επιστρέφει στην πηγή.
Άλλο ένα χρήσιμο μέτρο της ισχύος είναι η φαινομενική ισχύς (Apparent Power) που είναι το γινόμενο της RMS τάσης και του RMS ρεύματος. Για τα ωμικά φορτία η πραγματική ισχύς είναι ίση με τη φαινομενική ισχύ. Για όλα τα άλλα φορτία η πραγματική ισχύς είναι μικρότερη από τη φαινομενική ισχύ. Η φαινομενική ισχύς είναι ένα μέτρο της πραγματικής και της αέργου ισχύος, αλλά δεν είναι άθροισμα των δύο, αφού ως άθροισμα των δύο δεν λαμβάνει υπόψη τις διαφορές της φάσης.
Σχέση μεταξύ πραγματικής, αέργου και φαινομενικής ισχύος για ημιτονοειδή φορτία:
Real Power = Apparent Power x cos(theta)
Reactive Power = Apparent Power x sin(theta)
Το cos(theta) είναι επίσης γνωστό ως συντελεστής ισχύος (power factor).

Μια σημείωση για μη-γραμμικά φορτία
Ο συντελεστής ισχύος ισχύει μόνο για γραμμικά ημιτονοειδή φορτία. Τα περισσότερα τροφοδοτικά για DC συσκευές, όπως οι υπολογιστές,

ασκούν ένα μη γραμμικό φορτίο και η κυματομορφή του ρεύματος μοιάζει με αυτή:



Σχήμα 1.4.3 Κυματομορφή μη γραμμικών φορτίων

Μπορούμε να υπολογίσουμε ακόμα το συντελεστή ισχύος από την ακόλουθη εξίσωση:
Power Factor = Real Power / Apparent Power
αλλά ο συντελεστής ισχύος cos(theta) δεν θα ήταν σωστός, αφού αρμονικές θα έπρεπε να προστεθούν.
Η τιμή του συντελεστή ισχύος μετράει πόσο επηρεάζεται η απόδοση του ρεύματος και από την υστέρηση φάσης φ και από το αρμονικό περιεχόμενο του ρεύματος εισόδου.

Για να συνοψίσουμε
Υπάρχουν πολλά πράγματα που μπορούμε να μετρήσουμε σχετικά με τη χρήση της ενέργειας σε συστήματα εναλλασσόμενου ρεύματος (AC). Το καθένα με τις χρήσεις του. Ωστόσο, για τη μέτρηση της ενέργειας των νοικοκυριών η πραγματική ισχύς είναι η πιο χρήσιμη τιμή, δεδομένου ότι μας λέει πόση ενέργεια χρησιμοποιούν όλες οι συσκευές μας.

Μαθηματική Ανάλυση
Η θεωρία αυτή καλύπτει τα μαθηματικά όσων μελετήσαμε στην πρώτη θεωρία.

Πραγματική ισχύς

Πραγματική ισχύς (επίσης γνωστή ως ενεργός ισχύς) ορίζεται ως η ενέργεια που χρησιμοποιείται από μια συσκευή για να παράγει χρήσιμο έργο.
Από μαθηματική άποψη, είναι το ορισμένο ολοκλήρωμα της τάσης, u(t), επί το ρεύμα, i(t), ως εξής:

$$P = \frac{1}{T}\int u(t) \times i(t)\, dt \equiv U \times I \times \cos(\varphi)$$

U - Root-Mean-Square (RMS) τάση
I - Root-Mean-Square (RMS) ρεύμα
cos(φ) - Συντελεστής ισχύος

Η πραγματική ισχύς υπολογίζεται ως ο μέσος όρος των N γινομένων τάσης – ρεύματος. Άρα η μέθοδος αυτή ισχύει για όλες τις κυματομορφές.

Για σήμα διακριτού χρόνου ισοδύναμα είναι:

$$P \equiv \frac{1}{N}\sum_{n=0}^{N-1} u(n) \times i(n)$$

u(n) – δείγμα του u(t)
i(n) – δείγμα του i(t)
N – αριθμός δειγμάτων

Μια τιμή RMS ορίζεται ως η τετραγωνική ρίζα της μέσης τιμής των τετραγώνων των στιγμιαίων τιμών μιας περιοδικά μεταβαλλόμενης ποσότητας, κατά μέσο όρο πάνω σε ένα πλήρη κύκλο. Η διακριτή εξίσωση του χρόνου για τον υπολογισμό της τάσης RMS έχει ως εξής:

$$U_{rms} = \sqrt{\frac{\sum_{n=0}^{N-1} u^2(n)}{N}}$$

1.5 Τι είναι το λογισμικό Arduino;

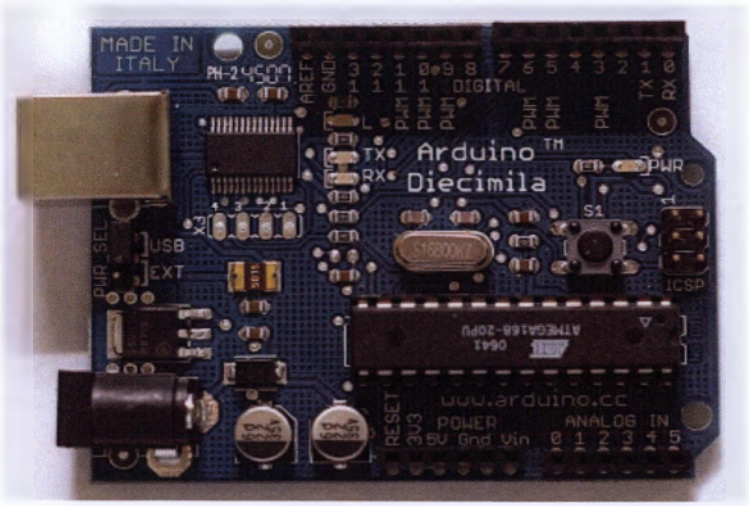
Το **Arduino** είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (πρόκειται για τη C++ με κάποιες

μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες. Επίσης, το διάγραμμα και πληροφορίες για το υλικό είναι διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.
Το Arduino είναι η καρδιά του συστήματος, αλλά και η γέφυρα μεταξύ του αναλογικού κόσμου των ηλεκτρονικών και του ψηφιακού κόσμου του λογισμικού. Το Arduino καθιστά δυνατή τη γραφική παράσταση της ενεργειακής κατανάλωσης σε έναν υπολογιστή, φορτώνει τις πληροφορίες στο διαδίκτυο και τις απεικονίζει στην οθόνη με ευκολία. Για την παρακολούθηση της ενέργειας το Arduino μπορεί να δέχεται τιμές από τους αισθητήρες και τα ηλεκτρονικά τους, να υπολογίζει τις τιμές για την πραγματική ισχύ, φαινομενική ισχύ, συντελεστή ισχύος, rms τάσεως και rms ένταση και να στέλνει αυτές τις τιμές στη θύρα USB για να τις επεξεργαστούμε στον υπολογιστή.
Η διεπαφή μεταξύ του αναλογικού και του ψηφιακού κόσμου, η ADC
Το Arduino μετατρέπει το αναλογικό σήμα εισόδου σε αριθμητικά στοιχεία με ένα ενσωματωμένο μετατροπέα αναλογικού σε ψηφιακό.
Το Arduino ADC έχει ανάλυση 10bits. Αυτό μαζί με το ποσοστό του δείγματος καθορίζει τις λεπτομέρειες της ψηφιακής αναπαράστασης του σήματος που έχουμε στην αναλογική είσοδο. Η αναλογική είσοδος του Arduino έχει ένα προεπιλεγμένο εύρος εισόδου από 0 έως 5V. 10 bits σημαίνει ότι μπορούμε να διαιρέσουμε αυτό το εύρος σε 2 ^ 10 ή 1024 τμήματα. Το ADC έχει κατά συνέπεια 5V / 1024 = 4.8mV ευαισθησία (Μπορεί να ανιχνεύσει αλλαγές στην τάση εισόδου 4.8mV). Μια αναλογική είσοδος διαβάζεται από την εντολή: sample(=δείγμα) = analogRead (pinNumber).

Το sample(=δείγμα) είναι ακέραιος. Εάν η τάση στο pin είναι 0 sample(=δείγμα) = 0. Αν η τάση είναι 5V, δείγμα = 1024. Αν η τάση είναι ίση με 2.5V, δείγμα = 512 και ούτω καθεξής.

Υλικό
Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωσή του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό περιοριστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό

αντηχητή σε κάποιες παραλλαγές). Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα bootloader, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής.
Γενικά όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων RS-232 και TTL. Οι πλακέτες Arduino που κυκλοφορούν σήμερα στην αγορά, συμπεριλαμβανόμενης και της Diecimila, προγραμματίζονται μέσω USB, εφαρμόζοντας ένα τσίπ προσαρμογέα USB-to-serial όπως το FTDI FT232. Κάποιες παραλλαγές, όπως το Arduino mini και το ανεπίσημο Boarduino, χρησιμοποιούν προσαρμογέα USB-to-serial σε μορφή πλακέτας ή καλωδίου.
Η πλακέτα του Arduino έχει εκτεθειμένες τις περισσότερες επαφές εισόδου/εξόδου για χρήση με άλλα κυκλώματα. Το Diecimila (εικόνα 1.4.1), για παράδειγμα, παρέχει 14 ψηφιακές επαφές εισόδου/εξόδου, από τις οποίες οι 6 μπορούν να παράξουν σήματα PWM, και 6 αναλογικές εισόδους. Αυτές οι επαφές είναι διαθέσιμες στην κορυφή της πλακέτας μέσω θηλυκών συνδέσεων μεγέθους 0,1 ιντσών. Διάφορες plug-in πλακέτες εφαρμογών γνωστές σαν "shields" είναι, επίσης, διαθέσιμες στο εμπόριο.

