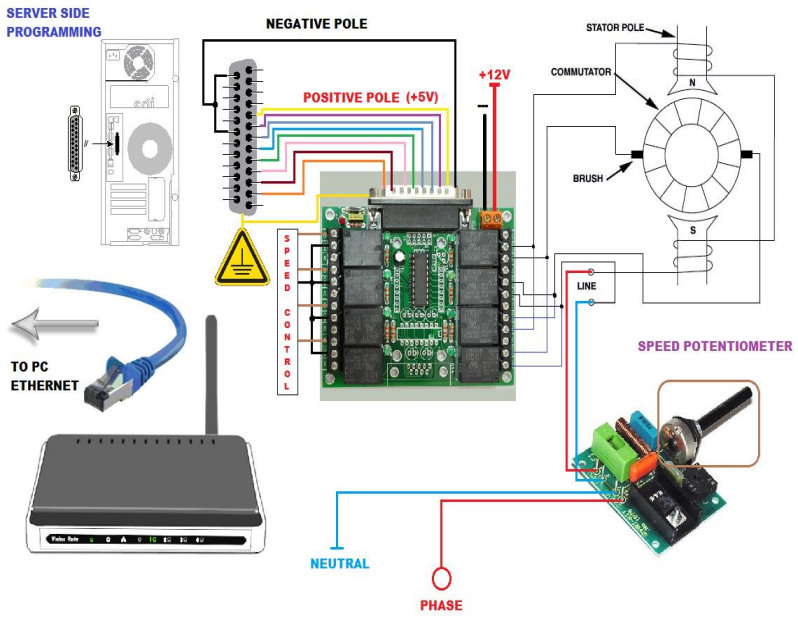
Α.Ε.Ι ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ
ΣΧΟΛΗ: ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ: ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.
Επιβλέπων: ΗΡΑΚΛΗΣ ΒΥΛΛΙΩΤΗΣ, Καθηγητής Εφαρμογών

Μελέτη και κατασκευή συστήματος ασύρματης διαχείρισης ηλεκτρικής συσκευής μέσω διαδικτύου (web controlling)

Design and construction of internet wireless electrical device management system (web controlling)



πτυχιακή εργασία:
Μανδελιάς Δημήτριος-Ηλίας 37878

Αιγάλεω
Οκτώβριος 2016

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΚΕΦΑΛΑΙΟ 2

ΚΕΦΑΛΑΙΟ 3 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Εισαγωγή

Ως τα τέλη του 19ου αιώνα τα ηλεκτρολογικά ζητήματα, θεωρητικά και πρακτικά εξετάζονταν κυρίως από τη φυσική και τους φυσικούς επιστήμονες. Στα τέλη του 19ου λοιπόν αιώνα αρχίζουν να ιδρύονται τα πρώτα ηλεκτρολογικά τμήματα σε πανεπιστήμια της Γερμανίας και της Αμερικής και έτσι η ηλεκτρολογία αρχίζει να ξεχωρίζει ως γνωστικό πεδίο άλλα και ως επάγγελμα χάρη στην εξάπλωση φυσικά και την εμπορευματοποίηση των δικτύων τηλεπικοινωνιών και ηλεκτροδότησης. Όπως είναι γνωστό φυσικά στον 20ο και στον 21ο αιώνα η επιστήμη αλλά και κυρίως η τεχνολογία προοδεύουν με γοργούς ρυθμούς, αυτό επιφέρει σημαντικές αλλαγές σε όλα τα επιστημονικά πεδία αλλά και στα επαγγέλματα των μηχανικών και γενικά όλων των τεχνολόγων. Έτσι η ηλεκτρολογία εξελίσσεται και αυτή, με την εισαγωγή περίπλοκων συστημάτων αυτοματισμού, με τη συνεργασία ηλεκτρονικών και ηλεκτρολογικών συστημάτων και συσκευών, και φυσικά με την χρησιμοποίηση των υπολογιστικών και των ψηφιακών συστημάτων.

Τελευταία εξέλιξη στο χώρο των αυτοματισμών αποτελούν τα plc(Programmable Logic Controllers) ή αλλιώς στα ελληνικά προγραμματιζόμενοι λογικοί ελεγκτές οι οποίοι χρησιμοποιούνται πλέον σε όλες τις βιομηχανίες όποτε αυτές χρειάζονται την εκτέλεση κάποιας πολύπλοκης αυτοματοποιημένης διαδικασίας. Τα plc λοιπόν αποτελούν ένα ζωντανό παράδειγμα της χρήσης υπολογιστικών και ψηφιακών συστημάτων στην ηλεκτρολογία. Τέτοια συστήματα και συσκευές έχουν πολλά πλεονεκτήματα έναντι των κλασσικών αυτοματισμών όπως μικρότερο κόστος συντήρησης, μεγάλη ευελιξία σε τροποποιήσεις και επεκτάσεις αυτοματισμών, οικονομία χώρου κ.α.

Σε αυτή λοιπόν την πτυχιακή εργασία προσπάθησα να εκμεταλλευτώ μεγάλο μέρος των μαθημάτων της σχολής μου(ψηφιακά συστήματα, ηλεκτρονικά, ηλεκτρονικά ισχύος, ηλεκτρικές μηχανές, προγραμματισμός, μικροελεγκτές, εσωτερικές ηλεκτρικές εγκαταστάσεις, ηλεκτρικά κυκλώματα, ηλεκτρικές μετρήσεις κ.α.) καθώς και τις τελευταίες επιστημονικές και τεχνολογικές εξελίξεις που μπορούν να σχετιστούν με την ηλεκτρολογία για να κατασκευάσω ένα σύστημα με το οποίο θα μπορώ εγώ αλλά και κάθε ιδιώτης να ελέγχει τις ηλεκτρικές και ηλεκτρονικές συσκευές του μέσω υπολογιστικών συστημάτων. Με την βοήθεια του διαδικτύου, της ενεργής του κοινότητας φυσικά άλλα και κάποιων τεχνολόγων και επιστημόνων προσπάθησα να πάω τη συσκευή μου ένα βήμα πιο πέρα από τα παραδοσιακά plc και να κάνω την διαχείριση της ασύρματη και απεριόριστης εμβέλειας. Απώτερος σκοπός λοιπόν της εργασίας μου αυτής είναι η απόκτηση εμπειρίας πάνω στην κατασκευή τέτοιων κυκλωμάτων, η περαιτέρω εκπαίδευση μου πάνω στα γνωστικά πεδία που χρησιμοποιήθηκαν στο συγκεκριμένο σύστημα, αλλά και η κατασκευή μιας λειτουργικής συσκευής.

Όσο αφορά τη μελέτη της συσκευής, αποφάσισα να την χωρίσω σε τρία μόνο κεφάλαια, με σκοπό φυσικά ο αναγνώστης να είναι σε θέση να αποκτήσει κάποιες

βασικές γνώσεις για κάθε εξάρτημα που χρησιμοποιείται αλλά και για την τεχνολογία και τη λογική με την οποία αυτά λειτουργούν, και έτσι μέσω όλων αυτών να έχει μια γενική κατανόηση ολόκληρου του κυκλώματος και της λειτουργίας του.

Με βάση τα παραπάνω το πρώτο κεφάλαιο θα αναφέρεται στα ξεχωριστά κυκλώματα που ενώθηκαν για να κατασκευαστεί η συσκευή μας και στον τρόπο λειτουργίας της, θα περιέχει τον ακριβή προγραμματισμό της και κάποιες επεξηγήσεις πάνω σε αυτόν αλλά και ένα εγχειρίδιο για να μπορεί κανείς να την ελέγχει σωστά.

Στο δεύτερο κεφάλαιο θα γίνει μια εισαγωγή στα ψηφιακά συστήματα που μας είναι χρήσιμα στο προγραμματισμό και τον έλεγχο του προγράμματος της συσκευής, σε κάποια μέρη του hardware που χρησιμοποιήσαμε, και στο ηλεκτρολογικό υλικό(ηλεκτρολογικά εξαρτήματα) από τα οποία αποτελούνται τα ξεχωριστά κυκλώματα της συσκευής μας. Επίσης θα μελετήσουμε λίγο και τις ηλεκτρικές μηχανές, αφού επιλέξαμε να ελέγχουμε μια τέτοια.

Τέλος το τρίτο και τελευταίο κεφάλαιο θα έχει σχέση κυρίως με το software μας και την κατανόηση των προγραμμάτων που χρησιμοποιήσαμε. Σε αυτό το κεφάλαιο θα δούμε βασικές εντολές, τελεστές και μεταβλητές των σημαντικότερων προγραμμάτων που χρησιμοποιήσαμε και παραδείγματα κώδικα πάνω σε αυτές, έτσι ώστε ο αναγνώστης να έχει μια σφαιρική αντίληψη πάνω στον προγραμματισμό και να μπορεί μόνος του να ελέγξει το βασικό μας πρόγραμμα.

ΚΕΦΑΛΑΙΟ 1

1.1 Μέθοδος λειτουργίας και εγχειρίδιο χρήστη

Η συσκευή λειτουργεί σε τέσσερις καταστάσεις, αριστερή περιστροφή, δεξιά περιστροφή, φρένο και νεκρό σημείο. Επίσης διαθέτει τέσσερις διαφορετικές ταχύτητες, οι οποίες είναι ρυθμιζόμενες από τέσσερα αντίστοιχα ποτενσιόμετρα εγκαταστημένα στο ρυθμιστή στροφών της συσκευής. Αφού έχουμε ολοκληρώσει τις συνδέσεις, έχουμε τη δυνατότητα να διαχειριστούμε τον ηλεκτρικό μας κινητήρα με δυο διαφορετικούς τρόπους.

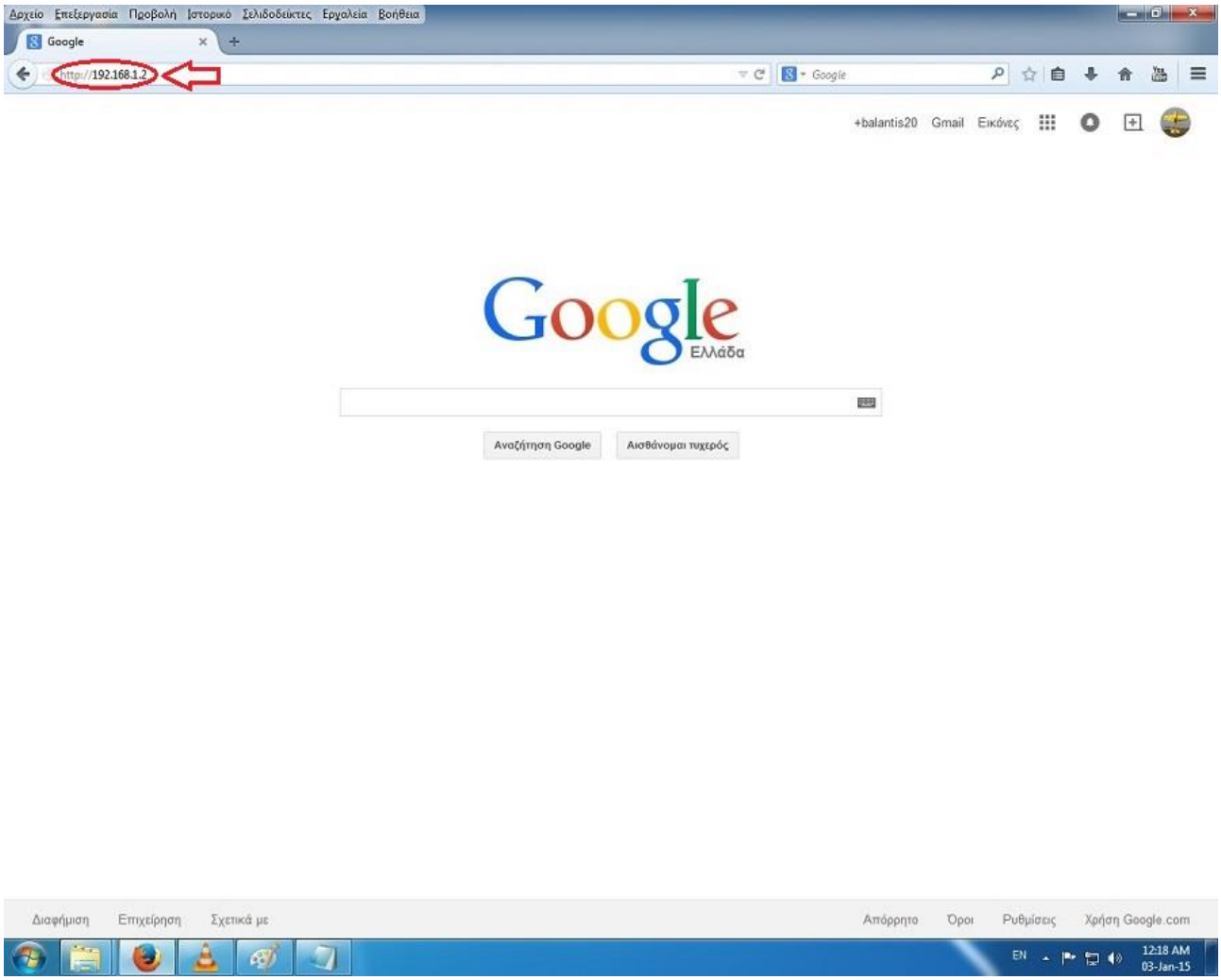
Ο πρώτος τρόπος είναι μέσω της τοπικής του σύνδεσης με τον ηλεκτρονικό υπολογιστή, χρησιμοποιώντας στην ουσία το ενσωματωμένο πρόγραμμα visual basic.net, αυτό το πρόγραμμα μας δίνει πλήρη διαχείριση του ηλεκτρικού κινητήρα μέσω του απαραίτητου πίνακα ελέγχου που έχει εγκατεστημένο.

Ο δεύτερος τρόπος είναι μέσω της χρήσης του περιηγητή(browser), με αυτόν τον τρόπο μπορούμε στην ουσία να έχουμε και απεριόριστη εμβέλεια στη διαχείριση της συσκευής μας μέσω φυσικά του διαδικτύου. Θα πρέπει λοιπόν να συνδεθούμε στο τοπικό δίκτυο(default SSID: VALE_DEKARAKI) και να πληκτρολογήσουμε στη γραμμή διευθύνσεων τον προκαθορισμένο σύνδεσμο(http://192.168.1.2).Μόλις μας εμφανίσει την απαραίτητη ιστιοσελίδα διαχείρισης, τότε θα είμαστε και έτοιμοι να χειριστούμε τον ηλεκτρικού κινητήρα μας στέλνοντας τις εντολές που θέλουμε μέσω των submit buttons. Ελέγχουμε την κατάσταση του ηλεκτρικοί κινητήρα ή οποιασδήποτε άλλης συσκευής θέλουμε να διαχειριστούμε μέσω του ενσωματωμένου display στο κέντρο της οθόνης. Χάρη στη μέθοδο ajax μπορούμε να έχουμε την εικόνα της κατάστασης του κινητήρα σε πραγματικό χρόνο, αποφεύγοντας τις συνεχείς ανανεώσεις της ιστοσελίδας.

Στις επόμενες σελίδες λοιπόν ακολουθεί ένα απλό εγχειρίδιο χρήσης της συσκευής(σε απεριόριστη εμβέλεια), μαζί με τις κατάλληλες εικόνες.

ΒΗΜΑ 1

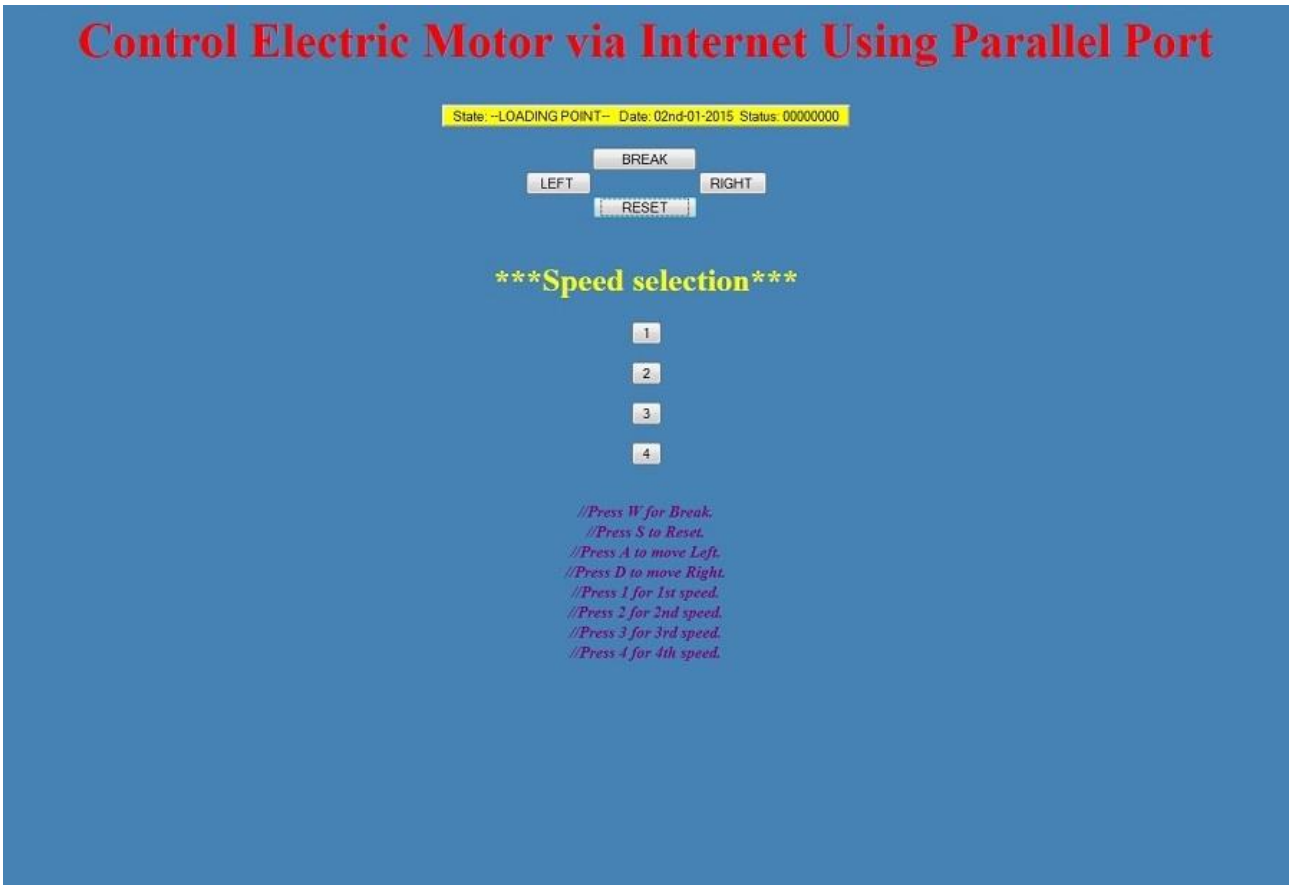
Αφού συνδεθούμε στο τοπικό ασύρματο δίκτυο(Default SSID: VALE_DEKARAKI) πληκτρολογούμε στην γραμμή διευθύνσεων τον προκαθορισμένο σύνδεσμο(http://192.168.1.2) για την πρόσβαση στο σύστημα ελέγχου του κινητήρα.