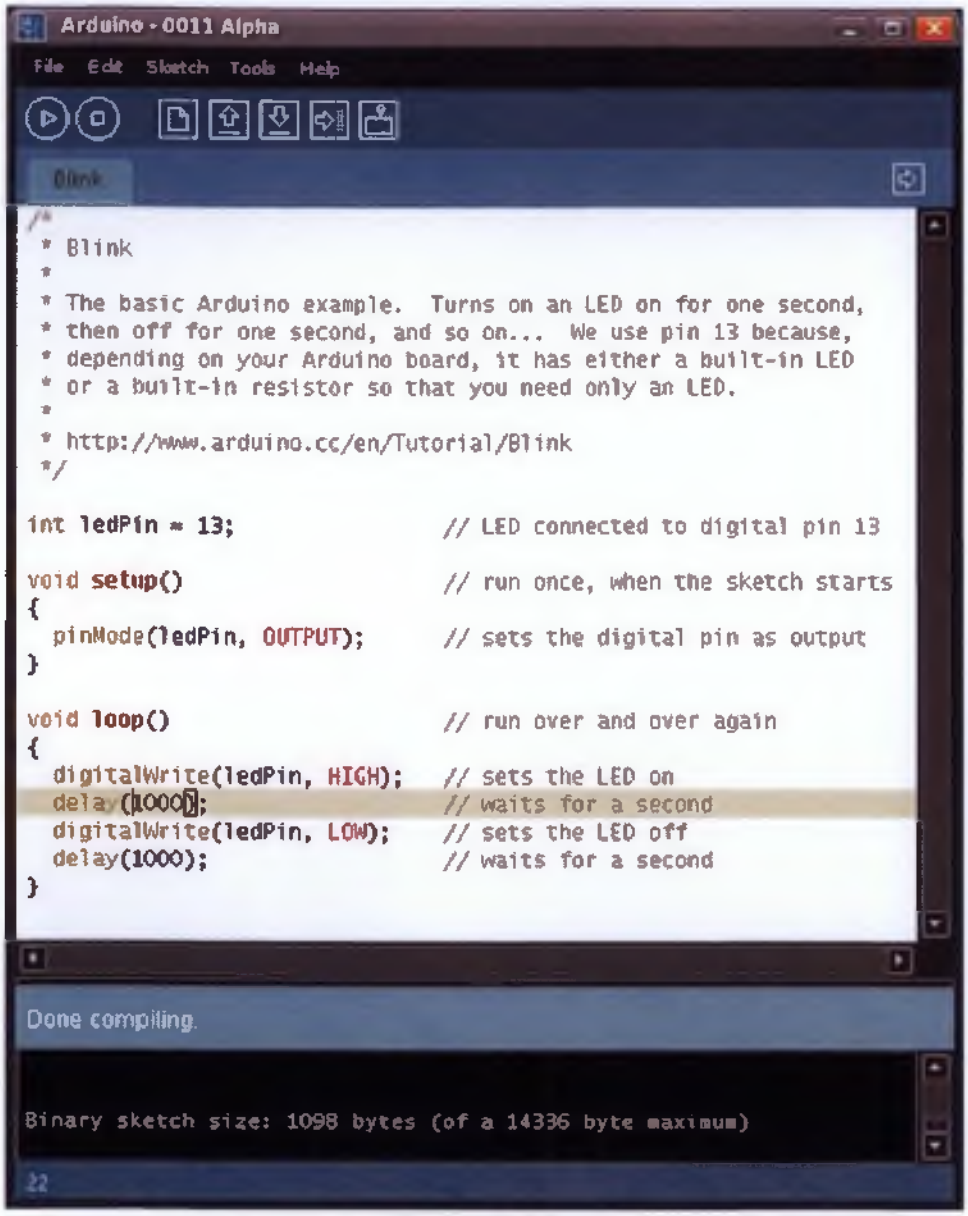


Εικόνα 1.5.1

Οι συμβατές με το Arduino πλακέτες Barebones και Boarduino διαθέτουν αρσενικές επαφές στην κάτω πλευρά της πλακέτας για να μπορούν να συνδεθούν με πλακέτες που δεν χρειάζονται συγκολλήσεις.

Λογισμικό
Το IDE του Arduino (Σχήμα 1.5.2) είναι γραμμένο σε Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα και

μεταγλωττιστή και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα.



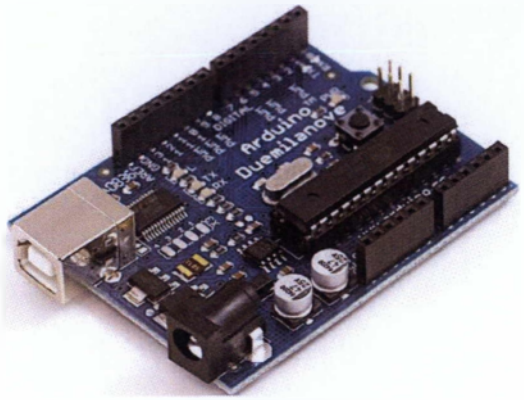
Εικόνα 1.5.2

1.6 Arduino Duemilanove

1.6.1 Γενικές Πληροφορίες

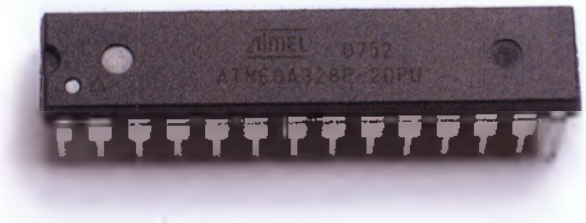
Το Arduino Duemilanove είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω

προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Περιέχει όλα αυτά που απαιτούνται για να υποστηρίξουν τον μικροελεγκτή. Συνδέεται απλά με έναν υπολογιστή με ένα καλώδιο USB και τροφοδοτείται με συνεχές ρεύμα ή με μπαταρία.



Εικόνα 1.6.1

Οι υπολογιστικές λειτουργίες εκτελούνται στον μικροελεγκτή ATmega328 που διαθέτει 16KB μνήμης flash για την αποθήκευση κώδικα. Έχει επίσης 1 KB SRAM και 512 bytes μνήμης EEPROM (τα οποία μπορούν να διαβαστούν και να γραφτούν με την βιβλιοθήκη EEPROM).

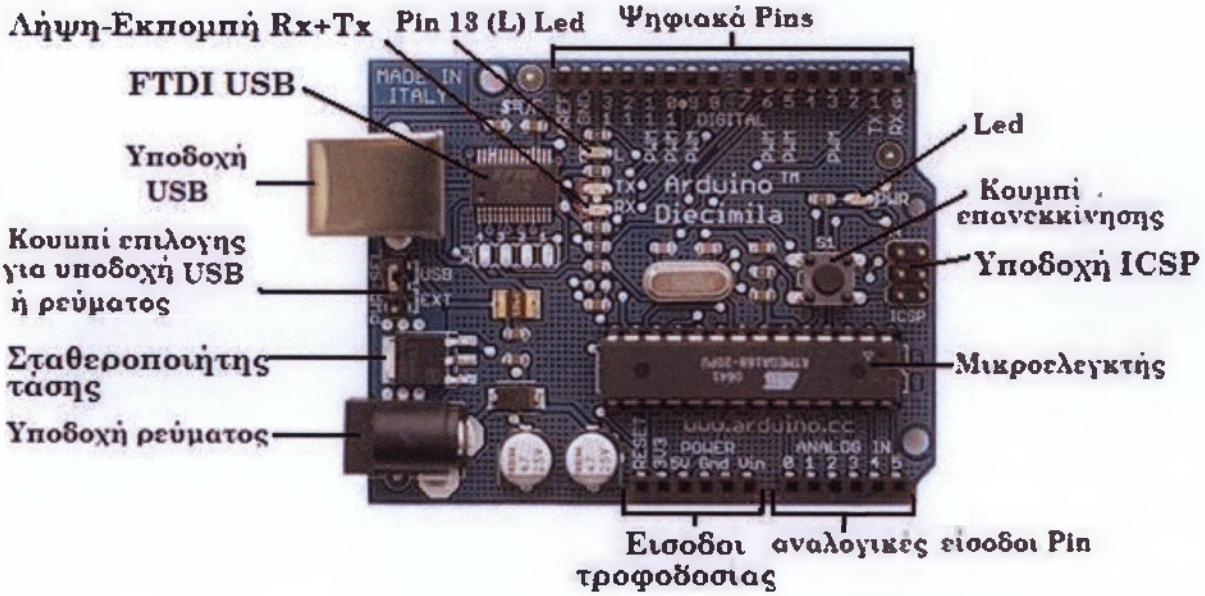


Εικόνα 1.6.2

Το Arduino Duemilanove τροφοδοτείται είτε από εξωτερική τροφοδοσία είτε απευθείας από την θύρα USB. Η επιλογή της πηγής γίνεται αυτόματα. Ως εξωτερική τροφοδοσία ορίζεται είτε μια μπαταρία, είτε μετασχηματιστής των 9Volt από 220V. Η μπαταρία μπορεί να συνδεθεί στις υποδοχές του Arduino Vin και GND όπου τοποθετούνται ο θετικός πόλος και ο αρνητικός αντίστοιχα. Από την άλλη αν τροφοδοτήσουμε με μετασχηματιστή απλά τοποθετούμε το βύσμα στην υποδοχή που υπάρχει με τον θετικό πόλο στο κέντρο.
Η πλακέτα μπορεί να λειτουργήσει με εξωτερική πηγή από 6 έως 20 Volt. Αν ωστόσο τροφοδοτηθεί με λιγότερα από 7 Volt τα pin εξόδου δεν θα καταφέρουν να εξάγουν τάση 5 Volt. Αν από την άλλη δώσουμε πάνω από 12 Volt θα υπερθερμανθεί ο σταθεροποιητής τάσης στην πλακέτα

και υπάρχει μεγάλη πιθανότητα να καταστραφεί. Συνεπώς μια ιδανική τάση είναι τα 9 Volt.
Το Arduino Duemilanove έχει την δυνατότητα να επικοινωνεί με τον Ηλεκτρονικό Υπολογιστή, με ένα άλλο Arduino ή άλλους μικροελεγκτές. Το ολοκληρωμένο ATMega 328 παρέχει σειριακή επικοινωνία TTL 5 Volt UART, η οποία είναι διαθέσιμη από τους ακροδέκτες (λήψη RX) 0 και (εκπομπή TX) 1 του ολοκληρωμένου. Επιπλέον, στην πλακέτα του Arduino είναι ενσωματωμένο ένα ολοκληρωμένο FTDI FT232RL το οποίο παρέχει σειριακή επικοινωνία με τον Ηλεκτρονικό Υπολογιστή για προγραμματισμό, πάνω από την θύρα USB με την βοήθεια των ανάλογων FTDI drivers. Οι drivers αυτοί περιλαμβάνονται στο software για το Arduino και παρέχουν μια ιδεατή θύρα επικοινωνίας στον Ηλεκτρονικό Υπολογιστή για τους σκοπούς της επικοινωνίας.

1.6.2 Τα μέρη του συστήματος



Εικόνα 1.6.3

Λειτουργίες Ακροδεκτών:

- Σειριακή Λειτουργία: 0 (RX) and 1 (TX). Χρησιμοποιούνται για λήψη (RX) και εκπομπή (TX) TTL σειριακών δεδομένων. Αυτοί οι ακροδέκτες είναι συνδεδεμένοι με τους αντίστοιχους του ολοκληρωμένου FTDI USB-to-TTL Serial.
- PWM: 3, 5, 6, 9, 10, and 11. Παρέχουν έξοδο 8-bit PWM με την συνάρτηση analogWrite().
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Αυτοί οι ακροδέκτες επιτρέπουν επικοινωνία SPI, η οποία αν και παρέχεται από το

hardware δεν είναι ακόμα διαθέσιμη στην γλώσσα προγραμματισμού του Arduino.

- LED: 13. Στον ακροδέκτη 13 υπάρχει ένα ενσωματωμένο LED. Όταν ο ακροδέκτης έχει τιμή HIGH, το LED φωτοβολεί.
- I2C: 4 (SDA), 5 (SCL). Υποστηρίζει το πρωτόκολλο I2C (TWI) χρησιμοποιώντας βιβλιοθήκες τις Γλώσσας προγραμματισμού Wiring.
- AREF.είναι ένα αναλογικό pin που χρησιμοποιείτε για μετατροπή αναλογικού σε ψηφιακό. Χρησιμοποιείται με την συνάρτηση analogReference().
- Reset. Αν τεθεί σε κατάσταση LOW (χαμηλή) τότε μπορούμε να κάνουμε επανεκκίνηση τον μικροελεγκτή, χρησιμοποιώντας το κουμπί επανεκκίνησης όπως φαίνεται παρακάτω.

1.6.3 Χρήσεις του Arduino Duemilanove

Το Arduino είναι μια πλακέτα ανάπτυξης πρωτοτύπων ανοιχτού υλικού και λογισμικού. Ενσωματώνει έναν μικροελεγκτή και συνδέεται με τον Η/Υ ώστε να προγραμματίζεται μέσα από ένα απλό περιβάλλον ανάπτυξης. Ένα Arduino μπορεί να χρησιμοποιηθεί για να αναπτύξουμε εφαρμογές ελέγχου (control) όπως να ελέγχουμε διάφορα φώτα, κινητήρες και άλλες συσκευές εξόδου του φυσικού κόσμου.

Τα Projects στον εν λόγω μικροελεγκτή μπορούν να είναι αυτόνομα (σε επίπεδο hardware) ή να επικοινωνούν με κάποιο software στον Η/Υ του προγραμματιστή (προγράμματα όπως τα Flash, Processing, MaxMSP).

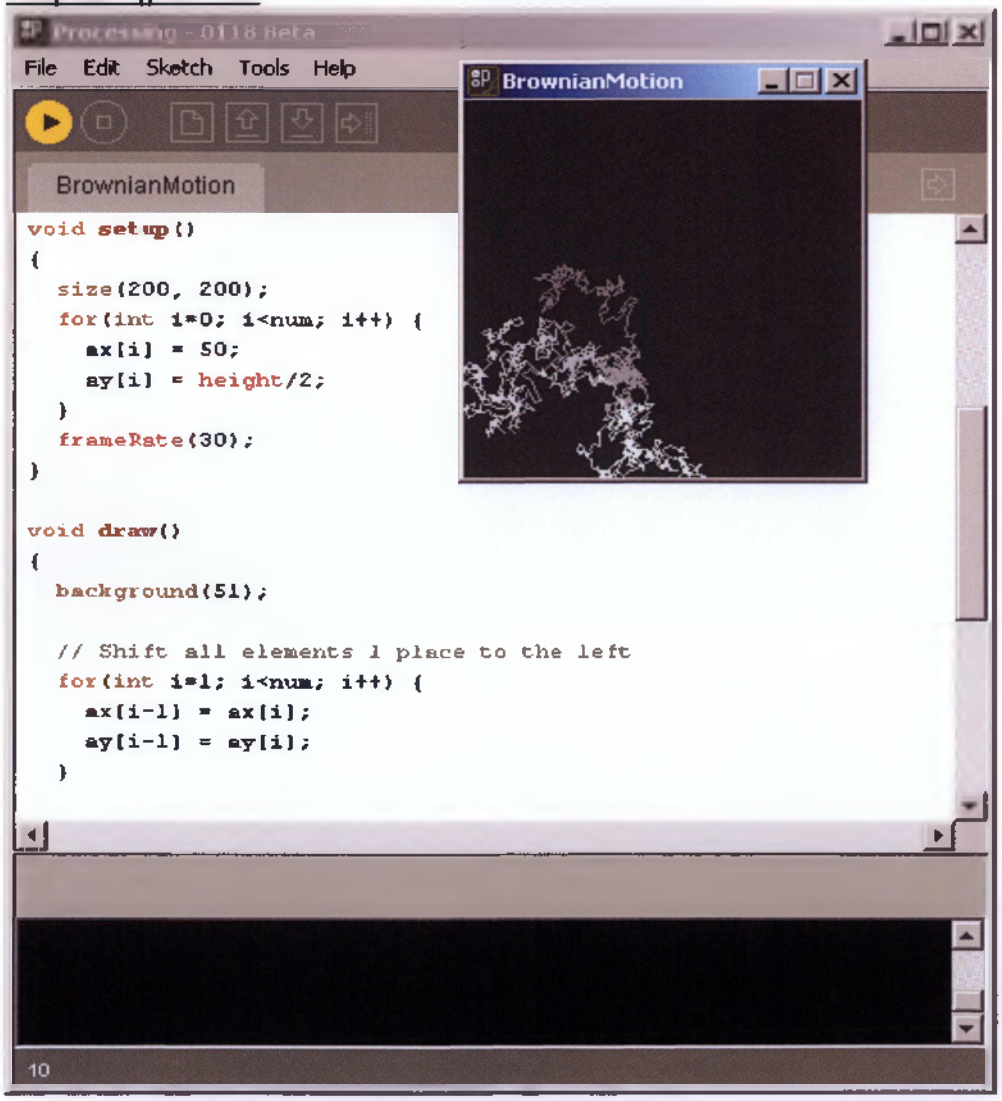
Οι πλακέτες μπορούν εύκολα να συναρμολογηθούν ακόμη και από έναν αρχάριο ή να αγοραστούν μονταρισμένες. Το περιβάλλον ανάπτυξης του λογισμικού βασίζεται στην γλώσσα προγραμματισμού Processing και την γλώσσα προγραμματισμού Wiring, οι οποίες είναι ανοιχτού κώδικα (open source) και μπορεί κάποιος να τις «κατεβάσει» δωρεάν. Η γλώσσα προγραμματισμού του Arduino εξομοιώνει απόλυτα το φυσικό περιβάλλον του μικροελεγκτή και είναι βασισμένη σε C/C++.