εικόνα 1.1: σελίδα google, στο πάνω αριστερά μέρος βλέπουμε τον σύνδεσμο που πρέπει να πληκτρολογήσουμε.

ΒΗΜΑ 2

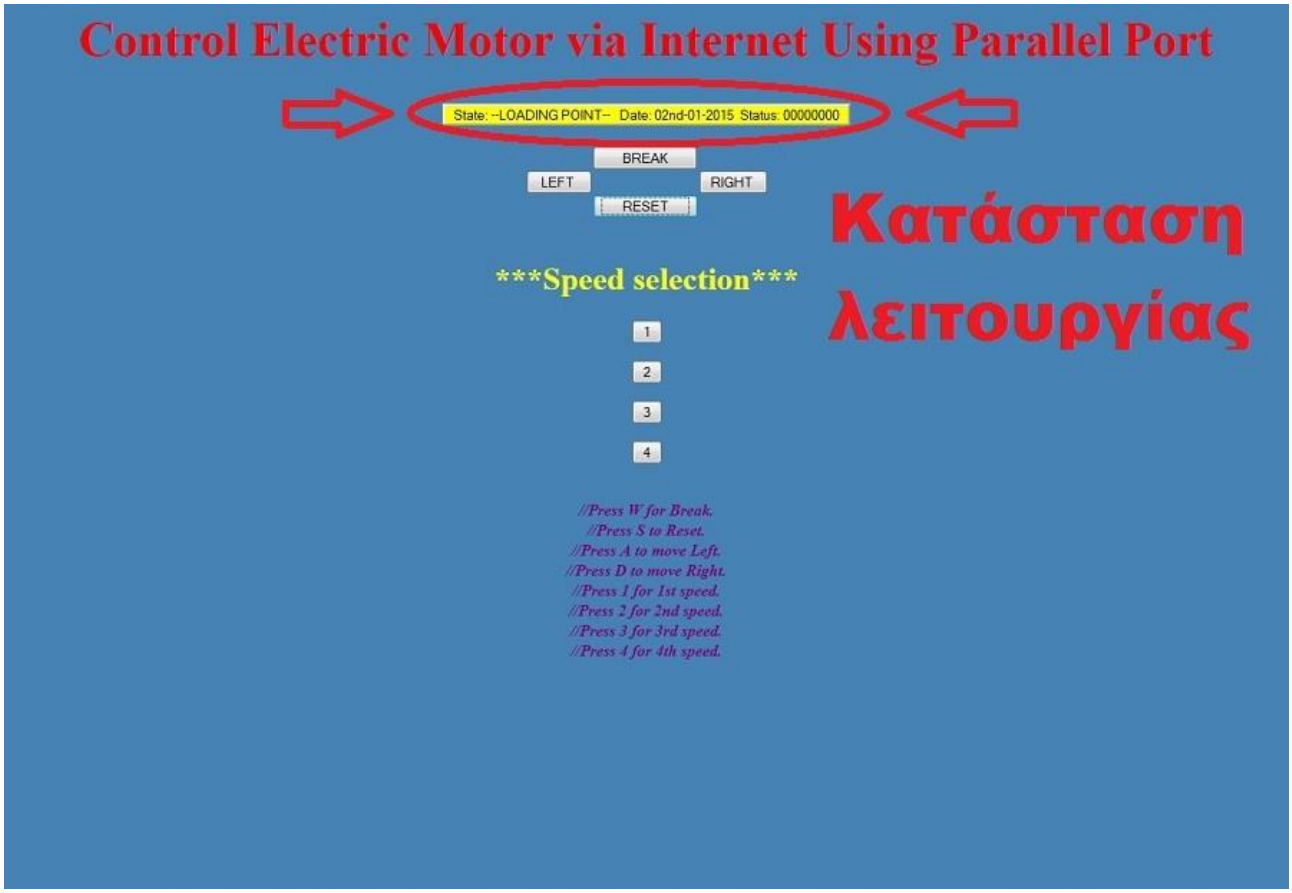
Μόλις μας εμφανίσει την απαραίτητη ιστοσελίδα διαχείρισης, τότε είμαστε έτοιμοι να χειριστούμε τον κινητήρα στέλνοντας τις εντολές που θέλουμε μέσω των submit buttons.



εικόνα 1.2: Το control panel της ιστιοσελίδας μας.

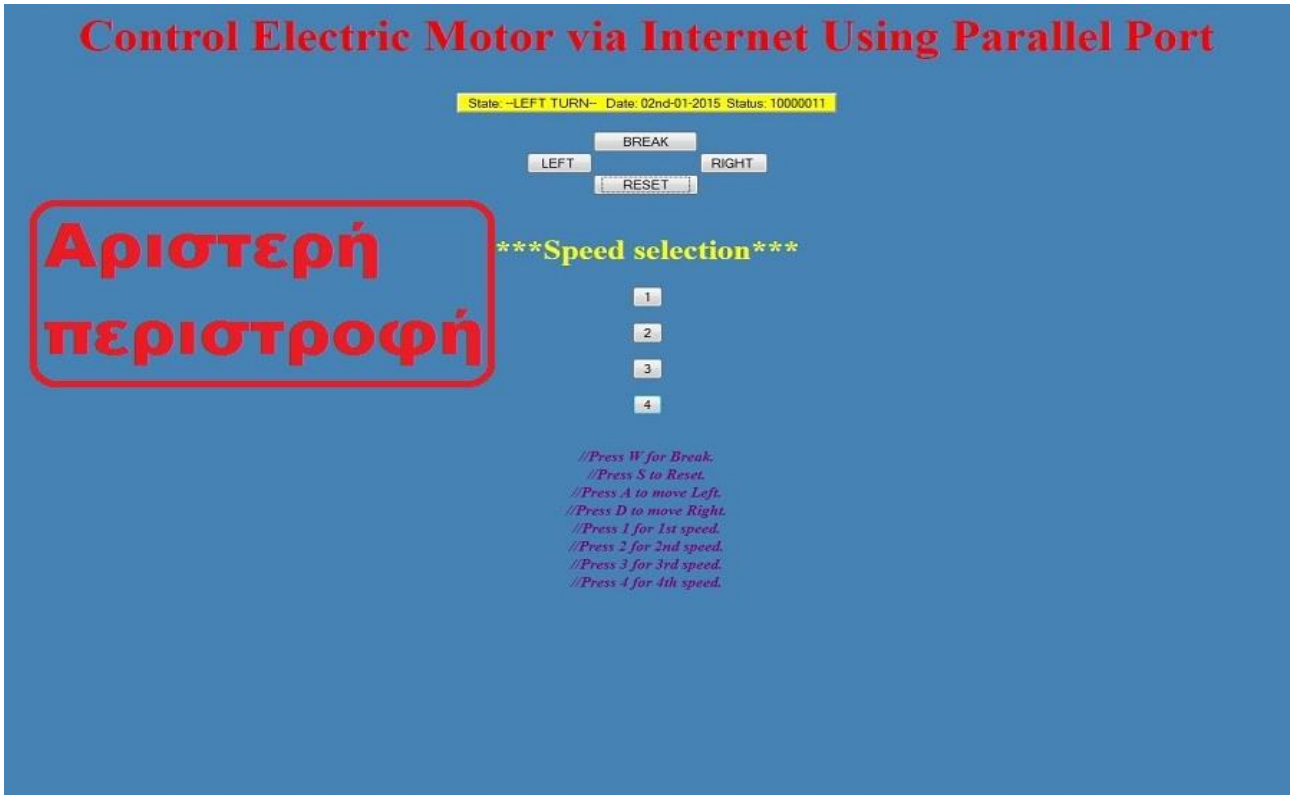
ΒΗΜΑ 3

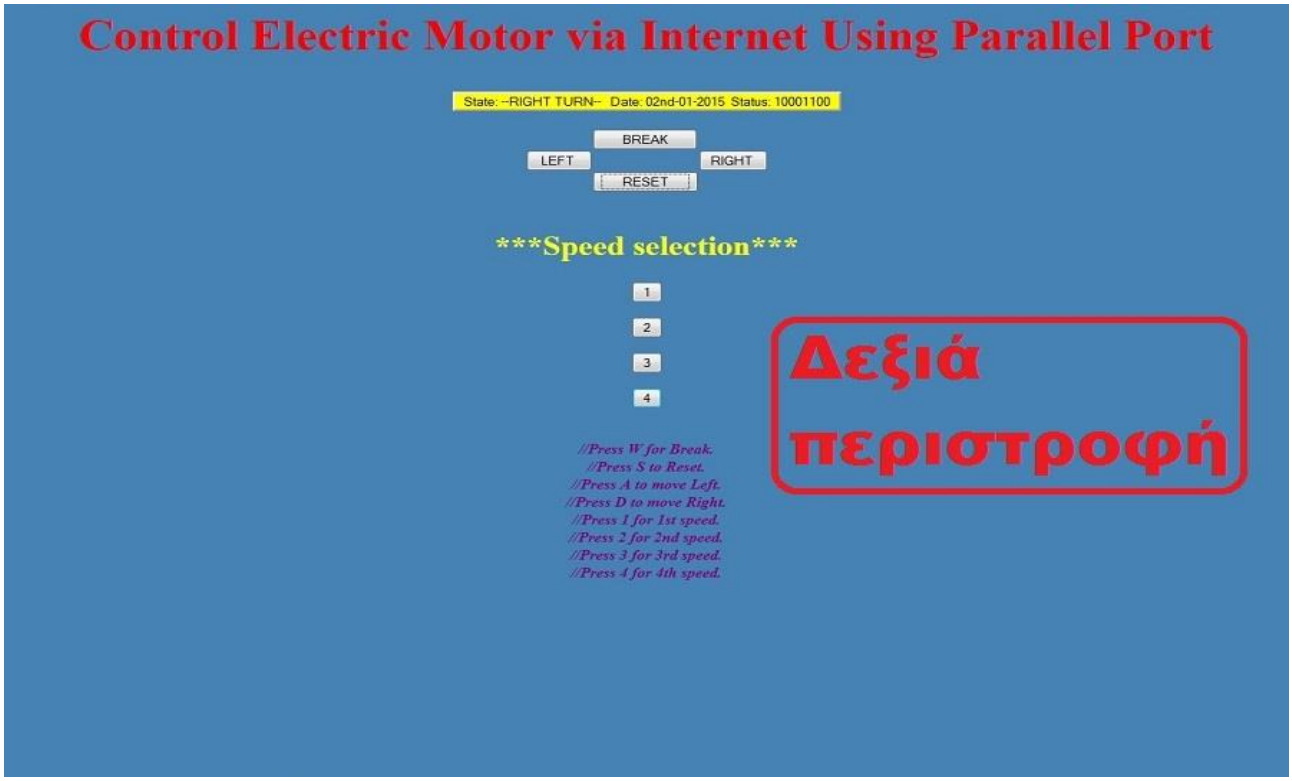
Ελέγχουμε την κατάσταση του κινητήρα ή της συσκευής που θέλουμε να διαχειριστούμε, μέσω του ενσωματωμένου display στο κέντρο της οθόνης. Η μέθοδος AJAX μας δείχνει την κατάσταση του κινητήρα σε πραγματικό χρόνο αποφεύγοντας συνεχείς ανανεώσεις της ιστοσελίδας.



Εικόνα 1.3: Στην κίτρινη λεζάντα βλέπουμε την κατάσταση λειτουργίας της συσκευής μας (στη συγκεκριμένη περίπτωση νεκρό σημείο).

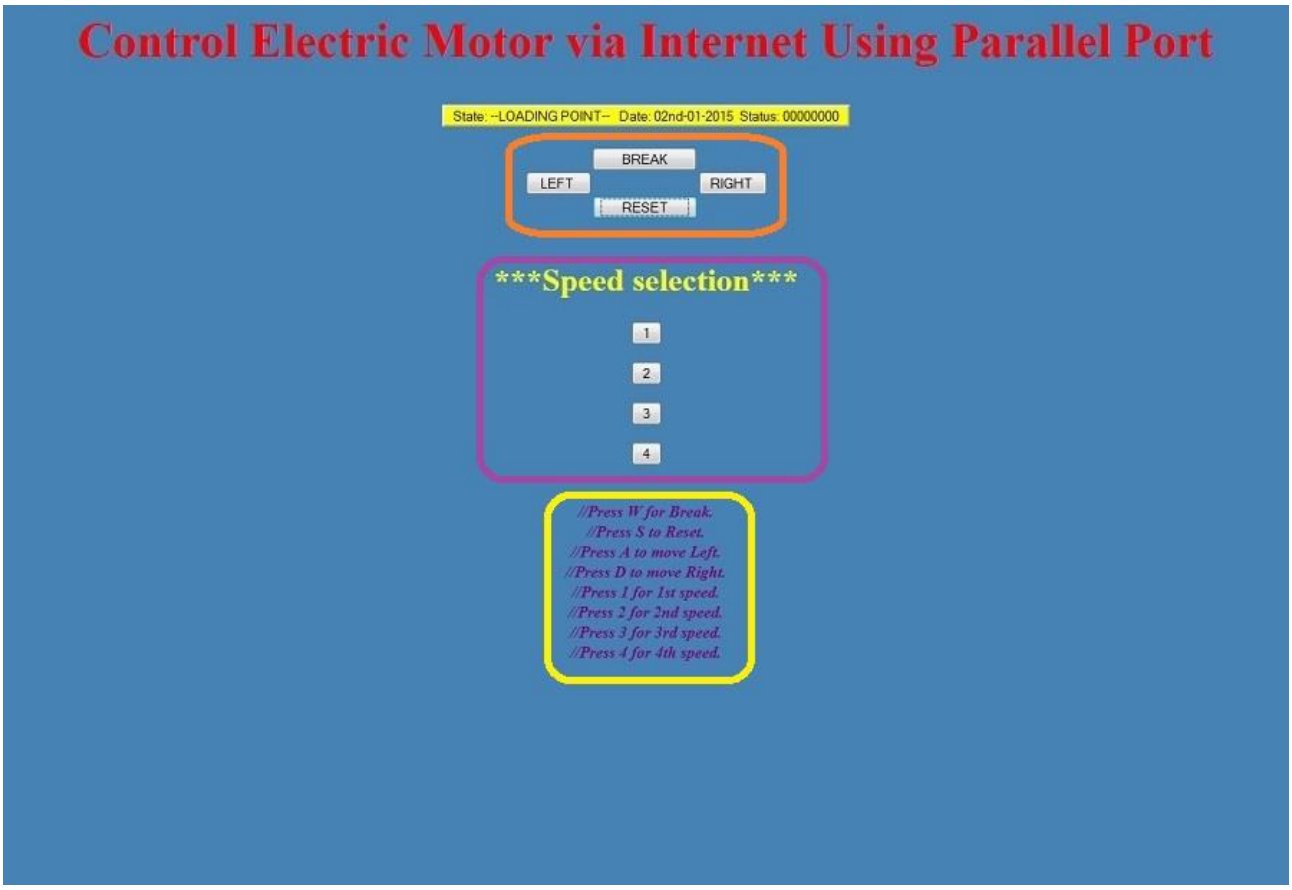
Στην παραπάνω εικόνα ο ηλεκτρικός κινητήρας βρίσκεται σε κατάσταση ηρεμίας, στις επόμενες δυο εικόνες σε αριστερόστροφη και δεξιόστροφη αντίστοιχα περιστροφή. Μπορούμε να διακρίνουμε ότι στην οθόνη μας εκτός από την ημερομηνία και την κατάσταση περιστροφής εμφανίζεται και ένας δυαδικός αριθμός(status), αυτός ο αριθμός μας δείχνει ποιοι ακροδέκτες της παράλληλης θύρας μας λειτουργούν τη συγκεκριμένη στιγμή.





Εικόνα 1.4, 1.5: Αριστερή και δεξιά περιστροφή.

ΑΝΑΛΥΣΗ ΚΑΤΑΣΤΑΣΕΩΝ



εικόνα 1.6 : Οι ξεχωριστές λειτουργίες του control panel.

Κάτω από τη λεζάντα της κατάστασης λειτουργίας έχουμε τα buttons των καταστάσεων λειτουργίας που μπορούμε να επιλέξουμε (break: φρένο, left: αριστερή περιστροφή, right: δεξιά περιστροφή, reset: μηδενισμός καταστάσεων ή αλλιώς νεκρό σημείο).

Αμέσως μετά βλέπουμε τις ταχύτητες που μπορούμε να επιλέξουμε, πατώντας φυσικά το αντίστοιχο button, 1, 2, 3, και 4 αντιστοιχούν στην πρώτη, δεύτερη, τρίτη και τέταρτη ταχύτητα.

Τέλος χάρη στη μέθοδο εισόδου εντολών με χρήση πλήκτρων (I/O inputs keydown events) μπορούμε να χρησιμοποιήσουμε και το πληκτρολόγιο για να δώσουμε εντολές, βλέπουμε στην εικόνα 1.6 τα πλήκτρα του πληκτρολογίου που μπορούμε να χρησιμοποιήσουμε και τι εντολές εκτελούν.

*****ΔΙΕΥΚΡΙΝΙΣΗ*****

Για την επίτευξη περιστροφής του κινητήρα είναι απαραίτητη η προεπιλογή της επιθυμητής ταχύτητας(1,2,3,4) και ύστερα της φοράς(Right/Left) ή το αντίθετο.Σε κάθε περίπτωση ο χρήστης ενημερώνεται για την κατάσταση από το ενσωματωμένο display.

1.2 Συσκευές που χρησιμοποιήθηκαν, κατασκευή και λειτουργιά του συστήματος

1.2.1 Συσκευές που χρησιμοποιήθηκαν

Ηλεκτρονικός υπολογιστής

Το πρώτο πράγμα που χρειαζόμαστε είναι ένας ηλεκτρονικός υπολογιστής ο οποίος θα λειτουργεί σαν server.Ο ηλεκτρονικός μας υπολογιστής λοιπόν θα πρέπει να είναι αρκετά ισχυρός έτσι ώστε να μπορεί να ''σηκώνει'' όλα τα προγράμματα που θα χρησιμοποιήσουμε, πράγμα αρκετά εύκολο αφού έτσι κι αλλιώς τα προγράμματα είναι σχετικά ελαφριά. Το πιο σημαντικό που πρέπει να έχει ο υπολογιστής είναι η παράλληλη θύρα, για την οποία θα μιλήσουμε αναλυτικότερα στα επόμενα κεφάλαια(βλέπε parallel port), για την ώρα αυτό που πρέπει να ξέρουμε είναι ότι χάρη σε αυτή τη θύρα ο υπολογιστής μας μπορεί να έχει επικοινωνία με τον έξω(αναλογικό) κόσμο. Με τη βοήθεια λοιπόν ενός ηλεκτρονικού υπολογιστή και μιας παράλληλης θύρας μπορούμε να ελέγχουμε την ηλεκτρική συσκευή μας. Εδώ θα πρέπει να μιλήσουμε ίσως για τις περιφερειακές συσκευές του υπολογιστή. Ξεκινώντας λοιπόν θα πρέπει να πούμε ότι σε περίπτωση που θέλουμε να ελέγξουμε τη συσκευή τοπικά, χωρίς δηλαδή τη βοήθεια του διαδικτύου, τότε καλό είναι ο υπολογιστής μας να έχει σύνδεση με μια οθόνη,

ποντίκι και κυρίως πληκτρολόγιο, στην περίπτωση τώρα της ασύρματης διαχείρισης όλα τα παραπάνω δεν είναι απαραίτητα, χρειαζόμαστε όμως σίγουρα σύνδεση στο διαδίκτυο.