1.7 Τι είναι το Processing;

Το Processing είναι μια ανοικτή γλώσσα προγραμματισμού και ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που κατασκευάζονται για τα ηλεκτρονικά εργαστήρια και τις κοινότητες οπτικού σχεδιασμού με

σκοπό τη διδασκαλία των βασικών του προγραμματισμού για ηλεκτρονικούς υπολογιστές σε ένα οπτικό πλαίσιο και να χρησιμεύσει ως θεμέλιο για ηλεκτρονικά sketchbooks. Ξεκίνησε το 2001 από τους Casey Reas και Ben Fry, οι οποίοι ήταν πρώτα στο MIT Media Lab. Ένας από τους στόχους του Processing είναι να λειτουργήσει ως ένα εργαλείο για να κάνει μη-προγραμματιστές να ασχοληθούν με τον προγραμματισμό, μέσω της οπτικής επικοινωνίας. Η γλώσσα βασίζεται στις γραφικές δυνατότητες της γλώσσας προγραμματισμού Java, απλοποιώντας τα χαρακτηριστικά και δημιουργώντας μερικά νέα.

Χαρακτηριστικά



Εικόνα 1.7.1

Το Processing περιλαμβάνει ένα sketchbook, μια εναλλακτική λύση του IDE για την οργάνωση projects.

Κάθε σκίτσο του Processing είναι στην πραγματικότητα μια υποκατηγορία της PApplet Java κλάσης που υλοποιεί τα περισσότερα από τα χαρακτηριστικά του Processing.
Κατά τον προγραμματισμό στο Processing όλες οι επιπλέον κλάσεις που καθορίζονται θα πρέπει να αντιμετωπίζονται ως εσωτερικές κλάσεις όταν ο κώδικας μεταφράζεται σε Java πριν από τη σύνταξη (compile). Αυτό σημαίνει ότι η χρήση στατικών μεταβλητών και των μεθόδων των κλάσεων απαγορεύεται εκτός αν θέσετε ρητά στο Processing ότι θέλετε κώδικα σε Java.

Σχετικά Projects

- Σχεδιασμός με αριθμούς

Το Processing βασίστηκε στο πρωτότυπο έργο που επιτελέστηκε για το project Design By Numbers στο MIT. Μοιράζεται πολλές από τις ίδιες ιδέες και είναι τέκνο του ίδιου πειράματος.

- Wiring, Arduino και Fritzing

Το Processing έχει γεννήσει ένα άλλο project, το Wiring, το οποίο χρησιμοποιεί τον επεξεργαστή IDE μαζί με μια απλοποιημένη εκδοχή της C+ + ως ένα τρόπο για να διδάξουν καλλιτέχνες πώς να προγραμματίζουν μι-κροελεγκτές. Υπάρχουν πλέον δύο ξεχωριστά hardware projects, Wiring και Arduino, χρησιμοποιώντας το περιβάλλον και τη γλώσσα του Wiring. Το Fritzing είναι ένα άλλο περιβάλλον λογισμικού του ίδιου είδους, το οποίο βοηθά τους σχεδιαστές και καλλιτέχνες να παρουσιάσουν τα διαδραστικά πρωτότυπά τους και να οδηγηθούμε στο πραγματικό προϊόν.

- Processing Monsters

Το Processing Monsters είναι ένα project από το Lukas Vojir με στόχο να βοηθήσει τους ανθρώπους να μάθουν τη γλώσσα με διασκεδαστικό τρόπο. Τα «τέρατα» είναι απλά γραφικά προγράμματα που είναι άσπρο – μαύρο και mouse reactive. Από την 3η Μαΐου 2010, υπάρχουν 70 «τέρατα» που εμφανίζονται στην ιστοσελίδα του Vojir.

2. ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

2.1 Εξοπλισμός

Ο εργαστηριακός εξοπλισμός που χρησιμοποιήθηκε ήταν ο ακόλουθος:
1. Πλακέτα Arduino
2. USB Καλώδιο
3. Ηλεκτρονικός Υπολογιστής
4. Λογισμικό Arduino
5. Λογισμικό Processing
6. Τροφοδοτικό Συνεχούς Τάσης
7. Γεννήτρια Συναρτήσεων

2.2 Κώδικας

Ο τύπος $x_{\text{rms}} = \sqrt{\frac{x_1^2 - x_2^2 + \cdots + x_n^2}{n}}$ γραμμένος σε κώδικα Arduino έχει ως εξής

Για το Arduino

```
int analogPin = 1;
double squared_voltage, voltage, sum, mean, rms;
int rms_integer;

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

void loop() {
    int x = 0;
    sum = 0;
    for (x = 1; x <= 10; x++)
    {
    voltage = analogRead(analogPin);
    squared_voltage = sq(voltage);
    sum = sum + squared_voltage;
    delay(100);
    }
  mean = sum / 10;
  rms = sqrt(mean);
  rms_integer=int(rms);
  rms_integer=map(rms_integer,0,1023,0,255);
```

```
23.   Serial.print(rms_integer,BYTE);
24. }
```

Επεξήγηση κώδικα: Γραμμές 1 – 3 Δήλωση μεταβλητών
Γραμμές 4 – 24 Υπολογισμός μεταβλητών

Για τη γραφική απεικόνιση των μετρήσεων στην οθόνη του υπολογιστή ο αντίστοιχος κώδικας στο Processing είναι ο εξής

Για το Processing

```
1. import processing.serial.*;
2.
3. Serial myPort;        // The serial port
4. int xPos = 41;        // horizontal position of the graph
5. int xPos_last =41;
6. float real_value_last=0;
7. float inByte_float;
8. PFont font;
9. int count_seconds=1;
10.
11. void setup () {
12.  // set the window size:
13.  size(400+40, 256+80);
14.  // set initial background:
15.  background(#DDDDDD);
16.  font = loadFont("Verdana-10.vlw");
17.  textFont(font);
18.  fill(0, 102, 153);
19.  line(40,296,400,296);
20.  for (int i = 0; i <= 256; i = i+12) {
21.  line(40, 296-i, 400, 296-i);
22.  }
23.  for (int i = 40; i < 400; i = i+20) {
24.  line(440-i, 40, 440-i, 296);
25.  }
26.  line(40,296,40,40);
27.  text("Vrms (Volts)", 10,30);
28.  for (int i = 0; i <= 256; i = i+20) {
29.  text(i, 20,301-i);
30.  }
31.  text("Time (seconds)", width/2-40,height-10);
32.  for (int i = 40; i <= 400; i = i+40) {
33.  text(+i-40, i-5,310);
```