Router (δρομολογητής)

Με τη βοήθεια λοιπόν ενός δρομολογητή έχουμε τώρα τη δυνατότητα της απεριόριστης εμβέλειας ασύρματης διαχείρισης της ηλεκτρικής μας συσκευής, μέσω internet φυσικά. Όπως είπαμε και πριν πλέον δεν έχουμε την ανάγκη άλλων περιφερειακών συσκευών καθώς ο έλεγχος γίνεται μέσω οποιασδήποτε συσκευής με σύνδεση στο διαδίκτυο, όπως για παράδειγμα ένα smartphone ή ένας άλλος υπολογιστής.

Parallel port controller

Η συσκευή όπως θα δούμε και σε επόμενα κεφάλαια χρησιμοποιείται μαζί με την παράλληλη θύρα του υπολογιστή έτσι ώστε να μπορούμε να δίνουμε μέσω αυτού τις εντολές που θέλουμε για τον έλεγχο του ηλεκτρικού κυκλώματος που χρησιμοποιούμε. Σε επόμενες σελίδες θα μιλήσουμε πιο αναλυτικά για τη λειτουργία της και για τη σχέση της με ολόκληρο το σύστημα, για την ώρα αρκεί να πούμε ότι εκτός από τη σύνδεση της με τον υπολογιστή, πρέπει να είναι συνδεδεμένη και με μια μόνιμη πηγή συνεχούς ρεύματος 12V.

Τροφοδοτικά

Οπως είναι γνωστό η εφαρμογή αυτή αποτελείται από συσκευές που λειτουργούν και με εναλλασσόμενο αλλά και με συνεχές ρεύμα. Αφού η πηγή μας ειναι εναλλασσόμενη τότε θα πρέπει να μετατρέψουμε το ρεύμα σε συνεχές, έτσι ώστε να τροφοδοτήσουμε τις ηλεκτρονικές συσκευές του συστήματος. Χρειαζόμαστε λοιπόν δυο τροφοδοτικά των 12V, το ένα για την τροφοδότηση του parallel port controller και το άλλο για αυτή του router.

Ρυθμιστής στροφών

Πρόκειται για μια συσκευή με την οποία ρυθμίζουμε τις στροφές σε τρυπάνια, μοτέρ κλπ. Ιδανικό, όταν θέλουμε να βιδώσουμε μια βίδα με το τρυπάνι μας ή να κάνουμε περιέλιξη πηνίου με τη βοήθεια δραπάνου όπου χρειάζονται λίγες στροφές. Είναι μια κατασκευή απλή, εύκολη και χρήσιμη ειδικά για τα παλαιού τύπου τρυπάνια που δεν διαθέτουν ρύθμιση στροφών.

Λειτουργία

Το κύκλωμα είναι απλό και κλασσικό. Υπάρχει ένα δικτύωμα RC το οποίο καθορίζει το σημείο έναυσης της διόδου DIAC(D1) η οποία με τη σειρά της ενεργοποιεί το TRIAC(TRI1).

Οι χρόνοι φόρτισης και εκφόρτισης του πυκνωτή C3 εξαρτώνται άμεσα από το ποτενσιόμετρο P2 και το τρίμερ P1. Όταν ο πυκνωτής C3 φορτίσει στη τιμή τάσης σκανδαλισμού του DIAC τότε αυτό οδηγεί το TRIAC το οποίο είναι ο αμφίδρομος διακόπτης μεταξύ της τάσης των 230 V AC και του φορτίου.

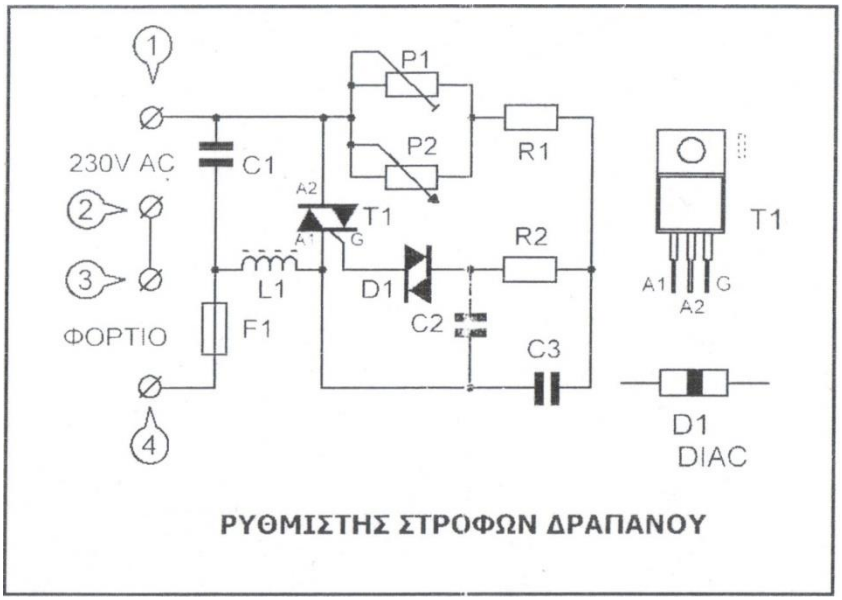
Οι αιχμές θορύβου που δημιουργούνται στο κύκλωμα από την λειτουργιά του TRIAC καταστέλλονται από τον πυκνωτή C1 και το πηνίο L1.

Η ασφάλεια εξαρτάται από το φορτίο. Για φορτίο 600 Watt η τιμή της θα πρέπει να είναι τουλάχιστον 3,15 A. Η ψήκτρα τύπου “Π” που συνοδεύει το kit κάνει θαυμάσια δουλειά στην θερμοαπαγωγή του TRIAC ώστε να μην υπάρχουν προβλήματα υπερθέρμανσης. Το τρίμμερ T1 ρυθμίζει το κάτω όριο του ποτενσιόμετρου P2 δηλαδή εάν θα σταματήσει εντελώς το τρυπάνι ή όχι.

Κατασκευή-σύνδεση-ρύθμιση

Η κατασκευή είναι πολύ εύκολη. Όλα τα εξαρτήματα συναρμολογούνται πάνω σε μια πλακέτα η οποία έχει το τοπογραφικό ώστε να μας καθοδηγήσει σε ποιο σημείο θα τοποθετηθεί το κάθε εξάρτημα. Αρχίζουμε την συναρμολόγηση με της αντιστάσεις R1, R2, το DIAC D1 τους πυκνωτές C1, C2, C3 την ασφαλειοθήκη όπου θα τοποθετηθεί η ασφάλεια, το πηνίο L1, το τρίμμερ P1 και το ποτενσιόμετρο P2. Στα σημεία 1, 2, 3 και 4 θα τοποθετηθούν τα Pins που υπάρχουν μέσα στο kit. Το TRIAC TRI1 θα τοποθετηθεί σε οριζόντια θέση αφού πρώτα το λυγίσουμε ανάλογα και κατόπιν θα βιδώσουμε και τα δυο πάνω στην πλακέτα. Προσοχή δεν πρέπει να ακουμπήσουμε τα ποδαράκια του πάνω στην ψήκτρα.

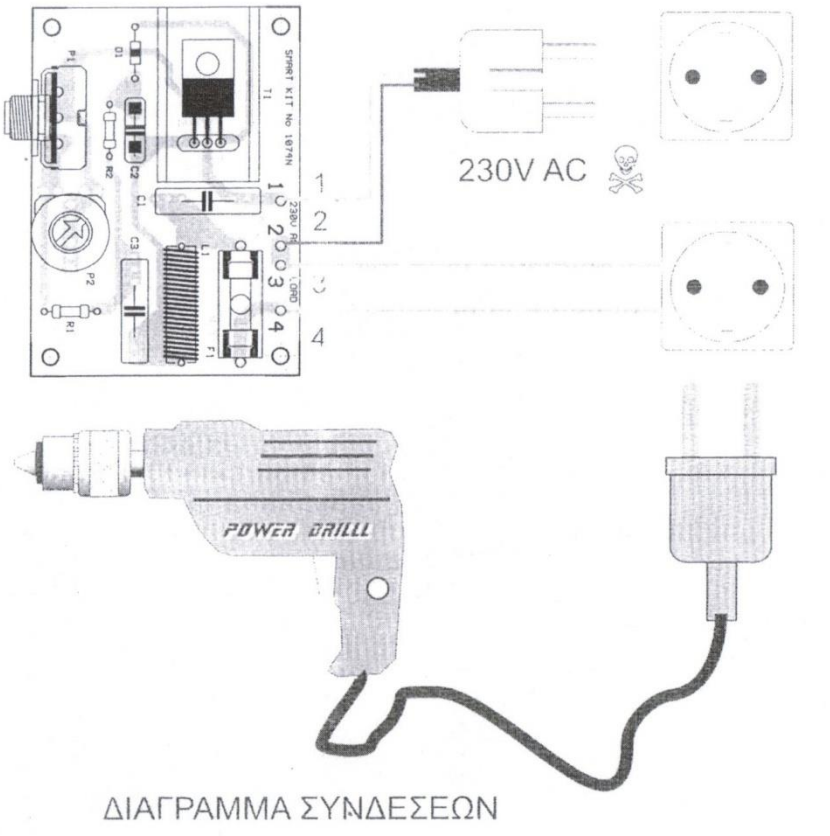
Η ρύθμιση γίνεται ως εξής: Το P1 ρυθμίζεται τέρμα αριστερόστροφα. Κατόπιν ρυθμίζεται το ποτενσιόμετρο P2 μέχρι το δράπανο να άγει στην ελάχιστη τιμή που θέλουμε. Στα σημεία (1) και (2) συνδέεται η τάση του δικτύου ενώ στα σημεία (3) και (4) το φορτίο.



Εικόνα 1.7: Το διάγραμμα του ρυθμιστή στροφών.

Τα υλικά

R1:	2,2 ΚΩ - 1/4 W – 5%	
R2:	6,8 ΚΩ - 1/4 W – 5%	
C1:	150 nF 250 V	πυκνωτής πολυεστέρα
C2:	33 nF 250 V	πυκνωτής πολυεστέρα
C3:	150 nF 250 V	πυκνωτής πολυεστέρα
P1:	1ΜΩ	τρίμερ
P2:	470 ΚΩ	γραμμικό ποτενσιόμετρο
F1:	3,15 A	ασφάλεια
L1:	πηνίο	
D1:	BR100 ή BD3	DIAC
TRI1:	BTB08-400B-BT136	400 V/10 A TRIAC



Εικόνα 1.8: Το διάγραμμα συνδέσεων του ρυθμιστή στροφών.

Αν δεν δουλέψει

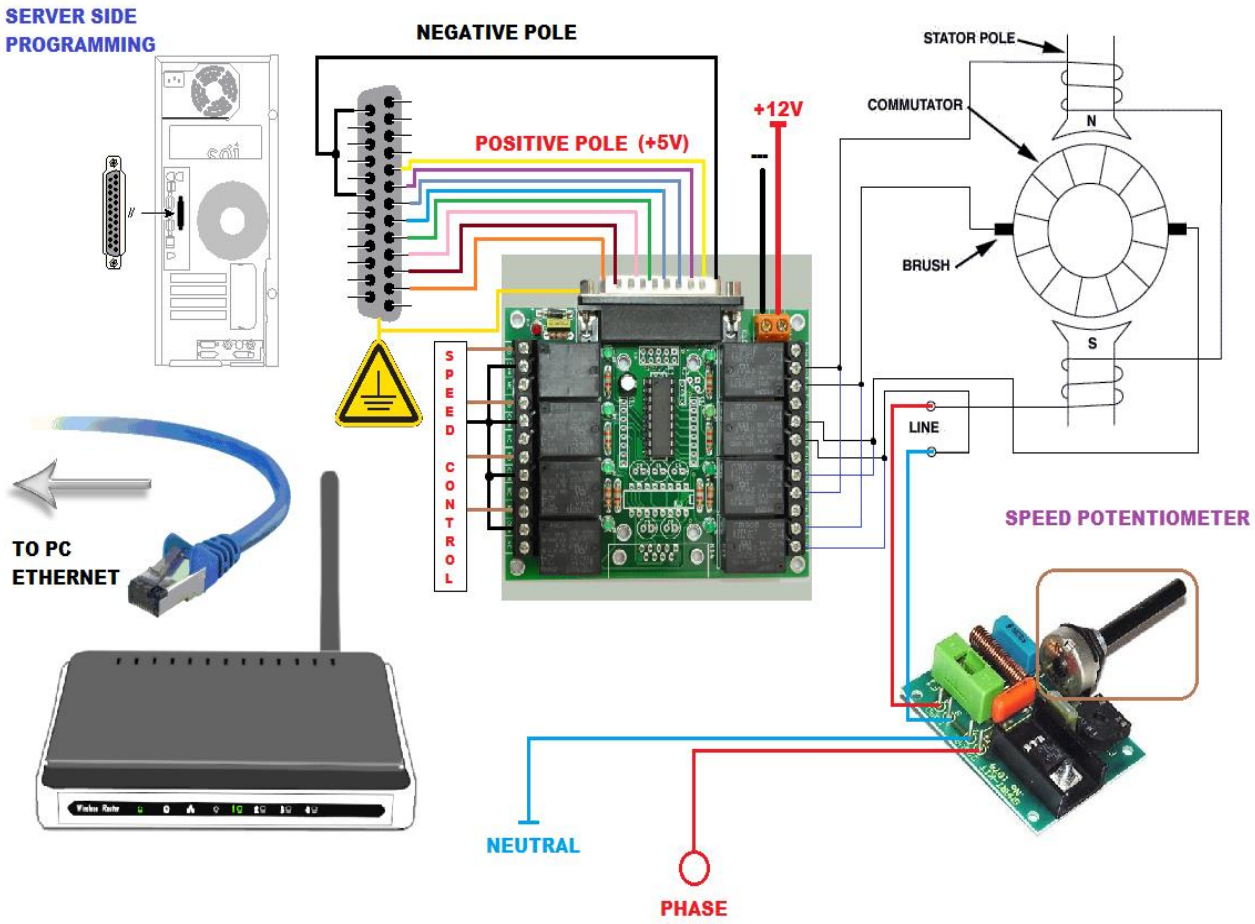
- Ελέγχουμε αν κολλήσαμε όλα τα εξαρτήματα, αν υπάρχει κάποια ψυχρή κόλληση κλπ.
- Ελέγχουμε τη θέση και τη φορά κάθε εξαρτήματος.
- Ελέγχουμε αν τοποθετήσαμε σωστά τις διόδους, τα led και τα ολοκληρωμένα κυκλώματα.
- Το κύκλωμα έχει σχεδιαστεί για να δουλεύει με την τάση που αναφέρεται στα σχέδια. Με διαφορετική τάση από την προτεινόμενη, εκτός του ότι δεν θα δώσει τα αποτελέσματα που περιμένουμε, υπάρχει και κίνδυνος καταστροφής κάποιου εξαρτήματος. Το ίδιο ισχύει και για την αντιστροφή της πολικότητας τροφοδοσίας.
- Καθαρίζουμε προσεχτικά την πλακέτα, είναι πιθανό τα κατάλοιπα σολντερίνης να δημιουργήσουν προβλήματα στη λειτουργία της πλακέτας.
- Ελέγχουμε τη σύνδεση των εξωτερικών συσκευών.
- Ελέγχουμε προσεκτικά όλες τις κολλήσεις και τις γειτονικές πίστες του τυπωμένου κυκλώματος.

Ηλεκτρική συσκευή

Όπως είπαμε μπορούμε με την συγκεκριμένη εφαρμογή να ελέγξουμε οποιαδήποτε ηλεκτρική συσκευή, αυτή τη φορά επιλέξαμε να ελέγξουμε έναν ηλεκτρικό κινητήρα έτσι ώστε να φανούν καλύτερα οι δυνατότητες της συσκευής μας. Ο ηλεκτρικός κινητήρας που επιλέξαμε λοιπόν είναι το δραπανοκατσάβιδο verto 50G505, το οποίο λειτουργεί στα 500W και αποτελεί έναν κινητήρα τύπου universal. Θα μιλήσουμε φυσικά για αυτούς τους κινητήρες αναλυτικότερα σε επόμενα κεφάλαια.

Εικόνα 1.9: Δραπανοκατσάβιδο verto 50G505.

1.2.2 κατασκευή και λειτουργία



Εικόνα 1.10: Αναλυτικό σχεδιάγραμμα της συνδεσμολογίας όλων των συσκευών που χρησιμοποιήθηκαν.

Η παραπάνω εικόνα αποτελεί μια εικονογράφηση-χαρτογράφηση ολόκληρου του κυκλώματος που φτιάξαμε, όπως βλέπουμε περιλαμβάνονται όλες οι συσκευές για τις οποίες μιλήσαμε παραπάνω και μπορούμε να δούμε τη σχέση που έχουν αυτές μεταξύ τους. Από το σχήμα λοιπόν του κυκλώματος μας μπορούμε να καταλάβουμε ότι όλες οι εντολές που δίνουμε ξεκινάνε από τον υπολογιστή, έτσι η παράλληλη θύρα του υπολογιστή μας συνδέεται κατευθείαν με το parallel port controller που χρησιμοποιήσαμε. Το parallel port controller το οποίο τροφοδοτείται ξεχωριστά με dc ρεύμα έχει πάνω του 8 relay, αυτά τα relay λοιπόν χρησιμεύουν στην ουσία ως ηλεκτρολογικοί διακόπτες στο σύστημα μας. Τέσσερα από αυτά τα relay χρησιμοποιούνται για της κίνηση του ηλεκτρικού μας κινητήρα, δυο για δεξιόστροφη κίνηση και δυο για αριστερόστροφη, όπως φαίνεται και στο σχήμα. Στο

ρυθμιστή στροφών μας έχουμε αντικαταστήσει το ποτενσιόμετρο του με ένα σύστημα τεσσάρων διαφορετικών ποτενσιόμετρων που ελέγχονται από τα τέσσερα υπολειπόμενα relay του parallel port controller, έτσι έχουμε τέσσερις διαφορετικές ταχύτητες στην ηλεκτρική μας μηχανή.

Από εκεί και πέρα για τη διαχείριση της συσκευής μέσω internet έχουμε συνδέσει ένα router στον υπολογιστή μας. Επίσης στο κομμάτι του προγραμματισμού(για το οποίο θα μιλήσουμε περισσότερο σε άλλη ενότητα) έχουμε εγκαταστήσει το απαραίτητο πρόγραμμα(xampp) το οποίο μετατρέπει τον υπολογιστή μας σε έναν εξυπηρετητή(server).Για τον προγραμματισμό της ιστιοσελίδας που είναι εγκατεστημένη στον server χρησιμοποιήσαμε διάφορες γλώσσες προγραμματισμού, από τις οποίες κυριότερες είναι η php(server side) και η html(client side).