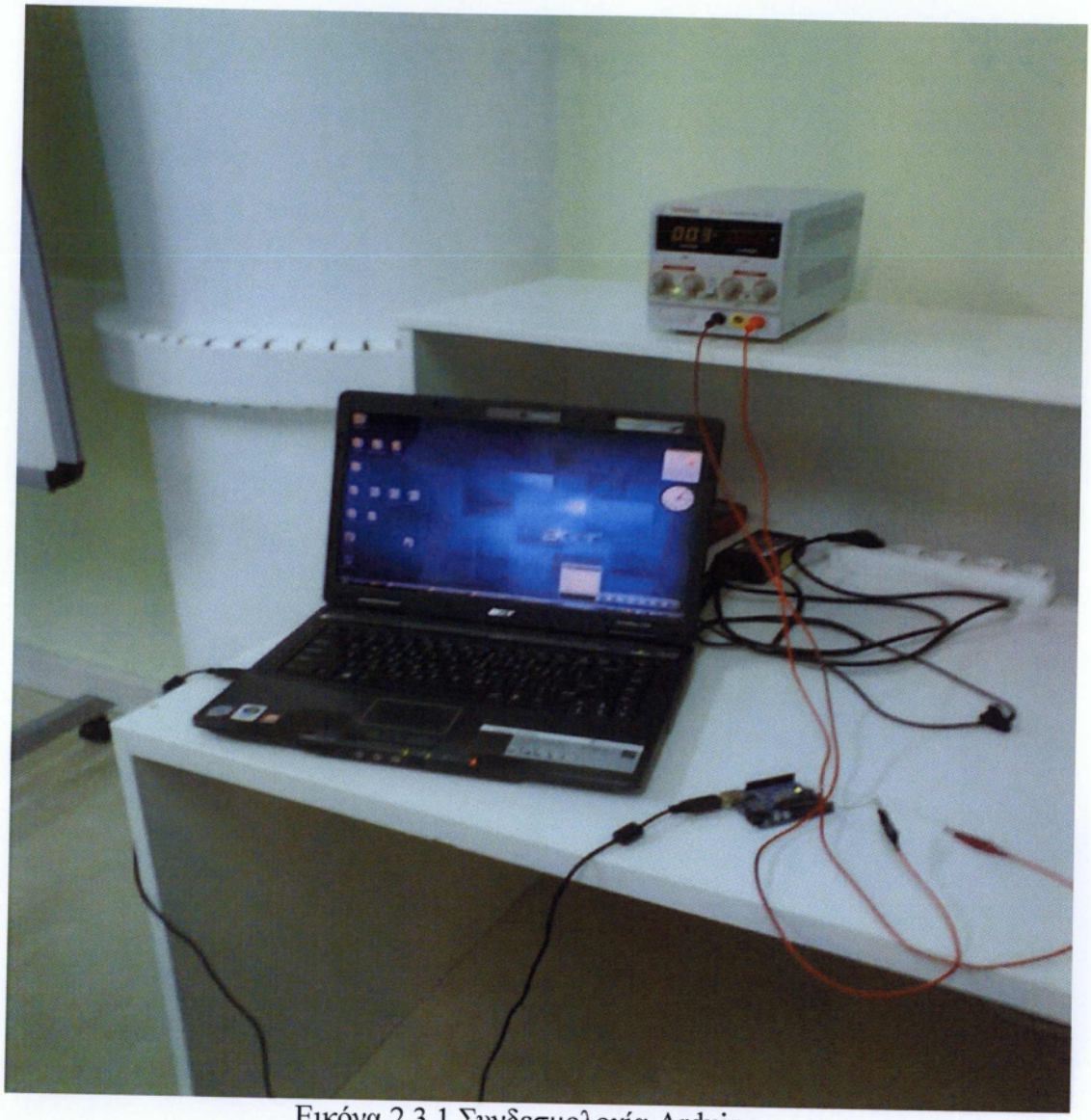
```
34.  }
35.
36.
37.  myPort = new Serial(this, "COM1", 9600);
38. }
39. void draw () {
40.
41. }
42.
43. void serialEvent (Serial myPort) {
44.  // get a Byte:
45.  while (myPort.available() <= 0) {
46.    delay(1);
47.  }
48.
49.  Integer inByte = myPort.read();
50.    println(inByte);
51.    // draw the line:
52.    stroke(127,34,255);
53.    strokeWeight(3);
54.    float real_value = inByte;
55.     line(xPos_last, height-real_value_last-40, xPos, height - real_value-
56. 40);
57.    // at the edge of the screen, go back to the beginning:
58.    if (xPos >= width-40) {
59.      xPos = 40;
60.    }
61.     // increment the horizontal position:
62.     real_value_last=real_value;
63.     xPos_last=xPos;
64.     xPos++;
65.     count_seconds++;
66.     if (count_seconds>=360) exit();
67.    }
```

Επεξήγηση κώδικα: Γραμμές 1 – 9 Δήλωση μεταβλητών
Γραμμές 10 – 35 Δημιουργία παράθυρου και ορίων για απεικόνιση του σήματος εξόδου
Γραμμές 36 – 47 Δήλωση «πόρτας» για επικοινωνία με τον υπολογιστή
Γραμμές 48 – 67 Σχεδίαση σήματος εξόδου

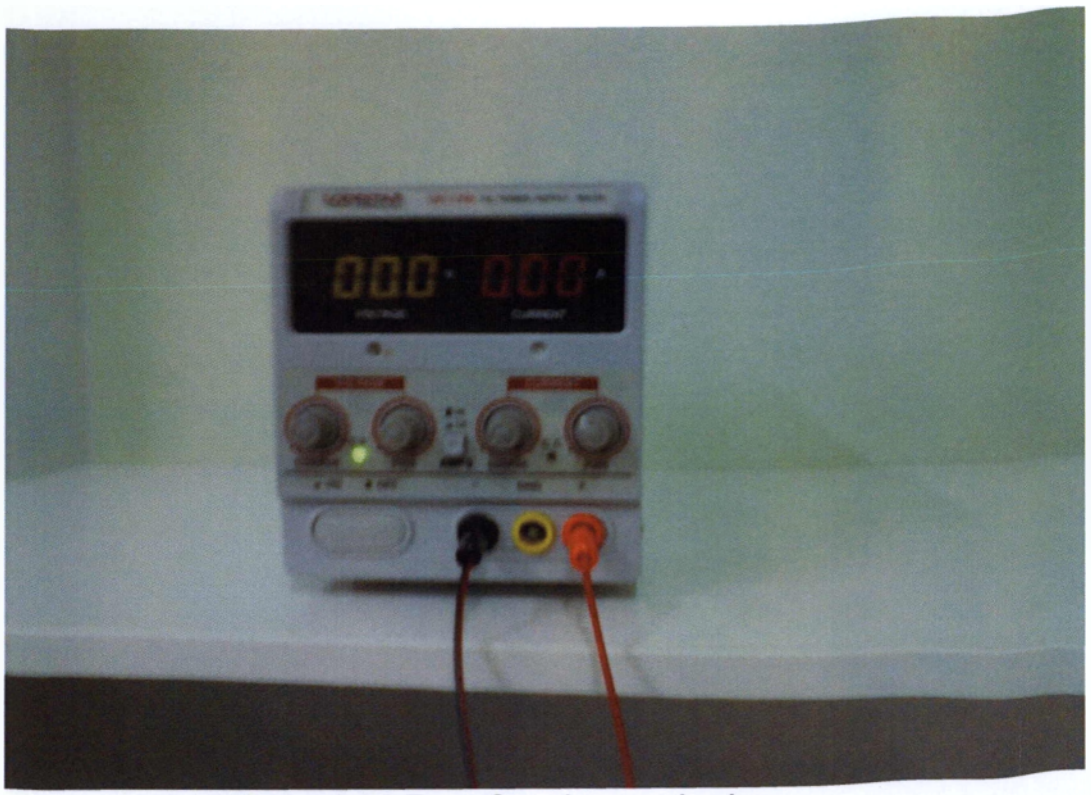
2.3 Συνδεσμολογία

Η συνδεσμολογία γίνεται μέσω ενός καλωδίου, όπου συνδέεται το Arduino με τον υπολογιστή.

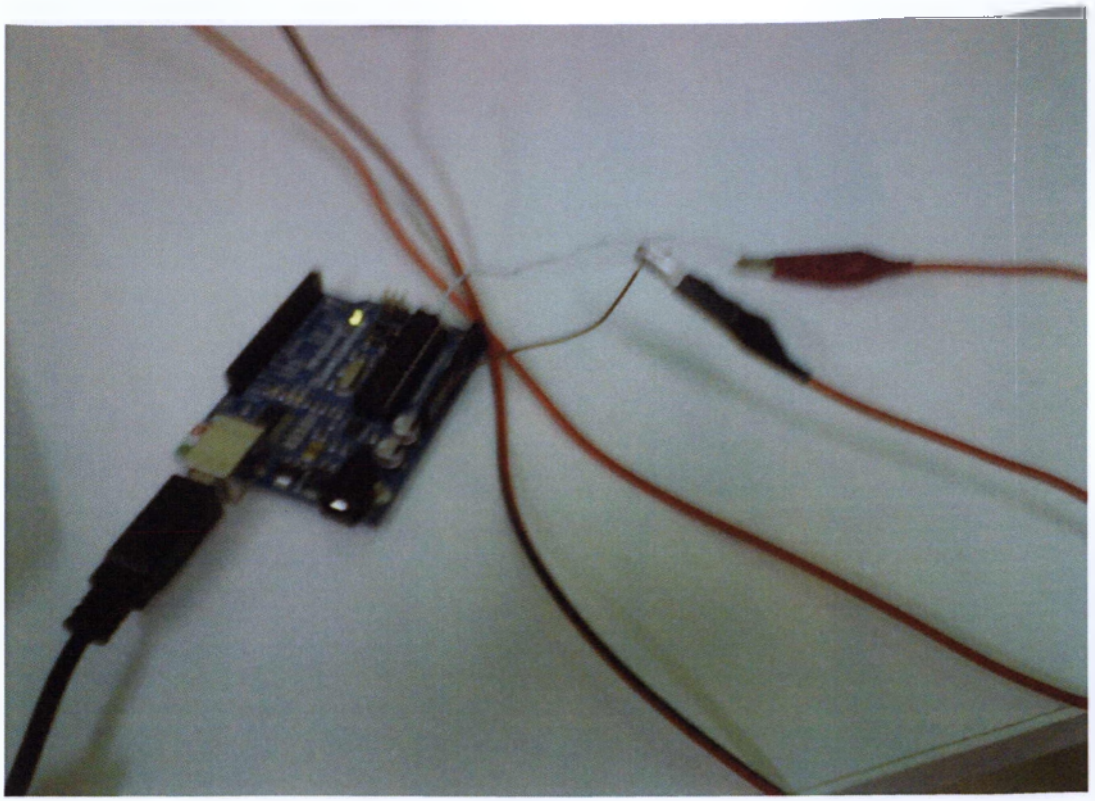
Στη συνέχεια γράφουμε τον αντίστοιχο κώδικα για τον σκοπό της εργασίας και συνδέουμε το κύκλωμα στο τροφοδοτικό συνεχούς τάσης. Η σύνδεση στο τροφοδοτικό συνεχούς τάσης παρουσιάζεται στην Εικόνα 2.3.1.