Η λειτουργία λοιπόν της συσκευής έχει ως εξής, όπως είπαμε ο parallel port controller μετατρέπει το σήμα εξόδου της παράλληλης θύρας σε τάση ικανή να οπλίσει τα 8 relay μας και έτσι έχουμε τη διαχείριση του κινητήρα μας σε τέσσερις διαφορετικές καταστάσεις και τέσσερις διαφορετικές ταχύτητες. Από τον επιθυμητό πίνακα ελέγχου επιλέγουμε αρχικά την ταχύτητα περιστροφής και στη συνέχεια τη φορά περιστροφής. Αν για παράδειγμα θέλουμε ο κινητήρας μας να περιστραφεί αριστερόστροφα με τη χαμηλότερη ταχύτητα, πατάμε αρχικά το πλήκτρο 1 και στη συνέχεια το πλήκτρο left. Αυτό στέλνει ένα ψηφιακό δυαδικό σήμα με χαρακτηριστικό αριθμό 10011(το φέρων σήμα είναι στην ουσία το 00010011 αφού εμείς έχουμε 8 διαφορετικές καταστάσεις στο parallel port controller) που στο δεκαδικό σύστημα αντιστοιχεί στον αριθμό 19.Το σήμα αυτό δηλώνει πως σε κάθε λογικό 1 που βλέπουμε στο συνολικό δυαδικό αριθμό θα έχουμε και αντίστοιχα 5 volt στη θύρα εξόδου. Ο controller αποκωδικοποιεί αυτό το σήμα και το στέλνει στο αντίστοιχο ρελλέ οπλισμού, στη συγκεκριμένη περίπτωση θα οπλίσουν τα ρελλέ 1,2,5.Το ρελλέ 5 καθορίζει μια συγκεκριμένη τιμή αντίστασης, που είναι συνδεδεμένη στον ακροδέκτη normally open, η οποία με τη σειρά της καθορίζει το ρεύμα διέγερσης του triac που είναι εγκατεστημένο στο σύστημα αυξομείωσης στροφών και αυτό με τη σειρά του καθορίζει το ρεύμα διέγερσης του κινητήρα μας.

Το ζεύγος των ρελλέ 1 και 2 είναι υπεύθυνα για τη μανδάλωση των αντίστοιχων ζευγών καλωδίων που καθορίζουν την αντίστοιχη μαγνητική ροή για την αριστερή περιστροφή του κινητήρα. Αντίστοιχα τα ρελλέ 3 και 4 καθορίζουν την δεξιά περιστροφή ενώ τα ρελλέ 5, 6, 7, 8 είναι υπεύθυνα για την επιλογή ταχύτητας περιστροφής του κινητήρα.

1.3 Οι κώδικες που χρησιμοποιήθηκαν (web controlling)

Σε αυτή την ενότητα θα παραθέσουμε τα κομμάτια κώδικα που χρησιμοποιήθηκαν στην εφαρμογή και θα εξηγήσουμε την λειτουργία τους.

1.3.1 Βασικό πρόγραμμα (visual basic)

Το βασικό πρόγραμμα είναι στην ουσία μια κλάση. Βλέπουμε ότι στην αρχή δηλώνουμε το είδος των μεταβλητών, μετά καταχωρούμε τις τιμές τους και μετά δηλώνουμε τις μεθόδους εισόδου που θα αναγνωρίζει το πρόγραμμα για να υλοποιεί τις εντολές.

Στο επόμενο στάδιο ενσωματώνουμε στον κώδικα υπομονάδες οι οποίες διεξάγουν μια συγκεκριμένη λειτουργία (δεξιόστροφη περιστροφή, αριστερόστροφη περιστροφή, φρένο κλπ.).

Συνοδευτικά με το πρόγραμμα έχουμε ορίσει μια υπομονάδα (module1) η οποία περιέχει μια συνάρτηση που καλεί τη βιβλιοθήκη io.dll η οποία με τη σειρά της ενεργοποιεί κατάλληλα την παράλληλη θύρα.

Στη συνέχεια του προγράμματος υπάρχουν υπομονάδες οι οποίες είναι εμφωλευμένες μέσα σε χρονικά (timers).

Στο υποπρόγραμμα φρένου τσεκάρεται στην ουσία τι τιμή έχει η παράλληλη θύρα (port in) και ενεργοποιείται το σωστό timer.

Κώδικας visual basic

```
Public Class Form1
    Dim i, d As Integer
    Dim speech, dec As String
    Dim j, g, check As Byte

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        i = 0
        j = 0
        g = 0
        d = 0
        speech = "LOADING POINT"
        dec = "00000000"
        Call PortOut(888, 0)
    End Sub
    'I/O Inputs'
    Protected Overrides Function ProcessCmdKey(ByRef msg As
```

```
System.Windows.Forms.Message, _
                                                ByVal keyData As
System.Windows.Forms.Keys) _
                                                As Boolean

        If msg.WParam.ToInt32() = CInt(Keys.W) Then
            Timer1.Enabled = True
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.S) Then
            i = 0
            j = 0
            g = 0
            speech = "LOADING POINT"
            dec = "00000000"
            Call PortOut(888, 0)
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.A) Then
            g = 3
            PortOut(888, 0)
            Timer3.Enabled = True
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.D) Then
            g = 12
            PortOut(888, 0)
            Timer3.Enabled = True
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.D1) Then
            j = 16
            PortOut(888, j + g)
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.D2) Then
            j = 32
            PortOut(888, j + g)
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.D3) Then
            j = 64
            PortOut(888, j + g)
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.D4) Then
            j = 128
            PortOut(888, j + g)
            Return True
        ElseIf msg.WParam.ToInt32() = CInt(Keys.Escape) Then
            Me.Close()
            Return True
        End If
        Return MyBase.ProcessCmdKey(msg, keyData)
    End Function

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
        i = 0
        j = 0
        g = 0
        speech = "LOADING POINT"
        Timer1.Enabled = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        g = 3
```

```
        PortOut(888, 0)
        Timer3.Enabled = True
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button3.Click
        g = 12
        PortOut(888, 0)
        Timer3.Enabled = True
    End Sub
    'Υπομονάδα Αριστερής/Δεξιάς περιστροφής'
    Private Sub Timer3_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer3.Tick
        If g = 3 Then
            If d = 0 Then
                PortOut(888, 0)
            ElseIf d = 1 Then
                PortOut(888, 1)
            ElseIf d = 2 Then
                PortOut(888, j + g)
                d = 0
                Timer3.Enabled = False
            End If
        ElseIf g = 12 Then
            If d = 0 Then
                PortOut(888, 0)
            ElseIf d = 1 Then
                PortOut(888, 4)
            ElseIf d = 2 Then
                PortOut(888, j + g)
                d = 0
                Timer3.Enabled = False
            End If
        End If
        d = d + 1
    End Sub
    'Νεκρό Σημείο'
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button4.Click
        i = 0
        j = 0
        g = 0
        speech = "LOADING POINT"
        dec = "00000000"
        Call PortOut(888, 0)
    End Sub
    'Υπομονάδα Φρένου'
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        check = PortIn(888)
        If (check = 19) Or (check = 35) Or (check = 67) Or (check =
131) Then
            Timer4.Enabled = True
            Timer1.Enabled = False
        ElseIf (check = 28) Or (check = 44) Or (check = 76) Or (check
= 140) Then
            Timer5.Enabled = True
            Timer1.Enabled = False
        Else
            Timer1.Enabled = False
        End If
```

```
    End Sub

    Private Sub Timer4_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer4.Tick
        If i = 0 Then
            PortOut(888, 0)
            i = 1
        ElseIf i = 1 Then
            PortOut(888, 12)
            i = 2
        ElseIf i = 2 Then
            PortOut(888, 28)
            i = 3
        Else
            i = 0
            PortOut(888, 0)
            speech = "LOADING POINT"
            Timer4.Enabled = False
        End If
    End Sub

    Private Sub Timer5_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer5.Tick
        If i = 0 Then
            PortOut(888, 0)
            i = 1
        ElseIf i = 1 Then
            PortOut(888, 3)
            i = 2
        ElseIf i = 2 Then
            PortOut(888, 19)
            i = 3
        Else
            i = 0
            PortOut(888, 0)
            speech = "LOADING POINT"
            Timer5.Enabled = False
        End If
    End Sub
    'Πρώτη ταχύτητα'
    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button5.Click
        j = 16
        PortOut(888, j + g)
    End Sub
    'Δεύτερη ταχύτητα'
    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button6.Click
        j = 32
        PortOut(888, j + g)
    End Sub
    'Τρίτη ταχύτητα'
    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button7.Click
        j = 64
        PortOut(888, j + g)
    End Sub
    'Τέταρτη ταχύτητα'
    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button8.Click
        j = 128
```

```
        PortOut(888, j + g)
    End Sub
    'Ενσωματωμένο display κατάστασης'
    Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer2.Tick
        Call PortIn(888)
        check = PortIn(888)
        Dim thisDay As DateTime = DateTime.Today
        Dim dec As Integer = check

        Dim ss As String = Convert.ToString(dec, 2)

        If (check = 3) Or (check = 12) Then
            speech = "SELECT SPEED"
        ElseIf (check = 16) Then
            speech = "1st SPEED"
        ElseIf (check = 32) Then
            speech = "2nd SPEED"
        ElseIf (check = 64) Then
            speech = "3rd SPEED"
        ElseIf (check = 128) Then
            speech = "4th SPEED"
        ElseIf (check = 19) Or (check = 35) Or (check = 67) Or (check
= 131) Then
            speech = "LEFT TURN"
        ElseIf (check = 28) Or (check = 44) Or (check = 76) Or (check
= 140) Then
            speech = "RIGHT TURN"
        End If
        TextBox1.Text = "State: " & speech & "    " & "Date: " &
thisDay & "    " & "Status: " & ss
    End Sub
    'Μήνυμα εξόδου προγράμματος'
    Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        If MessageBox.Show("Are you sure you want to quit?", "Are you
sure?", MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then
            e.Cancel = True
        End If
    End Sub
End Class
```

module 1

```
'Υπομονάδα διευθυντοποίησης και σύζευξης θυρών με την χρήση της
βιβλιοθήκης "io.dll"
Module Module1
    Public Declare Sub PortOut Lib "io.dll" (ByVal Port As Integer,
ByVal Value As Byte)
    Public Declare Function PortIn Lib "io.dll" (ByVal Port As
Integer) As Byte
End Module
```

1.3.2 php, html

Η βασική μεθοδολογία εδώ είναι να δηλώσουμε ένα σύνολο συναρτήσεων τις οποίες διέπουν κάποιες λειτουργίες και στη συνέχεια να επικαλούμαστε αυτές τις συναρτήσεις για να υλοποιήσουμε το κατάλληλο αποτέλεσμα.
Παραδείγματα αυτών των συναρτήσεων είναι τα παρακάτω:
portstatus: Συνάρτηση που ενημερώνει την ετικέτα.
Portcontrol: Συνάρτηση που δέχεται παραμέτρους υλοποιεί το αποτέλεσμα και ενημερώνει το portstatus.
Cursorkeydown: Είναι η συνάρτηση που αναγνωρίζει τα πλήκτρα που πιέζονται στο πληκτρολόγιο.
Timeout: Υπεύθυνη για φρενάρισμα.

Κώδικας php

```
<?php
      require("Sajax.php");

      function portstatus() {
              return  "State: " . file_get_contents("status.txt") ."
Date: " . date("dS-m-Y  ") ."Status: " . shell_exec("portcontrol.exe
LPT1DATA read print bin");
 }

        function portcontrol($x, $y) {
        if (($x >= 0) && ($x < 8)) {
             if ($y == 0)
               shell_exec("portcontrol.exe LPT1DATA read resetbit " .
$x . " write");
             elseif ($y == 1)
               shell_exec("portcontrol.exe LPT1DATA read setbit " .
$x . " write");
      elseif ($y == 2)  shell_exec("break.exe");      elseif ($y ==
3)  shell_exec("left.exe");   elseif ($y == 4)
shell_exec("right.exe");}
      $controlstring=file_get_contents("status.txt");
                if ($controlstring==FALSE) {$controlstring = 'Loading
Status';
 };     $state=shell_exec("lptout.exe read 888") ;        if
($state==0)
                {
               $controlstring='--LOADING POINT--';
                }
         elseif(($state==3) or ($state==12))
                {
                 $controlstring='--SELECT SPEED--';
 }        elseif(($state==19) or ($state==35) or ($state==67) or
($state==131))
                {
                  $controlstring='--LEFT TURN--';
 } elseif(($state==28) or ($state==44) or ($state==76) or
($state==140))
```

```
                {
                  $controlstring='--RIGHT TURN--';
 } elseif($state==16)
                {
                 $controlstring='--1st SPEED--';
 } elseif($state==32)
                {
                 $controlstring='2nd SPEED';
 } elseif($state==64)
                {
                 $controlstring='3rd SPEED';
 } elseif($state==128)
                {
                 $controlstring='4th SPEED';
 } // write file
                $filehandle=fopen ("status.txt",'w');
                fwrite($filehandle, $controlstring);
            fclose($filehandle);
                return portstatus();
        }

      sajax_init();
      // $sajax_debug_mode = 1;
      sajax_export("portstatus");
        sajax_export("portcontrol");
      sajax_handle_client_request();

?>
<html>
<head>
      <title>Ultimate Range Controller</title>
      <script>
      <?php
      sajax_show_javascript();
      ?>

      function do_portstatus_cb(z) {
            // update status field in form
            document.getElementById("status").value = z;
      }

      function do_portstatus() {
            x_portstatus(do_portstatus_cb);
            setTimeout('do_portstatus();',1000); // executes the next
data query in every n milliseconds
      }

      function do_portcontrol_cb(z) {
            // update status field in form
            document.getElementById("status").value = z;
        }

      function do_portcontrol(bit,value) {
            x_portcontrol(bit,value,do_portcontrol_cb);
      }

      </script>
<script type="text/javascript">
function color()
{
document.getElementById('status').style.background='yellow';
```

```
}
function CursorKeyDown(e)
{
   if (!e) e=window.event;
  switch(e.keyCode)
  {
  case 87:
    // W
document.getElementById('action1').focus();
    break;
  case 65:
    // A
document.getElementById('action2').focus();
    break;
  case 68:
    // D
document.getElementById('action3').focus();
    break;
  case 83:
    // S
document.getElementById('action4').focus();
    break;
  case 49:
    // 1
document.getElementById('action5').focus();
    break;
  case 50:
    // 2
document.getElementById('action6').focus();
    break;
  case 51:
    // 3
document.getElementById('action7').focus();
    break;
  case 52:
    // 4
document.getElementById('action8').focus();
    break;
  }
}
function timeout()
{
document.getElementById('status').focus();
do_portcontrol(0,2);
return false;
}
</script>
</head>
<body bgcolor="steelblue" onload="color();">
<?php shell_exec("lptout write 888 0"); $control='...Loading...';
$file=fopen ("status.txt",'w');  fwrite($file,$control);
fclose($file); ?>
<SCRIPT LANGUAGE="JavaScript">
<!--
do_portstatus();
// -->
</SCRIPT>
<center>
<font color="red" size="10"><b>Control Electric Motor via Internet
Using Parallel Port</b></font>
<br>
```

```
<br>
<P><input type="button" name="status" id="status"
value="...Loading..."  size="60" onkeydown="CursorKeyDown(event);">
<br>
<br>
<table border="0" cellpadding="0" cellspacing="0">
<tr>
<td></td>
<td><input name="r2c1" type="button" value="       BREAK       "
id="action1" onclick="timeout();" onkeydown="CursorKeyDown(event);"
onfocus="timeout();"></td>
<td></td>
</tr>
<tr>
<td><input name="r1c1" type="button" value="  LEFT   " id="action2"
onclick="document.getElementById('status').focus();do_portcontrol(0,3
);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="document.getElementById('status').focus();do_portcontrol(0,3
);return false;"></td>
<td></td>
<td><input name="r1c3" type="button" value=" RIGHT  " id="action3"
onclick="document.getElementById('status').focus();do_portcontrol(0,4
);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="document.getElementById('status').focus();do_portcontrol(0,4
);return false;"></td>
</tr>
<tr>
<td></td><td><input name="r1c2" type="button" Value="       RESET
" id="action4"
onclick="do_portcontrol(0,0);do_portcontrol(1,0);do_portcontrol(2,0);
do_portcontrol(3,0);do_portcontrol(4,0);do_portcontrol(5,0);do_portco
ntrol(6,0);do_portcontrol(7,0);return false;"
onkeydown="CursorKeyDown(event);"
onfocus="do_portcontrol(0,0);do_portcontrol(1,0);do_portcontrol(2,0);
do_portcontrol(3,0);do_portcontrol(4,0);do_portcontrol(5,0);do_portco
ntrol(6,0);do_portcontrol(7,0);return false;"
autofocus></td><td></td>
</tr>
</table>
<br>
<font color="yellow"><b><h1>***Speed selection***</h1></b></font>
<form><input name="r2c2" type="button" value="1" id="action5"
onclick="do_portcontrol(4,1);do_portcontrol(5,0);do_portcontrol(6,0);
do_portcontrol(7,0);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="do_portcontrol(4,1);do_portcontrol(5,0);do_portcontrol(6,0);
do_portcontrol(7,0);return false;"></form>
<form><input name="r2c3" type="button" value="2" id="action6"
onclick="do_portcontrol(4,0);do_portcontrol(5,1);do_portcontrol(6,0);
do_portcontrol(7,0);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="do_portcontrol(4,0);do_portcontrol(5,1);do_portcontrol(6,0);
do_portcontrol(7,0);return false;"></form>
<form><input name="r2c4" type="button" value="3" id="action7"
onclick="do_portcontrol(4,0);do_portcontrol(5,0);do_portcontrol(6,1);
do_portcontrol(7,0);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="do_portcontrol(4,0);do_portcontrol(5,0);do_portcontrol(6,1);
do_portcontrol(7,0);return false;"></form>
<form><input name="r2c5" type="button" value="4" id="action8"
onclick="do_portcontrol(4,0);do_portcontrol(5,0);do_portcontrol(6,0);
do_portcontrol(7,1);return false;"  onkeydown="CursorKeyDown(event);"
onfocus="do_portcontrol(4,0);do_portcontrol(5,0);do_portcontrol(6,0);
do_portcontrol(7,1);return false;"></form>
```

```
<br>
<font color="purple" size="3"><b><i>
//Press W for Break.<br>
//Press S to Reset.<br>
//Press A to move Left.<br>
//Press D to move Right.<br>
//Press 1 for 1st speed.<br>
//Press 2 for 2nd speed.<br>
//Press 3 for 3rd speed.<br>
//Press 4 for 4th speed.</i></b></font>
</center>
<br>
<br>
</body>
</html>
```

κώδικας ajax

```
<?php
if (!isset($SAJAX_INCLUDED)) {

	/*
	 * GLOBALS AND DEFAULTS
	 *
	 */
	$GLOBALS['sajax_version'] = '0.12';
	$GLOBALS['sajax_debug_mode'] = 0;
	$GLOBALS['sajax_export_list'] = array();
	$GLOBALS['sajax_request_type'] = 'GET';
	$GLOBALS['sajax_remote_uri'] = '';
	$GLOBALS['sajax_failure_redirect'] = '';

	/*
	 * CODE
	 *
	 */

	//
	// Initialize the Sajax library.
	//
	function sajax_init() {
	}

	//
	// Helper function to return the script's own URI.
	//
	function sajax_get_my_uri() {
		return $_SERVER["REQUEST_URI"];
	}
	$sajax_remote_uri = sajax_get_my_uri();

	//
	// Helper function to return an eval()-usable representation
	// of an object in JavaScript.
	//
	function sajax_get_js_repr($value) {
```

```
            $type = gettype($value);

            if ($type == "boolean") {
                  return ($value) ? "Boolean(true)" :
"Boolean(false)";
            }
            elseif ($type == "integer") {
                  return "parseInt($value)";
            }
            elseif ($type == "double") {
                  return "parseFloat($value)";
            }
            elseif ($type == "array" || $type == "object" ) {
                  //
                  // XXX Arrays with non-numeric indices are not
                  // permitted according to ECMAScript, yet everyone
                  // uses them.. We'll use an object.
                  //
                  $s = "{ ";
                  if ($type == "object") {
                        $value = get_object_vars($value);
                  }
                  foreach ($value as $k=>$v) {
                        $esc_key = sajax_esc($k);
                        if (is_numeric($k))
                              $s .= "$k: " . sajax_get_js_repr($v) .
", ";
                        else
                              $s .= "\"$esc_key\": " .
sajax_get_js_repr($v) . ", ";
                  }
                  if (count($value))
                        $s = substr($s, 0, -2);
                  return $s . " }";
            }
            else {
                  $esc_val = sajax_esc($value);
                  $s = "'$esc_val'";
                  return $s;
            }
      }

      function sajax_handle_client_request() {
            global $sajax_export_list;

            $mode = "";

            if (! empty($_GET["rs"]))
                  $mode = "get";

            if (!empty($_POST["rs"]))
                  $mode = "post";

            if (empty($mode))
                  return;

            $target = "";

            if ($mode == "get") {
                  // Bust cache in the head
                  header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
```

```
// Date in the past
                  header ("Last-Modified: " . gmdate("D, d M Y
H:i:s") . " GMT");
                  // always modified
                  header ("Cache-Control: no-cache, must-
revalidate");  // HTTP/1.1
                  header ("Pragma: no-cache");
// HTTP/1.0
                  $func_name = $_GET["rs"];
                  if (! empty($_GET["rsargs"]))
                        $args = $_GET["rsargs"];
                  else
                        $args = array();
            }
            else {
                  $func_name = $_POST["rs"];
                  if (! empty($_POST["rsargs"]))
                        $args = $_POST["rsargs"];
                  else
                        $args = array();
            }

            if (! in_array($func_name, $sajax_export_list))
                  echo "-:$func_name not callable";
            else {
                  echo "+:";
                  $result = call_user_func_array($func_name, $args);
                  echo "var res = " .
trim(sajax_get_js_repr($result)) . "; res;";
            }
            exit;
      }

      function sajax_get_common_js() {
            global $sajax_debug_mode;
            global $sajax_request_type;
            global $sajax_remote_uri;
            global $sajax_failure_redirect;

            $t = strtoupper($sajax_request_type);
            if ($t != "" && $t != "GET" && $t != "POST")
                  return "// Invalid type: $t.. \n\n";

            ob_start();
            ?>

            // remote scripting library
            // (c) copyright 2005 modernmethod, inc
            var sajax_debug_mode = <?php echo $sajax_debug_mode ?
"true" : "false"; ?>;
            var sajax_request_type = "<?php echo $t; ?>";
            var sajax_target_id = "";
            var sajax_failure_redirect = "<?php echo
$sajax_failure_redirect; ?>";

            function sajax_debug(text) {
                  if (sajax_debug_mode)
                        alert(text);
            }

            function sajax_init_object() {
```

```
                  sajax_debug("sajax_init_object() called..")

                  var A;

                  var msxmlhttp = new Array(
                        'Msxml2.XMLHTTP.5.0',
                        'Msxml2.XMLHTTP.4.0',
                        'Msxml2.XMLHTTP.3.0',
                        'Msxml2.XMLHTTP',
                        'Microsoft.XMLHTTP');
                  for (var i = 0; i < msxmlhttp.length; i++) {
                        try {
                              A = new ActiveXObject(msxmlhttp[i]);
                        } catch (e) {
                              A = null;
                        }
                  }

                  if(!A && typeof XMLHttpRequest != "undefined")
                        A = new XMLHttpRequest();
                  if (!A)
                        sajax_debug("Could not create connection
object.");
                  return A;
            }

            var sajax_requests = new Array();

            function sajax_cancel() {
                  for (var i = 0; i < sajax_requests.length; i++)
                        sajax_requests[i].abort();
            }

            function sajax_do_call(func_name, args) {
                  var i, x, n;
                  var uri;
                  var post_data;
                  var target_id;

                  sajax_debug("in sajax_do_call().." +
sajax_request_type + "/" + sajax_target_id);
                  target_id = sajax_target_id;
                  if (typeof(sajax_request_type) == "undefined" ||
sajax_request_type == "")
                        sajax_request_type = "GET";

                  uri = "<?php echo $sajax_remote_uri; ?>";
                  if (sajax_request_type == "GET") {

                        if (uri.indexOf("?") == -1)
                              uri += "?rs=" + escape(func_name);
                        else
                              uri += "&rs=" + escape(func_name);
                        uri += "&rst=" + escape(sajax_target_id);
                        uri += "&rsrnd=" + new Date().getTime();

                        for (i = 0; i < args.length-1; i++)
                              uri += "&rsargs[]=" + escape(args[i]);

                        post_data = null;
                  }
```

```
                  else if (sajax_request_type == "POST") {
                        post_data = "rs=" + escape(func_name);
                        post_data += "&rst=" +
escape(sajax_target_id);
                        post_data += "&rsrnd=" + new
Date().getTime();

                        for (i = 0; i < args.length-1; i++)
                              post_data = post_data + "&rsargs[]=" +
escape(args[i]);
                  }
                  else {
                        alert("Illegal request type: " +
sajax_request_type);
                  }

                  x = sajax_init_object();
                  if (x == null) {
                        if (sajax_failure_redirect != "") {
                              location.href = sajax_failure_redirect;
                              return false;
                        } else {
                              sajax_debug("NULL sajax object for user
agent:\n" + navigator.userAgent);
                              return false;
                        }
                  } else {
                        x.open(sajax_request_type, uri, true);
                        // window.open(uri);

                        sajax_requests[sajax_requests.length] = x;

                        if (sajax_request_type == "POST") {
                              x.setRequestHeader("Method", "POST " +
uri + " HTTP/1.1");
                              x.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
                        }

                        x.onreadystatechange = function() {
                              if (x.readyState != 4)
                                    return;

                              sajax_debug("received " +
x.responseText);

                              var status;
                              var data;
                              var txt =
x.responseText.replace(/^\s*|\s*$/g,"");
                              status = txt.charAt(0);
                              data = txt.substring(2);

                              if (status == "") {
                                    // let's just assume this is a
pre-response bailout and let it slide for now
                              } else if (status == "-")
                                    alert("Error: " + data);
                              else {
                                    if (target_id != "")
```

```
	document.getElementById(target_id).innerHTML = eval(data);
                                    else {
                                          try {
                                                var callback;
                                                var extra_data =
false;
                                                if (typeof
args[args.length-1] == "object") {
                                                      callback =
args[args.length-1].callback;
                                                      extra_data =
args[args.length-1].extra_data;
                                                } else {
                                                      callback =
args[args.length-1];
                                                }
                                                callback(eval(data),
extra_data);
                                          } catch (e) {
                                                sajax_debug("Caught
error " + e + ": Could not eval " + data );
                                          }
                                    }
                              }
                        }
                  }

                  sajax_debug(func_name + " uri = " + uri + "/post =
" + post_data);
                  x.send(post_data);
                  sajax_debug(func_name + " waiting..");
                  delete x;
                  return true;
            }

            <?php
            $html = ob_get_contents();
            ob_end_clean();
            return $html;
      }

      function sajax_show_common_js() {
            echo sajax_get_common_js();
      }

      // javascript escape a value
      function sajax_esc($val)
      {
            $val = str_replace("\\", "\\\\", $val);
            $val = str_replace("\r", "\\r", $val);
            $val = str_replace("\n", "\\n", $val);
            $val = str_replace("'", "\\'", $val);
            return str_replace('"', '\\"', $val);
      }

      function sajax_get_one_stub($func_name) {
            ob_start();
            ?>

            // wrapper for <?php echo $func_name; ?>
```

```
		function x_<?php echo $func_name; ?>() {
			sajax_do_call("<?php echo $func_name; ?>",
				x_<?php echo $func_name; ?>.arguments);
		}

		<?php
		$html = ob_get_contents();
		ob_end_clean();
		return $html;
	}

	function sajax_show_one_stub($func_name) {
		echo sajax_get_one_stub($func_name);
	}

	function sajax_export() {
		global $sajax_export_list;

		$n = func_num_args();
		for ($i = 0; $i < $n; $i++) {
			$sajax_export_list[] = func_get_arg($i);
		}
	}

	$sajax_js_has_been_shown = 0;
	function sajax_get_javascript()
	{
		global $sajax_js_has_been_shown;
		global $sajax_export_list;

		$html = "";
		if (! $sajax_js_has_been_shown) {
			$html .= sajax_get_common_js();
			$sajax_js_has_been_shown = 1;
		}
		foreach ($sajax_export_list as $func) {
			$html .= sajax_get_one_stub($func);
		}
		return $html;
	}

	function sajax_show_javascript()
	{
		echo sajax_get_javascript();
	}


	$SAJAX_INCLUDED = 1;
}
?>
```

1.3.3 web controlling

Όσο αφορά τώρα τη διαχείριση μέσω διαδικτύου είναι σημαντικό εκτός από τους κώδικες που χρησιμοποιήθηκαν να γνωρίζουμε και κάποια πράγματα σχετικά με το port forwarding (άνοιγμα θύρας στο router) και τον εξυπηρετητή apache.

Αρχικά λοιπόν θα πρέπει να γνωρίζουμε ότι ο apache είναι ένας εξυπηρετητής παγκόσμιου ιστού ο οποίος θα πρέπει να εγκατασταθεί στον υπολογιστή μας (server) μαζί με κάποια αναγκαία προγράμματα (xamp). Αυτό θα μας βοηθήσει να μπορούμε να επικοινωνούμε με τον server μας μέσω διαδικτύου και έτσι να δίνουμε εντολές στο κύκλωμά μας. Η εγκατάσταση του apache και του xamp είναι σχετικά απλή και καλύπτεται εύκολα με μια απλή αναζήτηση στο internet, γι αυτό το λόγο δεν θα ασχοληθούμε περαιτέρω.

Όταν λέμε Port Forwarding εννοούμε τη διαδικασία που πρέπει να γίνει για να είναι ένας υπολογιστής που βρίσκεται πίσω από έναν δρομολογητή (Router) προσβάσιμος από άλλους υπολογιστές στο διαδίκτυο, αυτό επιτρέπει σε εφαρμογές στο δίκτυό μας να είναι ανοιχτές προς το Internet .Όπως καταλαβαίνουμε λοιπόν, το άνοιγμα θύρας στο router μας είναι πολύ σημαντικό για να έχουμε web controlling στο κύκλωμά μας. Το port forwarding λοιπόν σχετίζεται πάνω από όλα με τη γνώση του router μας και με το πόσο καλά ξέρουμε να το χρησιμοποιούμε (σε κάθε διαφορετικού τύπου router η διαδικασία μπορεί να διαφέρει).Στη βιβλιογραφία λοιπόν αυτής της πτυχιακής εργασίας μπορεί κανείς να βρει συνδέσμους οι οποίοι χρησίμευσαν ως tutorials σε μένα αλλά και τον τρόπο να κάνει ο ίδιος port forwarding σε όλα σχεδόν τα διαφορετικά router.

ΚΕΦΑΛΑΙΟ 2

2.1 Βασικά ψηφιακά συστήματα

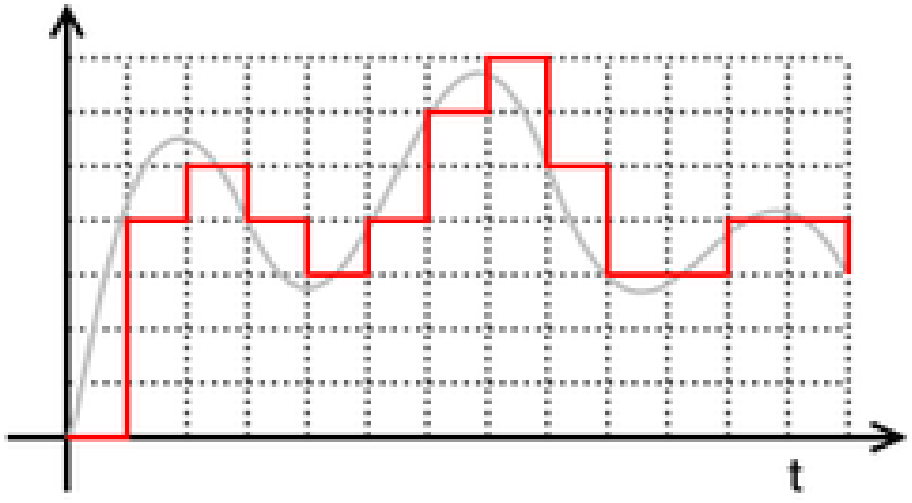
2.1.1 εισαγωγή

Καθώς στην εφαρμογή μας θα χρησιμοποιήσουμε μικροελεγκτές, ηλεκτρονικούς υπολογιστές, προγραμματισμό κ.α., είναι αναγκαίο να κάνουμε μια εισαγωγή στα ψηφιακά συστήματα. Αρχικά θα πρέπει να ξέρει κανείς ότι ψηφιακό σύστημα λέμε ένα σύστημα που δέχεται ψηφιακά δεδομένα από τα οποία παράγει νέα ψηφιακά δεδομένα σύμφωνα με κανόνες που, υλοποιούν τις βασικές πράξεις της λογικής, καθώς και κάθε πολύπλοκη σύνθεση αυτών. Στο κεφάλαιο αυτό θα ασχοληθούμε με έννοιες όπως είναι τα αναλογικά και ψηφιακά μεγέθη, οι ψηφιακές(δυαδικές) πληροφορίες, τι σημαίνει ψηφιακό σύστημα και ποια είναι η λειτουργία του, την άλγεβρα boole, τις λογικές πύλες κ.α.

2.1.2 Ψηφιακά και αναλογικά μεγέθη

Αναλογικά λέμε ότι είναι τα μεγέθη που οι καταστάσεις τους μπορούν να πάρουν συνεχείς τιμές και σε κάθε τιμή του μεγέθους αντιστοιχεί μία πληροφορία. Όλα τα μεγέθη της κλασσικής φυσικής είναι αναλογικά. Ένα καλό παράδειγμα αναλογικού μεγέθους είναι η ταχύτητα ενός αυτκινήτου, το αυτοκίνητο ξεκινάει από 0 χιλιόμετρα ανά ώρα και με τελική ταχύτητα τα 300 χλμ μπορεί να πάρει όλες τις ενδιάμεσες τιμές.

Ψηφιακά είναι τα μεγέθη που οι καταστάσεις τους παίρνουν μόνο διακριτές τιμές. Παραδείγματα ψηφιακών μεγεθών είναι οι ηλεκτρολογικοί διακόπτες που μπορούν να πάρουν μόνο δυο διακριτές τιμές(ανοικτός ή κλειστός), οι φωτεινοί σηματοδότες(πράσινο, κίτρινο, κόκκινο), το σκορ σε έναν αγώνα μπάσκετ, κ.α.



Εικόνα 2.1: Διάγραμμα ενός ψηφιακού και ενός αναλογικού μεγέθους.

2.1.3 Ψηφιακά σήματα και δυαδική κωδικοποίηση

Όπως είπαμε όλα τα μεγέθη στη φύση είναι αναλογικά, έτσι όταν σε κάποιες εφαρμογές υπάρχει η ανάγκη να χρησιμοποιήσουμε συσκευές που λειτουργούν με ψηφιακό σήμα, θα πρέπει να γίνει η μετατροπή των αναλογικών μεγεθών σε ψηφιακά. Για να μεταφέρει ένα ψηφιακό σήμα μια πληροφορία ενός αναλογικού μεγέθους, το μέγεθος αυτό πρέπει να διαμορφωθεί έτσι ώστε να πάρει τη μορφή διακριτών σταθμίδων. Για παράδειγμα εάν το μέγεθος που επεξεργαζόμαστε είναι η τάση του ηλεκτρικού ρεύματος από το 0V μέχρι τα 15V τότε μπορούμε να επιλέξουμε να την χωρίσουμε σε 16 στάθμες(0V, 1V, 2V, ..., 15V), το σήμα μας τότε θα είναι ένα 16δικο σήμα. Μπορούμε από το παραπάνω παράδειγμα να καταλάβουμε ότι κάθε σήμα που χωρίζεται σε N στάθμες, σε κάθε μία από τις οποίες αντιστοιχεί και μία πληροφορία, ονομάζεται N–αδικό σήμα.

Σήμερα το πιο διαδεδομένο είδος σήματος που χρησιμοποιείται σε όλα τα συστήματα είναι το δυαδικό σύστημα. Οι δυο καταστάσεις του δυαδικού σήματος είναι οι 0 και 1 και αντιστοιχούν στις περισσότερες εφαρμογές στα 0V και στα 5V, κάθε μια από τις δυο αυτές είναι 1 bit(binary digit). Ορίζεται δηλαδή ότι 1 bit πληροφορίας παριστά μία από τις δύο δυνατές καταστάσεις του δυαδικού σήματος. Όταν οι πληροφορίες που θέλουμε να αναπαραστήσουμε είναι περισσότερες από τις παραστάσεις του σήματός μας, τότε θα πρέπει να συνδυάσουμε πολλά σήματα μαζί. Με βάση τα παραπάνω 1 δυαδικό ψηφίο μπορεί να κωδικοποιήσει 2 καταστάσεις, 2 δυαδικά ψηφία μπορούν να κωδικοποιήσουν 2^2 καταστάσεις, 3 δυαδικά ψηφία μπορούν να κωδικοποιήσουν 2^3 καταστάσεις και ν δυαδικά ψηφία μπορούν να κωδικοποιήσουν 2^v καταστάσεις. Καθώς όπως γίνεται κατανοητό πολλές φορές χρειαζόμαστε ποσότητες πολλαπλάσιες του δυαδικού ψηφίου(bit), υπάρχουν τα πολλαπλάσια του, τα οποία είναι τα εξής:

- Τετράδα (nibble) = ποσότητα τεσσάρων (2^2) δυαδικών ψηφίων
- Byte (οκτάδα) = ποσότητα οκτώ (2^3) δυαδικών ψηφίων
- Kilobyte - KByte (KB) = ποσότητα 1024 (2^{10}) Bytes
- Megabyte - MByte (MB) = ποσότητα 1024 (2^{10}) KBytes = 220 Bytes
- Gigabyte - GByte (GB) = ποσότητα 1024 (2^{10}) MBytes = 230 Bytes
- Terabyte - TByte (TB) = ποσότητα 1024 (2^{10}) GByte = 240 Bytes

2.1.4 μετατροπές και πράξεις στο δυαδικό σύστημα

Για να έχουμε πλήρη κατανόηση του δυαδικού σήματος, της μνήμης και της λειτουργίας του υπολογιστή, των προγραμμάτων και των ψηφιακών συσκευών γενικότερα, είναι σημαντικό να μπορούμε να μετατρέψουμε αριθμούς του δυαδικού συστήματος στους αντίστοιχους του δεκαδικού και το αντίστροφο, καθώς και να είμαστε σε θέση να τελέσουμε κάποιες βασικές πράξεις στο δυαδικό σύστημα.