Εικόνα 2.3.1 Συνδεσμολογία Arduino



Εικόνα 2.3.2 Τροφοδοτικό συνεχούς τάσης

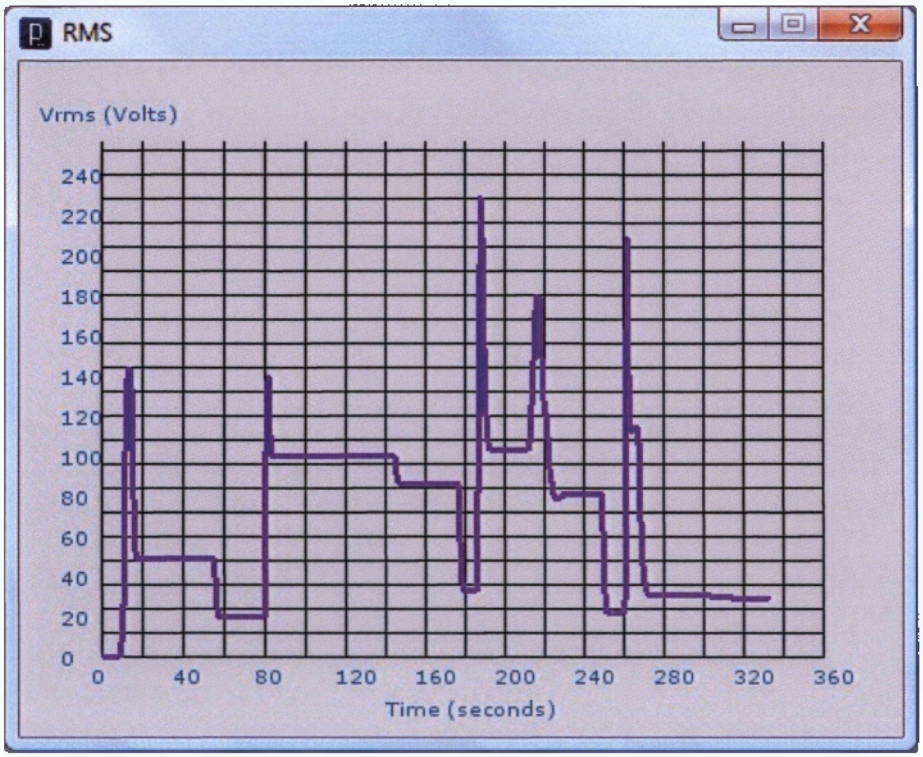


Σύνδεση

Η σύνδεση έγινε στο pin GND, βλέπε μαύρο καλώδιο, όπου συνδέθηκε η ένδειξη voltage του τροφοδοτικού συνεχούς τάσης και στο analog pin 1,

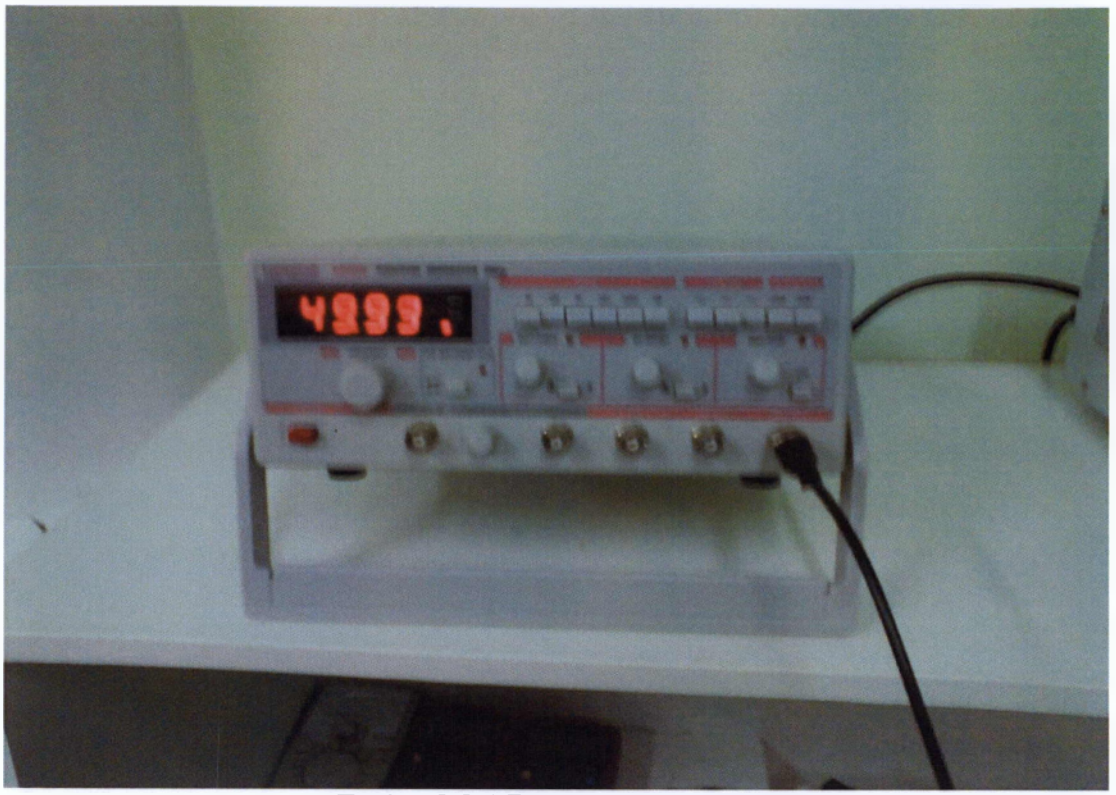
βλέπε κόκκινο καλώδιο, όπου συνδέθηκε η ένδειξη current του τροφοδοτικού συνεχούς τάσης (βλέπε Εικόνα 2.3.2).

Όταν «τρέξουμε» τον κώδικα του Processing και στην ένδειξη voltage του τροφοδοτικού συνεχούς τάσης δώσουμε τιμές μέχρι 5V, οι οποίες αλλάζουν όποτε θέλει ο χρήστης, θα πάρουμε ένα σήμα παρόμοιο με της Εικόνας 2.3.3. Βέβαια, καλό θα ήταν να δίνουμε τιμές μέχρι 3V για να μην καεί η πλακέτα, αφού θα χρησιμοποιείται αρκετά συχνά από τους καθηγητές του εργαστηρίου ηλεκτρονικών για πειράματα.



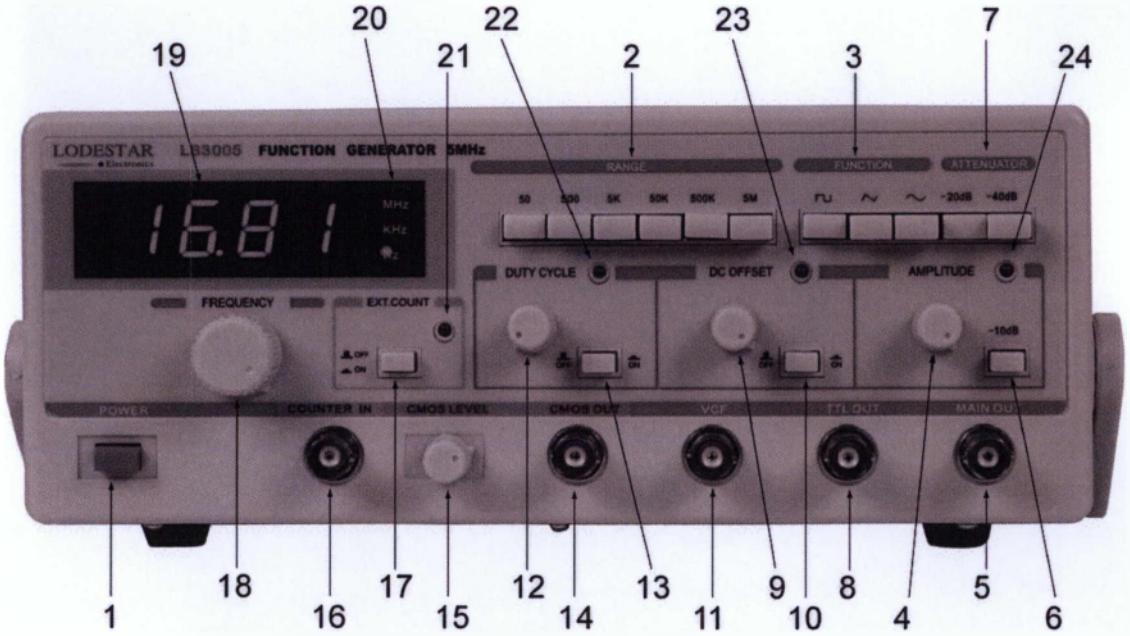
Εικόνα 2.3.3 Σήμα Εξόδου

Όμως το σήμα της Εικόνας 2.3.3 δεν μας επιτρέπει στη μορφή που είναι να δούμε εύκολα τις διακυμάνσεις στις τιμές των Volt στο χρόνο που έχουμε.
Έτσι, θα χρησιμοποιήσουμε μια γεννήτρια συναρτήσεων για να εξαλείψουμε αυτό το πρόβλημα.