Μετατροπή αριθμού του δυαδικού συστήματος σε αριθμό του δεκαδικού συστήματος

Η μετατροπή ενός δυαδικού αριθμού σε δεκαδικό είναι σχετικά απλή, στην ουσία ο δεκαδικός αριθμός ισούται με το άθροισμα των δυνάμεων του δυο επί των ψηφίων του αντίστοιχου αριθμού του δυαδικού συστήματος. Έστω ότι έχουμε για παράδειγμα τον δυαδικό αριθμό 111001, για την μετατροπή του σε δεκαδικό θα πρέπει να τελέσουμε τις παρακάτω πράξεις:

$$\mathbf{1}\cdot 2^5+\mathbf{1}\cdot 2^4+\mathbf{1}\cdot 2^3+\mathbf{0}\cdot 2^2+\mathbf{0}\cdot 2^1+\mathbf{1}\cdot 2^0= 57$$

όπως βλέπουμε ο δυαδικός αριθμός 111001 αντιστοιχεί στον δεκαδικό 57.Από τα αριστερά προς τα δεξιά πολλαπλασιάσαμε το κάθε ψηφίο του δυαδικού αριθμού με την αντίστοιχη δύναμη του δυο(από το 0 μέχρι να τελειώσουν τα δεκαδικά ψηφία αυξάνοντας τη δύναμη κατά ένα).

Αν έπρεπε να βγάλουμε έναν μαθηματικό τύπο για την παραπάνω εφαρμογή αυτός θα ήταν ο εξής:
Για έναν αριθμό στο δυαδικό σύστημα με n αριθμό ψηφίων ισχύει:

$$\textit{αριθμός στο δυαδικό σύστημα} = \Psi*2^{n-1} + \Psi*2^{n-2} + \ldots + \Psi*2^1 + \Psi*2^0 = \textit{αριθμός στο δεκαδικό σύστημα}$$

όπου Ψ είναι ψηφίο του δυαδικού συστήματος, είτε 1 είτε 0.

Μετατροπή αριθμού του δεκαδικού συστήματος σε αριθμό του δυαδικού συστήματος

Γενικά για να μετατρέψουμε έναν αριθμό του δεκαδικού συστήματος στον αντίστοιχο του δυαδικού, θα πρέπει κάνουμε διαδοχικές διαιρέσεις με το 2, οι διαιρέσεις γίνονται στον αρχικό αριθμό και στο πηλίκο της προηγούμενης διαίρεσης και σταματούν όταν πάρουμε πηλίκο 0 .Στην συνέχεια τοποθετούμε τα υπόλοιπα που βρήκαμε στην σειρά ξεκινώντας από το τελευταίο προς το πρώτο και έτσι

σχηματίζεται ο δυαδικός του αριθμός.

Αν για παράδειγμα έχουμε τον αριθμό 187 τότε θα πρέπει να πράξουμε ως εξής:

187/2=93 και υπόλοιπο 1

93/2=46 και υπόλοιπο 1

46/2=23 και υπόλοιπο 0

23/2=11 και υπόλοιπο 1

11/2=5 και υπόλοιπο 1

5/2=2 και υπόλοιπο 1

2/2=1 και υπόλοιπο 0

1/2=0 και υπόλοιπο 1

Άρα έχουμε τον δυαδικό αριθμό 10111011.

αριθμητικές πράξεις στο δυαδικό σύστημα

Οι αριθμητικές πράξεις στο δυαδικό σύστημα γενικά ακολουθούν την ίδια λογική που ακολουθούν και στο δεκαδικό σύστημα. Ακολουθούν κάποια παραδείγματα για καλύτερη κατανόηση.

Παράδειγμα 1: πρόσθεση των αριθμών 1011 και 111

κρατούμενο		1	1	1
		1	0	1	1
	+		1	1	1
άθροισμα	1	0	0	1	0

παράδειγμα 2: αφαίρεση του αριθμού 111 από τον αριθμό 1011

κρατούμενο		-1	-1	-1
		1	0	1	1
	-		1	1	1
διάφορα		0	0	1	0

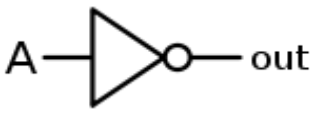
παράδειγμα 3: πολλαπλασιασμός του αριθμού 1100 με τον αριθμό 101

```
      1  1  0  0
    x    1  0  1
    ------------
      1  1  0  0
   0  0  0  0
1  1  0  0
-------------------
1  1  1  1  0  0
```

2.1.5 Λογικές πύλες

Μία λογική πύλη είναι ένα ηλεκτρονικό κύκλωμα το οποίο πραγματοποιεί μία λογική πράξη στις εισόδους της και παράγει μία έξοδο. Οι λογικές πύλες χρησιμοποιούνται στα ψηφιακά κυκλώματα για την εκτέλεση των τριών βασικών πράξεων της άλγεβρας boole(and, or, not) και όχι μόνο. Με βάση τα παραπάνω είναι λογικό οι είσοδοι και οι έξοδοι να παίρνουν αληθείς ή ψευδείς τιμές οι οποίες αναπαρίστανται με 0 ή 1 αντίστοιχα. Συνήθως στα ηλεκτρονικά κυκλώματα το 0 αντιστοιχεί στη σχεδόν μηδενική τάση και το 1 σε τάση που έχει ξεπεράσει τα 5V.

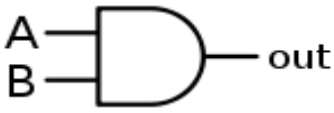
NOT



Η πύλη NOT έχει μόνο μία είσοδο και δίνει μόνο μία έξοδο, ενώ ο πίνακας αληθείας της είναι ο εξής:

Είσοδος	Έξοδος
0	1
1	0

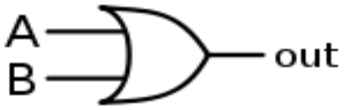
AND



Η πύλη AND εκτελεί την λογική πράξη AND μεταξύ των εισόδων της, η οποία στην άλγεβρα Boole συμβολίζεται με το σύμβολο (*).Οι πύλες AND κατασκευάζονται και με περισσότερες από δύο εισόδους και η έξοδος της AND δίνει λογική έξοδο 1 όταν όλες οι είσοδοί της βρίσκονται σε λογική κατάσταση 1.Ο πίνακας αληθείας της είναι ο παρακάτω:

Είσοδος Α	Είσοδος Β	Έξοδος
0	0	0
0	1	0
1	0	0
1	1	1

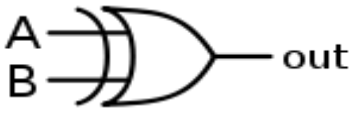
OR



Η πύλη OR εκτελεί την λογική πράξη OR μεταξύ των εισόδων της, η οποία στην άλγεβρα Boole συμβολίζεται με το σύμβολο (+).Οι πύλες OR κατασκευάζονται και με περισσότερες απο δύο εισόδους και δίνουν λογικό 1 όταν μία τουλάχιστον είσοδος τους είναι σε λογική κατάσταση 1. Ο πίνακας αληθείας της είναι ο παρακάτω:

Είσοδος Α	Είσοδος Β	Έξοδος
0	0	0
0	1	1
1	0	1
1	1	1

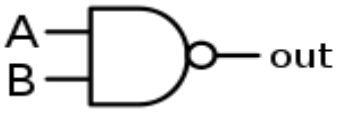
XOR



Η πύλη XOR εκτελεί την λογική πράξη XOR (ΑΠΟΚΛΕΙΣΤΙΚΟ Η') μεταξύ των εισόδων της. Η πράξη XOR στην άλγεβρα Boole συμβολίζεται με το σύμβολο (⊕). Ο πίνακας αληθείας της είναι ο παρακάτω:

Είσοδος Α	Είσοδος Β	Έξοδος
0	0	0
0	1	1
1	0	1
1	1	0

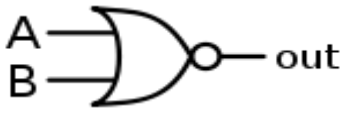
NAND



Η πύλη NAND (ΟΧΙ-ΚΑΙ) δίνει την αντίθετη έξοδο από την AND και ο πίνακας αληθείας της είναι ο παρακάτω:

Είσοδος Α	Είσοδος Β	Έξοδος
0	0	1
0	1	1
1	0	1
1	1	0

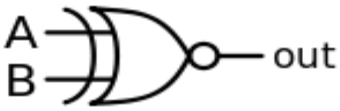
NOR



Η πύλη NOR (ΟΧΙ-Η') δίνει την αντίθετη έξοδο από την OR και ο πίνακας αληθείας της είναι ο παρακάτω:

Είσοδος Α	Είσοδος Β	Έξοδος
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



Η πύλη **XNOR** δίνει την αντίθετη έξοδο από την XOR και ο πίνακας αληθείας της είναι ο παρακάτω:

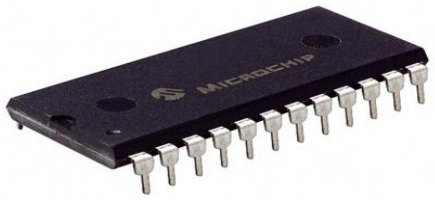
Είσοδος Α	Είσοδος Β	Έξοδος
0	0	1
0	1	0
1	0	0
1	1	1

2.2 Parallel port controller

2.2.1 εισαγωγή

Ένα από τα πιο σημαντικά εξαρτήματα στην εφαρμογή μας είναι το paralel port controller, με τη βοήθεια του οποίου μπορούμε να συνδέσουμε τον ηλεκτρονικό μας υπολογιστή(server) με το ηλεκτρολογικό μας σύστημα, έτσι ώστε να μπορούμε να ελέγχουμε οποιαδήποτε ηλεκτρική συσκευή μέσω αυτού. Για να κατανοήσουμε τη λειτουργία του, θα πρέπει να έχουμε μια γενική γνώση των συσκευών από τις οποίες αποτελείται άλλα και της παράλληλης θήρας του ηλεκτρονικού υπολογιστή.

2.2.2 Μικροελεγκτές



Εικόνα 2.2: Μικροελεγκτής.

Η πιο εξειδικευμένη και δύσχρηστη συσκευή πάνω στο parallel port controller είναι ο μικροελεγτής. Ο μικροελεγκτής είναι ένα μικρό αυτόνομο υπολογιστικό σύστημα, προγραμματισμένο να εκτελεί μια συγκεκριμένη ακολουθία εντολών, οι οποίες έχουν καταχωρηθεί στην μόνιμη μνήμη του. Κάθε φορά που επανεκινείται ένας μικροελεγκτής εκτελείται η ίδια διαδικασία, δέχεται δηλαδή δεδομένα, τα οποία επεξεργάζεται και στη συνέχεια ελέγχει τα συστήματα που εμείς θέλουμε ανάλογα με τα αποτελέσματα αυτής της επεξεργασίας. Οι πιο διαδεδομένες εταιρίες μικροελεγκτών είναι η microchip και η atmel, ενώ στη δικιά μας εφαρμογή χρησιμοποιείται ένας μικροελεγκτής της toshiba, ο ULN2803A.

Όπως γίνεται αντιληπτό οι μικροελεγκτές χρησιμοποιούνται ευρέως για τον έλεγχο αυτοματισμών, ηλεκτρονικών και ηλεκτρικών συσκευών κ.α., και η επιτυχία τους καθορίζεται σε σημαντικό βαθμό από μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή.Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και διάφορες παραλλαγές τους ενώ παλιότερα χρησιμοποιούταν και η assembly.

2.2.3 Ηλεκτρονόμος (relay)

Ο ηλεκτρονόμος είναι ένα από τα πιο βασικά εξαρτήματα που χρησιμοποιούνται για την κατασκευή ηλεκτρικών κυκλωμάτων και κυκλωμάτων αυτοματισμού. Με τη βοήθεια των ηλεκτρονόμων έχουμε τη δυνατότητα να ελέγχουμε έμμεσα κυκλώματα ισχύος πολύ υψηλών φορτίων κατασκευάζοντας βοηθητικά κυκλώματα που λειτουργούν με ανεξάρτητη τάση. Έτσι ο έλεγχος πολύ μεγάλων φορτίων μπορεί να γίνει με περισσότερη ασφάλεια. Στο parallel port controller που χρησιμοποιήσαμε στην εφαρμογή μας χρησιμοποιούνται 8 ηλεκτρονόμοι στην ουσία ως διακόπτες έτσι ώστε να γίνεται ο έλεγχος του κινητήρα μας.
Τα βασικά μέρη ενός ηλεκτρονόμου είναι:

1) Το πηνίο του ηλεκτρονόμου.
2) Ο **πυρήνας**(ή αλλιώς μαγνήτης), που είναι το σταθερό μέρος του σιδηρομαγνητικού υλικού του ηλεκτρονόμου.
3) Ο **οπλισμός**, δηλαδή το κινητό μέρος του σιδηρομαγνητικού υλικού του ηλεκτρονόμου.
4) Οι κύριες **επαφές**, ή αλλιώς επαφές ισχύος του ηλεκτρονόμου. Αυτές είναι οι επαφές στις οποίες συνδέονται οι επαφές τροφοδοσίας του φορτίου.
5) Οι **βοηθητικές επαφές**, που είναι επαφές οι οποίες χρησιμοποιούνται μόνο στα κυκλώματα αυτοματισμού για τον έλεγχο ενδεικτικών λυχνιών, για να αυτοσυγκρατούν τους ηλεκτρονόμους, να τους μανδαλώνουν ηλεκτρικά, για να θέτουν σε λειτουργία ηλεκτρικά κυκλώματα κ.α. Σε αντίθεση με τις κύριες επαφές, το ρεύμα το οποίο μπορεί να διέρχεται μέσω των βοηθητικών επαφών είναι μικρό (2 - 5 A).Τέλος, μια βοηθητική επαφή ενός ηλεκτρονόμου μπορεί να είναι κανονικά κλειστή(συμβολίζεται ως NC από το αγγλικό normaly closed) ή κανονικά ανοικτή(συμβολίζεται ως NO από το αγγλικό normaly open).Οι κανονικά ανοικτές επαφές είναι επαφές οι οποίες σε κατάσταση ηρεμίας είναι ανοικτές και κλείνουν όταν ενεργοποιηθεί ο ηλεκτρονόμος, ενώ οι κανονικά κλειστές επαφές είναι κλειστές όταν βρίσκονται σε κατάσταση ηρεμίας και ανοίγουν όταν ενεργοποιηθεί ο ηλεκτρονόμος.

Γενικά η λειτουργιά του ηλεκτρονόμου έχει ως εξής, όταν το πηνίο του ηλεκτρονόμου βρεθεί υπό τάση δημιουργείται γύρω από αυτό μαγνητικό πεδίο, έτσι ο πυρήνας του ηλεκτρονόμου μετατρέπεται σε έναν μαγνήτη και αυτό έχει σαν αποτέλεσμα την έλξη του οπλισμού προς τον πυρήνα. Χάρη σε αυτήν την κίνηση του οπλισμού και την μηχανική σύνδεσή της με τις βοηθητικές επαφές του ηλεκτρονόμου, μπορούμε να ελέγχουμε το ρελέ μας έτσι ώστε να το χρησιμοποιούμε με επιτυχία στις διάφορες εφαρμογές μας.

Εικόνα 2.3, 2.4 Στην αριστερή εικόνα βλέπουμε ένα απλό ρελέ ισχύος, ενώ στη δεξιά ένα ρελέ που διεγείρεται με συνεχές(dc) ρεύμα.

Τα ρελε μπορούν φυσικά να διεγείρονται με συνεχές ή με εναλλασσόμενο ρεύμα. Στα ρελέ που διεγείρονται με dc ρεύμα μπαίνει μια δίοδος παράλληλα με το πηνίο τους για να προστατέψει το υπόλοιπο κύκλωμα από την αποδιέγερση του πηνίου και το επαγωγικό ρεύμα που προκαλείται από αυτή.

2.2.4 Η παράλληλη θύρα του υπολογιστή (parallel port)

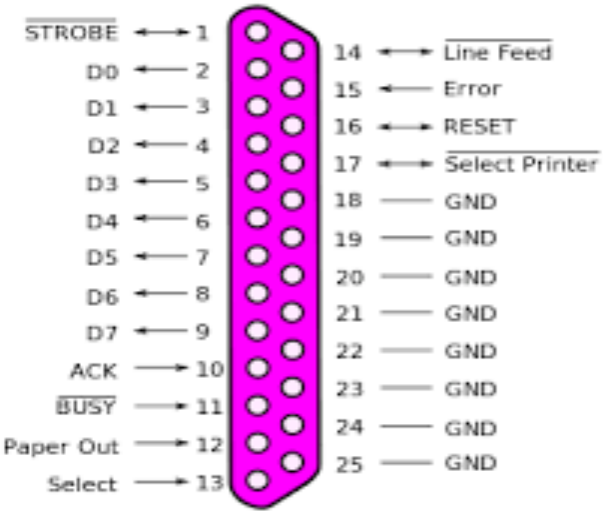
Εικόνα 2.5: Μια παράλληλη θύρα

.

Οι θύρες του υπολογιστή είναι συνήθως οι φυσικές συνδέσεις του υπολογιστή προς τον εξωτερικό κόσμο στις οποίες προσαρμόζονται καλώδια για τη σύνδεση περιφερειακών συσκευών. Οι θύρες χωρίζονται σε δυο κατηγορίες, τις παράλληλες

και τις σειριακές. Ο όρος παράλληλες αναφέρεται στο γεγονός ότι οι θύρες αυτές χρησιμοποιούν πολλά καλώδια μαζεμένα σε ένα χοντρότερο έτσι ώστε η μεταφορά των δεδομένων να γίνεται με παράλληλο τρόπο. Οι παράλληλες θύρες χρησιμοποιούνται εδώ και πολλά χρόνια για τη σύνδεση σαρωτών και εκτυπωτών με τον υπολογιστή, ωστόσο τα τελευταία χρόνια τείνουν να καταργηθούν αφού υπάρχουν ταχύτερες σύγχρονες σειριακές θύρες με μικρότερα, ελαφρύτερα και φθηνότερα καλώδια όπως η θύρα usb.Παρ' όλα αυτά η παράλληλη θύρα είναι η θύρα που χρησιμοποιείται περισσότερο σήμερα σε ερασιτεχνικές εφαρμογές, καθώς παραμένει μία εύκολη λύση για την έξοδο/είσοδο πολλών ψηφιακών σημάτων παράλληλα.

Συνήθως η παράλληλη θύρα είναι θηλυκού τύπου και έχει 25 ακροδέκτες, από τους οποίους μόνο οι 8 (ακροδέκτες από 2 έως 9) είναι ελεύθεροι να χρησιμοποιηθούν για τις εφαρμογές μας. Οι 8 αυτοί ακροδέκτες χρησιμοποιούνται για να εξάγουμε δεδομένα και παρέχουν τάση από 2,4V έως 5,5V για το λογικό 1 και από -0,5V έως 0,4V για το λογικό 0.Οι υπόλοιποι ακροδέκτες δεν μπορούν να χρησιμοποιηθούν στις διάφορες εφαρμογές μας καθώς είτε χρησιμοποιούνται από τον υπολογιστή είτε είναι γειώσεις(ακροδέκτες από 18 έως 25).Στο παρακάτω σχήμα φαίνεται η χρήση μιας παράλληλης θύρας όταν αυτή είναι συνδεδεμένη με έναν εκτυπωτή.