Εικόνα 2.3.4 Γεννήτρια Συναρτήσεων

Ακολουθεί μια σύντομη περιγραφή της γεννήτριας συναρτήσεων.

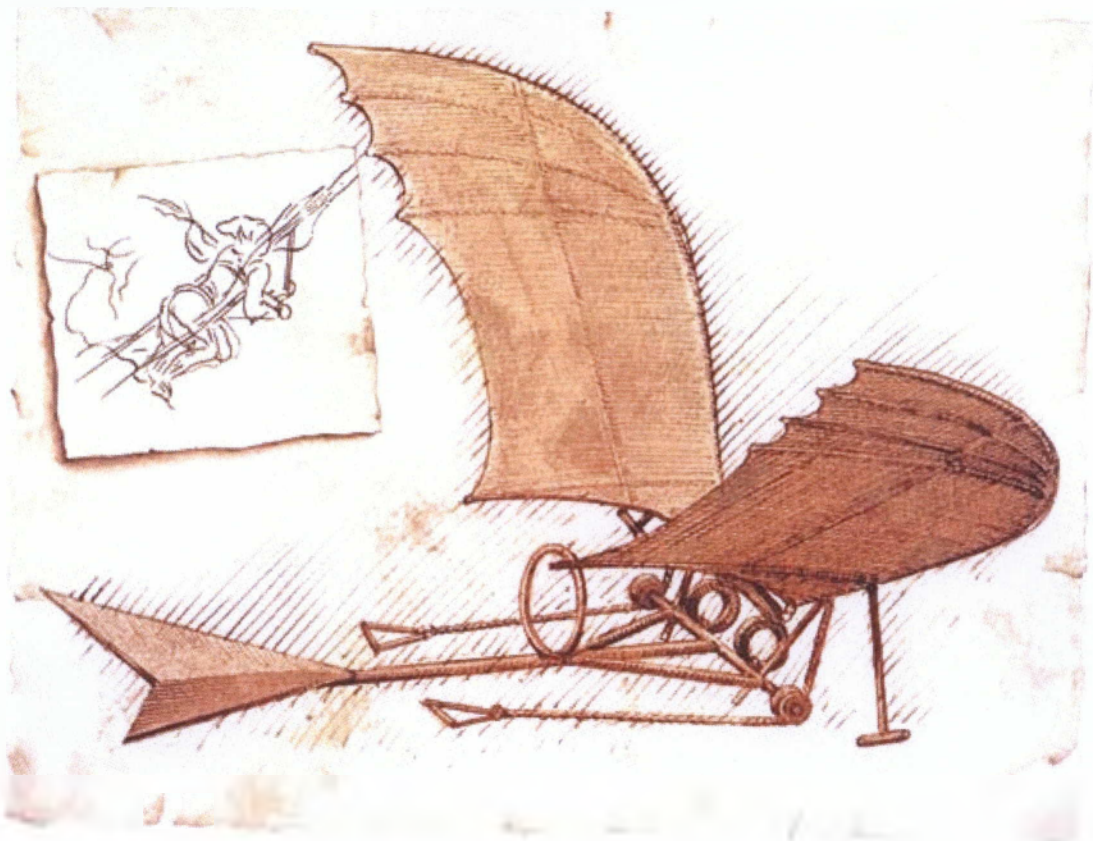


1. Διακόπτης ON/OFF
2. Κουμπιά επιλογής περιοχής συχνοτήτων
3. Κουμπιά επιλογής μορφής σήματος εξόδου
4. Ροοστάτης για τη ρύθμιση του πλάτους του σήματος εξόδου
5. Ακροδέκτης σήματος εξόδου
6. Κουμπί υποβιβασμού του σήματος κατά 10 dB
7. Κουμπιά υποβιβασμού της στάθμης του σήματος εξόδου
8. Έξοδος σήματος TTL. (Ανεξάρτητο των λοιπών ρυθμίσεων εξαρτώμενο μόνο από τη συχνότητα)
9. Ρύθμιση DC Offset του σήματος
10. Κουμπί εισαγωγής DC Offset στο σήμα
11. Voltage Controlled Frequency. Υποδοχή εξωτερικού σήματος ελέγχου της συχνότητας της γεννήτριας
12. Ροοστάτης ρύθμισης του Duty Cycle τετραγωνικών παλμών
13. Κουμπί ενεργοποίησης της ρύθμισης του duty cycle τετραγωνικών παλμών
14. Έξοδος σήματος CMOS
15. Ροοστάτης ρύθμισης του πλάτους του σήματος CMOS
16. Εξωτερική είσοδος στον μετρητή του οργάνου
17. Κουμπί ενεργοποίησης εξωτερικού ελέγχου του μετρητή. (Η συχνότητα εξόδου θα καθορίζεται από το σήμα στην είσοδο 16)
18. Ροοστάτης ρύθμισης συχνότητας
19. Οθόνη απεικόνισης συχνότητας
20. Λυχνία ένδειξης περιοχής συχνότητας
21. Λυχνία ένδειξης εξωτερικού ελέγχου μετρητή
22. Λυχνία ένδειξης ελέγχου duty cycle
23. Λυχνία ένδειξης εισαγωγής DC Offset
24. Λυχνία ένδειξης υποβιβασμού του σήματος

3. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία αυτή σχεδιάστηκε και υλοποιήθηκε σύστημα μέτρησης και καταγραφής της ενεργού (RMS) τιμής της ηλεκτρικής τάσης σε πραγματικό χρόνο για εργαστηριακή χρήση. Η πολυπλοκότητα της σχεδίασης υπήρξε σχετικά περιορισμένη, το δε κόστος αρκετά μικρό (βλέπε πίνακα παραρτήματος 4.1).
Επιπλέον, η αντοχή είναι μεγάλη, η δε συντήρηση είναι εξαιρετικά απλή μιας και η αντικατάσταση γίνεται εύκολα.

4. ΠΑΡΑΡΤΗΜΑΤΑ



4.1 Πίνακας Εξαρτημάτων

Στον παρακάτω πίνακα φαίνεται αναλυτικά το κόστος των εξαρτημάτων που χρησιμοποιήσαμε στην κατασκευή μας.

Πλακέτα	20€
Καλώδιο USB	3€

4.2 Φύλλα Δεδομένων (Datasheets)

Βιβλιοθήκη του Arduino Duemilanove

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)

Arithmetic Operators

- = (assignment)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT
- true | false

- Integer Constants

Data Types

- boolean
- char
- byte
- int
- unsigned int
- long
- unsigned long
- float
- double
- string
- array
- void

Conversion

- char()
- byte()
- int()
- long()
- float()

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogRead()
- analogWrite() - *PWM*

Advanced I/O

- tone()
- noTone()
- shiftOut()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sq()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Εικόνα 4.3.1

Boolean Operators

- && (and)
- || (or)
- ! (not)

Compound Operators

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)

Random Numbers

- randomSeed()
- random()

Communication

- Serial

Εικόνα 4.3.2

Βιβλιοθήκη του Processing

Structure
[] (array access)
= (assign)
catch
class
, (comma)
// (comment)
{} (curly braces)
delay()
/** */ (doc comment)
. (dot)
draw()
exit()
extends
false
final
implements
import
loop()
/* */ (multiline comment)
new
noLoop()
null
() (parentheses)
popStyle()
private
public
pushStyle()
redraw()
return
; (semicolon)
setup()
size()
static
super
this
true

Shape
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()

3D Primitives
box()
sphere()
sphereDetail()

Attributes
ellipseMode()
noSmooth()
rectMode()
smooth()
strokeCap()
strokeJoin()
strokeWeight()

Color

Setting
background()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading
alpha()
blendColor()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

Image
PImage
createImage()

Loading & Displaying
image()
imageMode()
loadImage()
noTint()
requestImage()
tint()

Pixels
blend()

try
void

Environment
cursor()
focused
frameCount
frameRate()
frameRate
height
noCursor()
online
screen
width

Data
Primitive
boolean
byte
char
color
double
float
int
long

Composite
Array
ArrayList
HashMap
Object
String
XMLElement

Conversion
binary()
boolean()
byte()
char()
float()
hex()
int()
str()
unbinary()
unhex()

String Functions
join()
match()

Vertex
beginShape()
bezierVertex()
curveVertex()
endShape()
texture()
textureMode()
vertex()

Loading & Displaying
loadShape()
shape()
shapeMode()

Input
Mouse
mouseButton
mouseClicked()
mouseDragged()
mouseMoved()
mousePressed()
mousePressed
mouseReleased()
mouseX
mousey
pmouseX
pmouseY

Keyboard
key
keyCode
keyPressed()
keyPressed
keyReleased()
keyTyped()

Files
BufferedReader
createInput()
createReader()
loadBytes()
loadStrings()
open()
selectFolder()
selectInput()

Web
link()
param()

copy()
filter()
get()
loadPixels()
pixels[]
set()
updatePixels()