Εικόνα 2.6: Οι επαφές μιας απλής παράλληλης θύρας.

Όσο αφορά τον προγραμματισμό τις παράλληλης θύρας του υπολογιστή αυτό που θα πρέπει να ξέρουμε είναι οι διευθύνσεις τόσο της ίδιας της παράλληλης θύρας όσο και των ακροδεκτών της. Η διεύθυνση του δικού μας parallel port είναι για παράδειγμα το 378h στο δεκαεξαδικό σύστημα (δηλαδή 888 στο δεκαδικό).Με

βάση τα προγράμματα που χρησιμοποιούμε μπορούμε να γράψουμε τις διευθύνσεις και της παράλληλης θύρας αλλά και των ακροδεκτών της και στο δεκαδικό σύστημα, έτσι οι διευθύνσεις των ακροδεκτών της παράλληλης θύρας θα είναι οι εξής:

ακροδέκτης	2	3	4	5	6	7	8	9
Τιμή στο πρόγραμμα	1	2	4	8	16	32	64	128

Οι τιμές(διευθύνσεις) των ακροδεκτών έχουν καθοριστεί με βάση το δυαδικό σύστημα(βλέπε υποενότητα ψηφιακά).Έτσι με βάση τον παραπάνω πίνακα γίνεται κατανοητό ότι αν θέλουμε να αναφερθούμε στον δεύτερο ακροδέκτη της παράλληλης θύρας τότε θα πρέπει να γράψουμε στο πρόγραμμά μας (888,1) ενώ αν θέλουμε να αναφερθούμε στον τρίτο, τον τέταρτο και τον έκτο θα πρέπει να γράψουμε (888,22).Προσθέτοντας δηλαδή τις τιμές των ακροδεκτών μπορούμε να αναφερθούμε συγκεκριμένα σε όποιους από τους ακροδέκτες θέλουμε, έτσι όλοι οι ακροδέκτες μαζί έχουν την τιμή 255.Όλα τα παραπάνω χάρη στο δυαδικό σύστημα αρίθμησης.

2.3 Ρυθμιστής στροφών

2.3.1. Εισαγωγή

Όπως είδαμε στο προηγούμενο υποκεφάλαιο οι χρήσεις ενός ηλεκτρικού κινητήρα στη σύγχρονη κοινωνία είναι πολλές, για να χρησιμοποιήσουμε όμως σωστά έναν τέτοιο κινητήρα υπάρχει η ανάγκη να τον ‘ελέγξουμε’, να μπορούμε δηλαδή να ρυθμίσουμε τις στροφές και την ταχύτητα του, το αν θα λειτουργεί δεξιόστροφα η αριστερόστροφα αλλά και να τον φέρουμε σε κατάσταση πέδης η σε κατάσταση μη λειτουργίας όταν το επιθυμούμε. Σε αυτό το υποκεφάλαιο θα ασχοληθούμε όπως φανερώνει και ο τίτλος του με την ρύθμιση των στροφών του ηλεκτρικού μας κινητήρα. Γενικά υπάρχουν πολλοί τρόποι για να ρυθμίσει κανείς έναν ηλεκτρικό κινητήρα, συνήθως όμως αυτό επιτυγχάνεται με μια ηλεκτρονική διάταξη που παραθέτουμε στην είσοδο του κινητήρα. Έτσι για να πραγματοποιηθεί το πρακτικό μέρος της πτυχιακής αυτής εργασίας χρησιμοποιήθηκε μια τέτοια διάταξη η οποία ταίριαζε με της ανάγκες μας αλλά και με τα χαρακτηριστικά του ηλεκτρικού κινητήρα που χρησιμοποιήσαμε.

2.3.2 Αντιστάσεις, ποτενσιόμετρα και τρίμμερ

Αντιστάσεις

Κάθε είδους υλικό παρουσιάζει αντίσταση στην ροή του ρεύματος. Το μέγεθος της αντίστασης που παρουσιάζει ένα υλικό εξαρτάται από το στοιχείο που είναι κατασκευασμένο, την διάμετρός του και το μήκος του. Όσο μεγαλύτερη είναι η διάμετρος τόσο μικρότερη είναι η αντίσταση και όσο μεγαλύτερο είναι το μήκος του υλικού τόσο αυξάνεται η αντίστασή του. Τα μέταλλα έχουν διαφορετική αντίσταση μεταξύ τους η οποία αυξάνεται με την αύξηση της θερμοκρασίας τους, ενώ μερικά υλικά όπως ο άνθρακας μειώνουν την αντίστασή τους όταν αυξάνεται η θερμοκρασία τους.

Στα ηλεκτρονικά κυκλώματα η χρησιμοποίηση αντιστάσεων δημιουργεί περιορισμό της ροής του ρεύματος και πτώση τάσης στα άκρα τους. Μονάδα μέτρησής τους είναι το Ohm (Ω).

Γενικά οι αντιστάσεις χωρίζονται σε διάφορες κατηγορίες. Υπάρχουν οι σταθερές γραμμικές αντιστάσεις, που με τη σειρά τους χωρίζονται σε αντιστάσεις σύρματος, στρώματος και μίγματος ανάλογα με το υλικό από το οποίο αποτελούνται. Στις μεταβλητές γραμμικές αντιστάσεις έχουμε τα ποτενσιόμετρα και τους ροοστάτες και στις μη γραμμικές τα θερμίστορ, τα βαρίστορ και τις φωτοαντιστάσεις.

Επίσης οι αντιστάσεις έχουν κάποια χαρακτηριστικά μεγέθη τα οποία είναι τα εξής:

1. Τιμή αντίστασης
Είναι η τιμή της αντίστασης με μονάδα μέτρησης το Ωμ.

2. Ισχύς της αντίστασης
Είναι η μέγιστη τιμή της ισχύος που μπορεί να καταναλωθεί πάνω στην αντίσταση υπό μορφή θερμότητας.

3. Ανοχή
Είναι η απόκλιση της τιμής της αντίστασης από την τιμή που δίνει ο κατασκευαστής. Οι τιμές ανοχής ορίζονται από τον κατασκευαστή και είναι ±10%, ±5%, ±2%, ±1% και ±0,5%.

4. Γραμμικότητα
Οι αντιστάσεις στην λειτουργία τους πρέπει να ακολουθούν τον νόμο του Ohm (I=V/R), για να μην παρατηρούνται μεταβολές των τάσεων και των εντάσεων στις οποίες λειτουργούν. Στις μη γραμμικές αντιστάσεις όμως επιδιώκουμε τέτοιες μεταβολές.

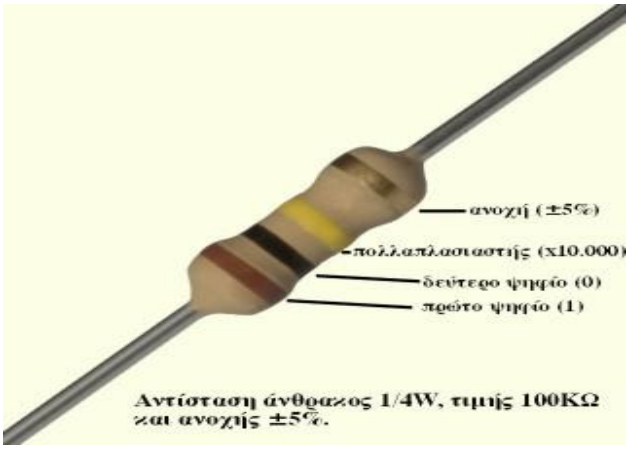
5. Τάση θορύβου

Η θερμότητα που αναπτύσουν οι αντιστάσεις στο σώμα τους έχουν σαν αποτέλεσμα να δημιουργείται θερμικός θόρυβος στο κύκλωμα με συνέπεια την ύπαρξη παρασιτικής τάσης στα άκρα τους που μετριέται σε μV ανά V της ονομαστικής τάσης λειτουργίας τους.

6. Σταθερότητα της αντίστασης

Η συνεχείς λειτουργία της αντίστασης, στο πέρασμα του χρόνου παρουσιάζει αλλοιώσεις στο υλικό κατασκευής της με αποτέλεσμα την αλλοίωση της ονομαστικής τιμής της. Η σταθερότητα της αντίστασης εκφράζεται με την σταθερότητα της ονομαστικής της τιμής ανεξάρτητα του χρόνου λειτουργίας της.

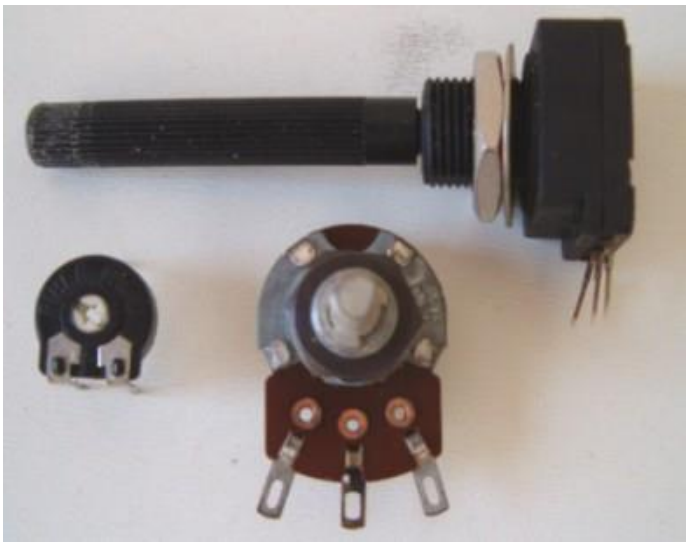
Τέλος πρέπει να ειπωθεί ότι παρόλο που χρησιμοποιούνται πολλές μέθοδοι υπολογισμού αντιστάσεων η πιο δημοφιλής είναι αυτή του υπολογισμού ωμικής αντίστασης με τη χρήση του χρωματικού κώδικα.



Χρώμα	1ο Ψηφίο	2ο Ψηφίο	3ο Ψηφίο	Πολ/στής	Ανοχή	Συν.Θερμ.
			Μπορεί να μην υπάρχει			Μπορεί να μην υπάρχει
Ασημί				10^{-2}	10%	
Χρυσό				10^{-1}	5%	
Μαύρο	0	0	0	10^{0}		200ppm/OC
Καφέ	1	1	1	10^{1}	1%	100ppm/OC
Κόκκινο	2	2	2	10^{2}	2%	50ppm/OC
Πορτοκαλί	3	3	3	10^{3}		15ppm/OC
Κίτρινο	4	4	4	10^{4}		25ppm/OC
Πράσινο	5	5	5	10^{5}	0,50%	
Μπλε	6	6	6	10^{6}	0,25%	10ppm/OC
Μωβ	7	7	7	10^{7}	0,10%	5ppm/OC
Γκρι	8	8	8	10^{8}	0,05%	
Λευκό	9	9	9	10^{9}		1ppm/OC

Εικόνες 2.7, 2.8: Χρωματικός κώδικας σε μια αντίσταση και η πινακίδα με το χρωματικό κώδικα.

Ποτενσιόμετρα και τρίμμερ



Εικόνα 2.9: Ποτενσιόμετρο και τρίμμερ.

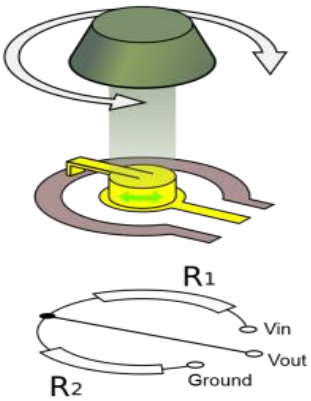
Οι αντιστάσεις μεταβλητής τιμής είναι εκείνες που μπορούμε να μεταβάλουμε την ονομαστική τους τιμή και κατασκευάζονται από άνθρακα ή σύρμα ανάλογα με την ισχύ τους. Το ποτενσιόμετρα ή τα τρίμμερ που χρησιμοποιούμε, είναι μεταβλητές αντιστάσεις κυρίως από άνθρακα, ενώ για μεγαλύτερες ισχύς κατασκευάζονται από αγωγό υψηλής αντίστασης. Συνήθως τα τρίμμερ και τα ποτενσιόμετρα χαμηλής ισχύος αποτελούνται από μια μονωτική πλάκα πάνω στην οποία στηρίζεται το υλικό της αντίστασης και ένα δρομέα που περιστρέφεται εφαπτόμενος πάνω στο υλικό αυτό. Οι μεταβλητές αντιστάσεις σύρματος αποτελούνται από ένα μονωτικό υλικό κυλινδρικής μορφής, πάνω στο οποίο τυλίγεται ένας αγωγός υψηλής αντίστασης. Στον συρμάτινο αγωγό εφάπτεται ένας δρομέας ο οποίος ολισθαίνει κατά μήκος της αντίστασης σύρματος με αποτέλεσμα να παίρνουμε την μεταβλητή τιμή της αντίστασης που θέλουμε.

Τα ποτενσιόμετρα άνθρακα χωρίζονται σε δύο κατηγορίες: α) στα γραμμικά ποτενσιόμετρα και β) στα λογαριθμικά ποτενσιόμετρα. Τα γραμμικά ποτενσιόμετρα είναι εκείνα που η τιμή της αντίστασης μεταβάλλεται με σταθερό βήμα ανάλογα με την γωνία στροφής του ποτενσιόμετρου. Τα λογαριθμικά ποτενσιόμετρα ή αλλιώς μη γραμμικά, είναι εκείνα που η αντίστασή τους μεταβάλλεται στην αρχή αργά και κατόπιν γρηγορότερα και η παραμικρή αύξηση της γωνίας στροφής του δρομέα προκαλεί απότομη μεταβολή της αύξησης ή μείωσης της τιμής της αντίστασης με μεγάλη διαφορά τιμής από την προηγούμενη θέση.

Τα λογαριθμικά ποτενσιόμετρα χρησιμοποιούνται για την ρύθμιση της έντασης ήχου (volume control), σε ενισχυτές και προενισχυτές ήχου ενώ τα γραμμικά σε

κυκλώματα τα οποία θέλουμε να έχουμε σταθερό βήμα και ακρίβεια στην ρύθμιση. Τα ποτενσιόμετρα αναγράφουν την ονομαστική τους τιμή, καθώς και τον τύπο τους, δηλαδή αν είναι γραμμικά ή λογαριθμικά. Τα γραμμικά ποτενσιόμετρα αναγνωρίζονται από το γράμμα Α, ενώ τα λογαριθμικά από το γράμμα Β.

Λειτουργία του ποτενσιόμετρου



Εικόνα 2.10: Διάγραμμα ποτενσιόμετρου.

Η γκρι λωρίδα σε σχήμα πετάλου είναι φτιαγμένη από ειδικό υλικό υψηλής αντίστασης. Περιστρέφοντας τον άξονα μπορούμε να μετακινήσουμε τη θέση του δρομέα (απεικονίζεται με κίτρινο χρώμα) πάνω στο σύρμα. Καθώς η αντίσταση είναι ανάλογη με το μήκος του σύρματος, όταν ο δρομέας μετακινείται αριστερόστροφα τότε η αντίσταση του αριστερού τμήματος του σύρματος γίνεται μικρότερη ενώ του δεξιού μεγαλύτερη. Το αντίστροφο συμβαίνει όταν κινείται ο δρομέας δεξιόστροφα.

Η συνολική αντίσταση του ποτενσιόμετρου παραμένει αμετάβλητη ανεξάρτητα από τη θέση στην οποία βρίσκεται ο δρομέας. Ως εκ τούτου, το ρεύμα που το διαρρέει παραμένει πρακτικά σταθερό.

Αν συνδέσουμε τη γείωση στον πρώτο ακροδέκτη, στο μεσαίο ακροδέκτη την τάση εξόδου και στον δεξιό ακροδέκτη την τάση εισόδου τότε η τάση εξόδου θα είναι μηδέν όταν το ποτενσιόμετρο βρίσκεται τέρμα αριστερά και θα αυξάνεται καθώς το περιστρέφουμε δεξιόστροφα, μέχρι να γίνει ίση με την τάση εισόδου στο δεξί τέρμα.

2.3.3 Πυκνωτές

Εικόνα 2.11: Διάφοροι πυκνωτές.

Ένας πυκνωτής είναι στην ουσία ένα σύστημα δύο γειτονικών αγωγών ανάμεσα στους οποίους παρεμβάλλεται μονωτικό υλικό. Οι δύο αγωγοί ονομάζονται οπλισμοί του πυκνωτή και συνήθως κατασκευάζονται από μέταλλα όπως ο ορείχαλκος, ο επικαδμιωμένος σίδηρος ή το αλουμίνιο, ενώ το παρεμβαλλόμενο υλικό ονομάζεται διηλεκτρικό του πυκνωτή και για την κατασκευή του χρησιμοποιούνται μη αγώγιμα υλικά όπως χαρτί, λάδι, γυαλί, αέρας, ταντάλιο, πολυπροπυλένιο, μίκα και πολλά άλλα. Οι πυκνωτές χρησιμοποιούνται ως όργανα αποθήκευσης ηλεκτρικής ενέργειας καθώς έχουν την ιδιότητα να αποθηκεύουν ηλεκτρικό φορτίο όταν εφαρμοστεί τάση στα άκρα τους. Μερικές χρήσεις των πυκνωτών είναι σε ηλεκτρονικά φίλτρα όπου φιλτράρει ανεπιθύμητα ηλεκτρικά σήματα, σε τροφοδοτικά που μετατρέπουν το εναλλασσόμενο ρεύμα σε συνεχές κ.α.

Όταν ένας πυκνωτής είναι φορτισμένος, οι οπλισμοί του έχουν ηλεκτρικά φορτία κατά μέτρο ίσα και αντίθετα. Ονομάζουμε φορτίο του πυκνωτή το φορτίο του θετικά φορτισμένου οπλισμού του. Η ποσότητα του φορτίου που μπορεί να συγκρατήσει ένας πυκνωτής είναι ανάλογη της επιφάνειας των οπλισμών του και αντίστροφος ανάλογη της μεταξύ τους απόσταση. Το πηλίκο του φορτίου ενός πυκνωτή προς την τάση(τη διαφορά δυναμικού δηλαδή που αναπτύσσεται μεταξύ των οπλισμών ενός φορτισμένου πυκνωτή) του ονομάζεται χωρητικότητα και εκφράζει την ικανότητα ενός πυκνωτή να αποθηκεύει ενέργεια. Η χωρητικότητα ενός πυκνωτή συμβολίζεται με το γράμμα C και μονάδα μέτρησής της είναι το Farad.

Κύρια χαρακτηριστικά μεγέθοι των πυκνωτών:

- Ονομαστική χωρητικότητα.

Η τιμή της χωρητικότητας για την οποία έχει κατασκευαστεί ένας πυκνωτής. Η τιμή αυτή είναι υπολογισμένη για συγκεκριμένες θερμοκρασίες και συχνότητες λειτουργίας.

- Ανοχή χωρητικότητας.