Rendering
PGraphics
createGraphics()
hint()

Typography
PFont

Loading & Displaying
createFont()
loadFont()
text()
textFont()

Attributes
textAlign()
textLeading()
textMode()
textSize()
textWidth()

Metrics
textAscent()
textDescent()

Math
PVector

Operators
+= (add assign)
+ (addition)
-- (decrement)
/ (divide)
/= (divide assign)
++ (increment)
- (minus)
% (modulo)
* (multiply)
*= (multiply assign)
-= (subtract assign)

matchAll()
nf()
nfc()
nfp()
nfs()
split()
splitTokens()
trim()

Array Functions
append()
arrayCopy()
concat()
expand()
reverse()
shorten()
sort()
splice()
subset()

Control
Relational Operators
== (equality)
> (greater than)
>= (greater than or equal to)
!= (inequality)
< (less than)
<= (less than or equal to)

Iteration
For
while

Conditionals
break
case
?: (conditional)
continue
default
else
if
switch()

Logical Operators
&& (logical AND)
! (logical NOT)
|| (logical OR)

status()

Time & Date
day()
hour()
millis()
minute()
month()
second()
year()

Output
Text Area
print()
println()

Image
save()
saveFrame()

Files
PrintWriter
beginRaw()
beginRecord()
createOutput()
createWriter()
endRaw()
endRecord()
saveBytes()
saveStream()
saveStrings()
selectOutput()

Transform
applyMatrix()
popMatrix()
printMatrix()
pushMatrix()
resetMatrix()
rotate()
rotateX()
rotateY()
rotateZ()
scale()
shearX()
shearY()
translate()

Lights, Camera

Bitwise Operators
& (bitwise AND)
| (bitwise OR)
<< (left shift)
>> (right shift)

Calculation
abs()
ceil()
constrain()
dist()
exp()
floor()
lerp()
log()
mag()
map()
max()
min()
norm()
pow()
round()
sq()
sqrt()

Trigonometry
acos()
asin()
atan()
atan2()
cos()
degrees()
radians()
sin()
tan()

Random
noise()
noiseDetail()
noiseSeed()
random()
randomSeed()

Constants
HALF_PI (1.57079...)
PI (3.14159...)
QUARTER_PI (0.78539...)
TWO_PI (6.28318...)

Lights
ambientLight()
directionalLight()
lightFalloff()
lightSpecular()
lights()
noLights()
normal()
pointLight()
spotLight()

Camera
beginCamera()
camera()
endCamera()
frustum()
ortho()
perspective()
printCamera()
printProjection()

Coordinates
modelX()
modelY()
modelZ()
screenX()
screenY()
screenZ()

Material Properties
ambient()
emissive()
shininess()
specular()

4.3 Το Arduino Duemilanove

Μικροελεγκτής	ATmega328
Τάση Λειτουργίας	5V
Τάση Εισόδου	7-12V
Όρια Τάσης	6-20V
Ψηφιακοί Ακροδέκτες I/O	14
Ψηφιακοί Ακροδέκτες Εισόδου	6
DC ρεύμα ανά I/O Ακροδέκτη	40 mA
DC ρεύμα για 3.3V Ακροδέκτη	50 mA
Μνήμη Flash	16 KB
SRAM	1 KB
EEPROM	512 bytes
Ταχύτητα Ρολογιού	16 MHz

-analogRead()

Διαβάζει την αξία από το analog pin. Ο πίνακας Arduino περιλαμβάνει 6 κανάλια (8 κανάλια mini και nano, 16 στο mega), 10 bit αναλογικά στον ψηφιακό μετατροπέα. Αυτό σημαίνει ότι θα ενσωματώσει τις τάσεις μεταξύ 0 και 5 Volt στις τιμές ακέραιων αριθμών μεταξύ 0 και 1023. Αυτό παράγει ένα αποτέλεσμα μεταξύ των: 5 Volt/1024 μονάδες ή, .0049 βολτ (4.9 mV) ανά μονάδα. Τη σειρά και το αποτέλεσμα εισαγωγής μπορεί να αλλάξει χρησιμοποιώντας την συνάρτηση analogReference( ). Διαρκεί περίπου 100 microseconds (0.0001 s) για να διαβάσει μια αναλογική εισαγωγή, έτσι το μέγιστο ποσοστό που μπορεί να αναγνωρίσει (διαβάσει) είναι περίπου 10.000 φορές το δευτερόλεπτο.

Σύνταξη

analogRead(pin)

Παράμετροι

pin: ο αριθμός εισαγωγής (analog input pin) που διαβάζει (από 0 έως 5, από 0 έως 7 σε Arduino Mini και Nano).

Επιστρέφει

int (0 to 1023)

Εάν το αναλογικό Pin εισαγωγής δεν συνδέεται με τίποτα, η αξία που επιστρέφεται από το analogRead( ) θα κυμανθεί βασισμένο σε διάφορους παράγοντες (π.χ. οι τιμές των άλλων αναλογικών εισαγωγών κλπ.).

Για να προγραμματίσουμε το Arduino Duemilanove χρησιμοποιήσαμε αυτές τις δυο εντολές που αναφέρθηκαν πιο πριν. Αρχικά ξεκινήσαμε με την παραγωγή (random) τυχαίων αριθμών με αναλογική είσοδο analogPin=3,analogPin2=10,analogPin1=5 που θα παράγει στην έξοδο τυχαίους αριθμούς.

Όσον αφορά την συνάρτηση **Serial**.begin(9600); καθορίζει το ρυθμό δεδομένων σε bits ανά δευτερόλεπτο (baud) για τη σειριακή μετάδοση δεδομένων.

Για την επικοινωνία με τον υπολογιστή, συνήθως χρησιμοποιούμε μία από αυτές τις τιμές: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ή 115200.

ΠΗΓΕΣ

1. www.wikipedia.org
2. www.google.gr
3. www.arduino.cc
4. www.processing.org
5. www.openenergymonitor.org
6. technically.us/spde/About
7. technically.us/code/x/runaway-processing/
8. technically.us/code/x/flocking-with-spde/
9. processing.org/discourse/yabb2/YaBB.pl?num=1219975973
10. github.com/rosado/clj-processing
11. www.rmx.cz/monsters/
12. www.processinghacks.com/
13. www.openprocessing.org/
14. www.processingblogs.org/
15. ejohn.org/blog/processingjs/
16. groups.google.com/group/processingjs
17. hyper-metrix.com/processing/docs
18. luckylarry.co.uk/category/programming/processing/

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Η γλώσσα C σε βάθος, Χατζηγιαννάκης Νικόλαος, Εκδόσεις Κλειδάριθμος
2. Σημειώσεις Προγραμματισμού Ι, Θεωρία: κ. Πανάγος Νικόλαος, Εργαστήριο: κ. Πανάγος Νικόλαος, κ. Μπάρδης Γεώργιος
3. Κατασκευή Διαμορφωτή / Αποδιαμορφωτή Δέλτα για Εκπαιδευτική Χρήση, Χρήστος Κυλάφας, Σπάρτη 2009
4. Κατασκευή Γεννήτριας Θορύβου, Άννα Ντρέμπου, Σπάρτη 2010
5. Εργαστηριακές Ασκήσεις Αναλογικών Ηλεκτρονικών, Γιάννης Λιαπέρδος, Σπάρτη 2008 (Εργαστηριακό Εγχειρίδιο)
6. Εργαστηριακές Ασκήσεις Ψηφιακών Ηλεκτρονικών, Γιάννης Λιαπέρδος, Σπάρτη 2008 (Εργαστηριακό Εγχειρίδιο)
7. Getting Started with Arduino by Massimo Banzi Co-founder of Arduino
8. Practical Arduino Cool Projects for Open Source Hardware, Jonathan Oxer / Hugh Blemings
9. Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks, Joshua Noble, http://oreilly.com/catalog/9780596154141/
10. Handbook of Power Quality by Angelo Baggini, University of Bergamo – Italy
11. Power Quality in Electrical Systems, Alexander Kusko / Marc T. Thompson
12. Signal Processing of Power Quality Disturbances, Math H.J. Bollen / Irene Y.H. Gu
13. Συστήματα τηλεπικοινωνιών John G. Proakis Masoud Salihi έκδοση ΕΘΝΙΚΟ και ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
14. Getting Started with Processing, Casey Reas, Ben Fry
15. Algorithms for Visual Design Using the Processing Language, Κώστας Τερζίδης, http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470375485.html
16. Processing: A Programming Handbook for Visual Designers and Artists, Casey Reas, Ben Fry, John Maeda, http://mitpress.mit.edu/catalog/item/ default.asp?ttype=2&tid=11251
17. Ben Fry, Visualizing Data, O'Reilly Media, http://oreilly.com/catalog/9 780596514556/
18. Ira Greenberg, Processing: Creative Coding and Computational Art (Foundation), friends of ED, http://friendsofed.com/book.html?isbn=159059617X
19. Daniel Shiffman, Learning Processing: A Beginner's Guide to Programming Images, Animation and Interaction, http://www.learningprocessing.com/

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ – ΕΙΚΟΝΩΝ