  Η ανοχή του πυκνωτή σε τιμές διαφορετικές από την ονομαστική του.

- Τάση λειτουργίας.

  Η μέγιστη τάση που μπορεί να δεχτεί ένας πυκνωτής χωρίς να καταστραφεί, δίνεται συνήθως από τον κατασκευαστή.

- Συχνότητα αναφοράς.

  Η συχνότητα λειτουργίας για την οποία ισχύει η ονομαστική χωρητικότητα του πυκνωτή.

- Αντίσταση μόνωσης.

  Η αντίσταση μεταξύ των ηλεκτροδίων του πυκνωτή καθώς και η αντίσταση μεταξύ των ηλεκτροδίων και του περιβλήματος του πυκνωτή.

<u>Είδη πυκνωτών</u>

Οι πυκνωτές χωρίζονται σε δυο κατηγορίες, τους διηλεκτρικούς και τους ηλεκτρολυτικούς πυκνωτές. Οι δυο αυτές κατηγορίες πυκνωτών διαφέρουν στο τρόπο κατασκευής τους και στο τρόπο χρήσης τους, έχουν όμως την ίδια αρχή λειτουργίας.

Οι διηλεκτρικοί πυκνωτές χωρίζονται με τη σειρά τους σε άλλες δυο επιμέρους κατηγορίες, τους πυκνωτές σταθερής χωρητικότητας και τους πυκνωτές μεταβλητής χωρητικότητας.

Πυκνωτές σταθερής χωρητικότητας:

1) Πυκνωτές χαρτιού.

   Έχουν για υλικό διηλεκτρικού ειδικά κατεργασμένο χαρτί, περιελιγμένο σε κυλινδρική μορφή και εμποτισμένο συνήθως σε μονωτικό λάδι για την προστασία του από την υγρασία.

2) Πυκνωτές πλαστικής ταινίας.

   Έχουν ως υλικό διηλεκτρικού διάφορα είδη πλαστικού. Υπάρχουν πυκνωτές πολυεστέρα, πολυπροπυλενίου, πολυστερηνίου, πολυανθρακικού κ.α., το κάθε υλικό με τις δικές του διαφορετικές ιδιότητες, με αυτόν τον τρόπο καλύπτουν ένα μεγάλο φάσμα χωρητικοτήτων και τάσεων.

3) Πυκνωτές μίκας.

   Χρησιμοποιούν ως υλικό διηλεκτρικού τη μίκα και διακρίνονται σε δυο τύπους, στους πυκνωτές φύλλων μίκας και στους πυκνωτές ταινίας μίκας.

4) Πυκνωτές γυαλιού.

   Με υλικό διηλεκτρικού το γυαλί, χάρη στο οποίο οι πυκνωτές αυτοί έχουν πολύ υψηλή θερμοκρασία λειτουργίας, πολύ υψηλή αντίσταση μόνωσης αλλά και πολύ χαμηλές τιμές χωρητικοτήτων.

5) Κεραμικοί πυκνωτές.

   Στους κεραμικούς πυκνωτές το διηλεκτρικό είναι κάποιο κεραμικό υλικό, στο οποίο γίνεται πρόσμιξη με τιτάνιο, βάριο ή ασβέστιο. Μπορούν να λειτουργησούν σε πολύ υψηλές τάσεις που φτάνουν την περιοχή των KV.

Οι πυκνωτές μεταβλητής χωρητικότητας αποτελούνται από τους μεταβλητούς πυκνωτές που έχουν σαν διηλεκτρικό τον αέρα και από τους ρυθμιζόμενους πυκνωτές με διηλεκτρικά υλικά όπως το γυαλί, ο αέρας, ο χαλαζίας, η μίκα κ.α.

Η λειτουργιά των ηλεκτρολυτικών πυκνωτών στηρίζεται στην αρχή της ηλεκτρόλυσης, και χάρη σε αυτό η παρουσία του μετάλλου της ανόδου, ενός ηλεκτρολύτη και της καθόδου είναι απαραίτητη. Είναι επίσης αναγκαίο να πολωθούν ορθά(άνοδος θετικό δυναμικό και κάθοδος αρνητικό) καθώς σε αντίθετη περίπτωση(ανάστροφη πόλωση) θα υποστούν σοβαρές ζημιές και θα καταστραφούν. Το διηλεκτρικό υλικό των πυκνωτών αυτών είναι το οξείδιο του μετάλλου της ανόδου. Τέλος πρέπει να υπωθεί ότι οι εν λόγω πυκνωτές παρουσιάζουν μεγάλες τιμές σταθερής χωρητικότητας.

2.3.4 Θυρίστορ, diac και triac

Τα θυρίστορ είναι ηλεκτρονικά εξαρτήματα που αποτελούνται από διαδοχικά στρώματα ημιαγωγών τύπου P και N και λειτουργούν στην ουσία ως ένας διακόπτης με δυο καταστάσεις, αυτή της αγωγιμότητας και αυτή της αποκοπής. Μια από τις πιο κοινές και σημαντικές εφαρμογές των θυρίστορ είναι ο έλεγχος της λειτουργίας των κινητήρων. Οι κυριότεροι τύποι θυρίστορ είναι:

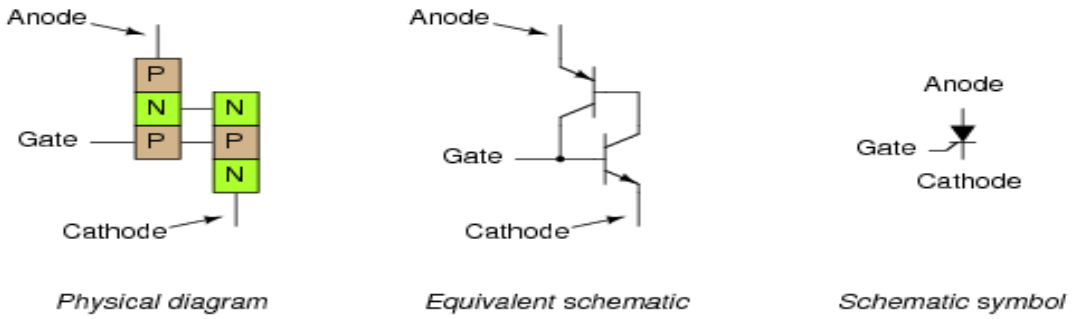
1) Ο ελεγχόμενος ανορθωτής πυριτίου, SCR(silicon control rectifier).

2) Η αμφίδρομη δίοδος θυρίστορ, DIAC(diode alternating current).

3) Ο αμφίδρομος ελεγχόμενος ανορθωτής πυριτίου, TRIAC(triode alternating current).

SCR

Ο ελεγχόμενος ανορθωτής πυριτίου είναι μία συσκευή η οποία έχει τέσσερα, εναλλασσόμενα ημιαγώγιμα στρώματα και είναι σχεδόν πάντα κατασκευασμένη από πυρίτιο. Το scr έχει τρεις ακροδέκτες συνδεδεμένους σε τρία από τα τέσσερα ημιαγώγημα στρώματα του, την άνοδο, την κάθοδο και την πύλη.



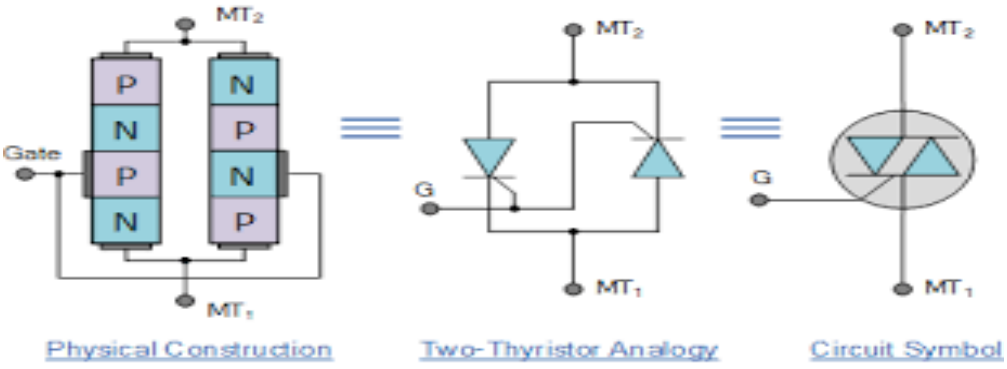
Εικόνα 2.12: Διάγραμμα και σχηματικό σύμβολο ενός SCR.

Ο πιο εύκολος τρόπος για να καταλάβουμε τη λειτουργία ενός SCR είναι να δούμε τα τέσσερα στρώματα του ως δυο PNP και NPN τρανζίστορ που αλληλοσυνδέονται έτσι ώστε να είναι ισοδύναμα με μια συσκευή(SCR) τεσσάρων στρωμάτων.

Μπορούμε να ενεργοποιήσουμε έναν SCR κάνοντας έστω και στιγμιαία την τάση πύλης του θετική(δίνοντας δηλαδή ένα μικρό ρεύμα στην πύλη) και να τον απενεργοποιήσουμε όταν σχεδόν μηδενίσουμε την τάση ανόδου – καθόδου του. Όταν ο SCR είναι ενεργοποιημένος και άγει ένα υψηλό ρεύμα από την κάθοδο προς την άνοδο, τότε η συσκευή άγει στην ορθή κατεύθυνση, ενώ αν η πολικότητα της καθόδου ως προς άνοδο αντιστραφεί, τότε η συσκευή άγει μόνο ένα μικρό ρεύμα διαρροής, το οποίο θα ρέει κατά την αντίθετη κατεύθυνση.

Ένας ελεγχόμενος ανορθωτής πυριτίου μπορεί να χρησιμοποιηθεί σε εφαρμογές όπως ο έλεγχος ισχύος ενός κυκλώματος εναλλασσόμενου ρεύματος(π.χ. κινητήρας ή κύκλωμα φωτισμού), η προστασία φορτίων από υπερτάσεις και σε συνδυασμό με μικροεπεξεργαστές έναντι των απλών ηλεκτρονόμων.

Triac

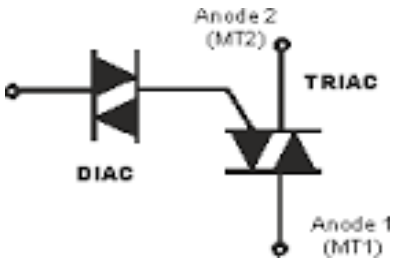


Εικονα 2.13: Διαγραμματα και συμβολο ενός triac.

Μπορούμε να δούμε ένα triac ως μία συσκευή NPNP, παραλλήλως συνδεδεμένη με μία συσκευή PNPN, όπως στο παραπάνω σχήμα. Ένα triac έχει τρεις αγωγούς σύνδεσης, δυο κύριους ακροδέκτες που ο καθένας είναι συνδεδεμένος σε μια PN επαφή, και μια πύλη, επίσης συνδεδεμένη σε μια PN επαφή.

Το triac εν αντιθέση με το SCR επιτρέπει τη διέλευση ρεύματος και προς τις δυο κατευθύνσεις, ενώ η αγωγή η όχι του ρεύματος ελέγχεται με ένα σήμα στην πύλη του(θετικός ή αρνητικός παλμός ανάλογα με την κατάσταση αγωγιμότητας που θέλουμε).Το triac χρησιμοποιείται κυρίως για να ελέγχουμε το εναλλασσόμενο ρεύμα σε ηλεκτρολογικές διατάξεις φωτισμού ή σε διατάξεις με μικρούς κινητήρες.

Diac



Εικόνα 2.14: Συνδυασμός diac και triac.

Το diac ένα ηλεκτρονικό στοιχείο το οποίο χρησιμοποιείται συνήθως ως συσκευή σκανδαλισμού για ένα triac και αποτελείται από τρία ημιαγώγιμα στρώματα και δυο ακροδέκτες.

Όσο το diac είναι σε μη αγώγιμη κατάσταση άγει μόνο ένα μικρο ρεύμα

διαρροής, όταν όμως του παρέχεται τάση αρκετά υψηλή τότε ενεργοποιείται και το ρεύμα που το διαπερνάει αυξάνεται απότομα. Με βάση τα παραπάνω μπορούμε να πούμε ότι ένα diac λειτουργεί στην ουσία όπως ένας αμφίδρομος διακόπτης που ενεργοποιείται όταν ξεπερασθεί η τάση διάσπασής του.

2.4 Ηλεκτρικοί κινητήρες

2.4.1 Εισαγωγή και ιστορική αναδρομή

Γενικά οι στρεφόμενες ηλεκτρικές μηχανές μπορούν να λειτουργήσουν σαν κινητήρες, σαν γεννήτριες(μετατροπή μηχανικής ενέργειας σε ηλεκτρική) και σαν πέδη(φρένο).

Κατά τη λειτουργιά μιας ηλεκτρικής μηχανής σαν κινητήρα παρέχεται ηλεκτρική ενέργεια στο κύριο τύλιγμα της μηχανής(τύλιγμα τυμπάνου) και αποδίδεται μηχανική ενέργεια πάνω σε μια περιστρεφόμενη άτρακτο. Στη σύγχρονη εποχή η χρήση του ηλεκτρικού κινητήρα είναι ευρεία τόσο στη βιομηχανία όσο και στην καθημερινή ζωή των περισσότερων ανθρώπων.

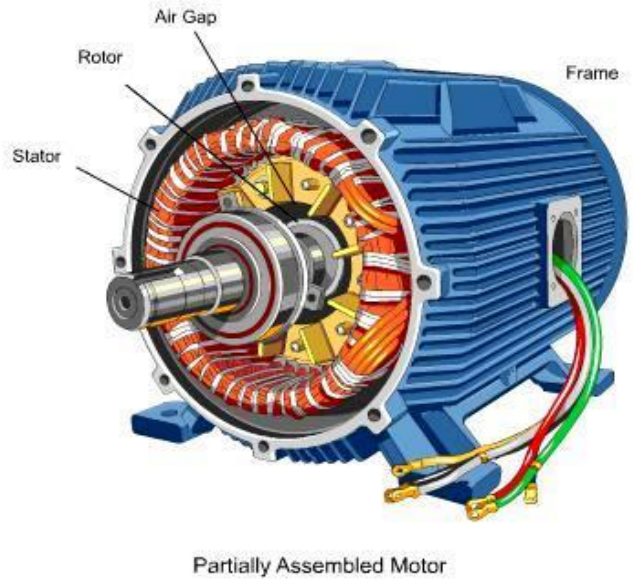
Εικόνα 2.15: Παραδείγματα καθημερινής χρήσης ηλεκτρικών κινητήρων.

Για να φτάσει ο ηλεκτρικός κινητήρας στη μορφή που έχει σήμερα χρειάστηκε να περάσουν πολλά χρόνια, ξεκινώντας από τον Σκοτσέζο μοναχό Andrew Gordon το 1740 και τον Ούγγρο φυσικό Anyos Jeldik που το 1828 δημιούργησε την πρώτη συσκευή με στάτορα, δρομέα και μετατροπέα ηλεκτρικού ρεύματος. Ο πρώτος εμπορικός κινητήρας συνεχούς ρεύματος κατασκευάστηκε το 1871 και το 1886 ο Frank Julian Sprague εφηύρε τον πρώτο πρακτικό κινητήρα συνεχούς ρεύματος. Οι πρώτοι πρακτικοί κινητήρες εναλλασσόμενου ρεύματος ήταν εφευρέσεις τόσο του

Nikola tesla όσο και του Galileo Ferraris.Ετσι φτάσαμε στη σημερινή μορφή των ηλεκτρικών κινητήρων και τις τεράστιες αλλαγές που αυτοί έφεραν στη ζωή και την κοινωνία μας.

2.4.2 Βασικές κατηγορίες ηλεκτρικών κινητήρων και η λειτουργία τους

Οι ηλεκτρικοί κινητήρες παρά την διάκριση τους σε πολλές κατηγορίες και υποκατηγορίες έχουν κάποια κοινά στοιχειά δομής. Οι ηλεκτροκινητήρες αποτελούνται από το σταθερό μέλος που λέγεται στάτης, και το στρεφόμενο το λεγόμενο και δρομέα. Οι πυρήνες του στάτη και του δρομέα κατασκευάζονται από σιδερομαγνητικό υλικό σε μορφή μονωμένων μεταξύ τους ελασμάτων, με αυτόν το τρόπο μειώνεται η μαγνητική αντίσταση των δρόμων της μαγνητικής ροής και ελαττώνεται η απώλεια δινορρευμάτων.Ο δρομέας και ο στάτης χωρίζονται από ένα μικρό διάκενο αέρα.Το τύλιγμα τύμπανου βρίσκεται είτε στο στάτη είτε στο δρομέα.Τέλος το μαγνητικό κύκλωμα συμπληρώνεται με το σιδερομαγνητικό υλικό του άλλου κύριου μέλους της μηχανής στο οποίο είναι τοποθετημένα τα πηνία διέγερσης η τυλίγματα πεδίου που ενεργούν σαν κύριες πήγες μαγνητικής ροής.Τα παραπάνω περιγράφουν μια παραδοσιακή δομή ενός ηλεκτροκινητήρα.

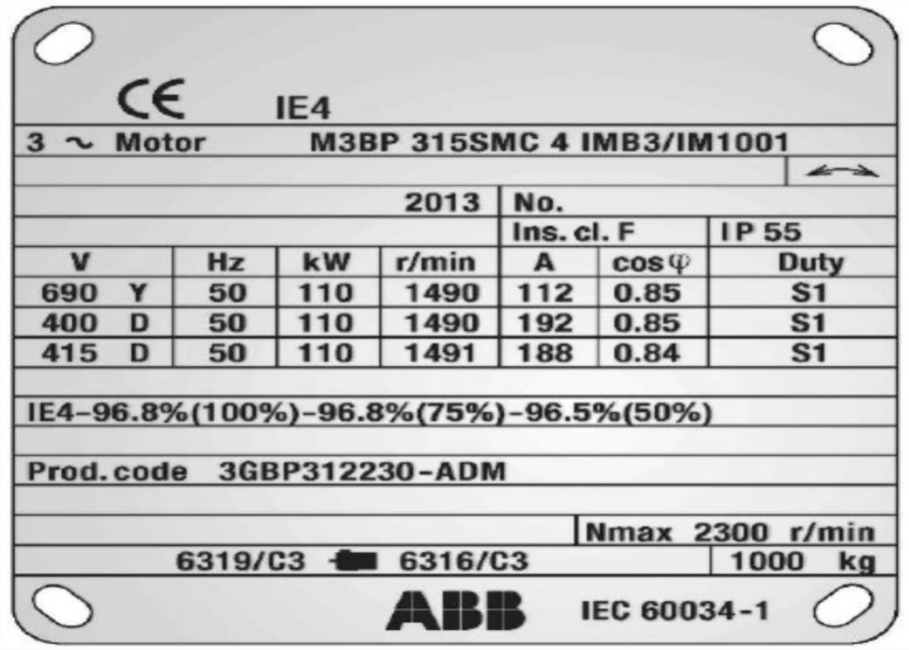


Εικόνα 2.16: Ηλεκτρικός κινητήρας και τα μέρη του.

Τα απαραίτητα στοιχεία για κάθε ηλεκτροκινητήρα τα οποία και προσδιορίζουν αυτόν εμπορικά είναι:

- Η απαιτούμενη τάση για την τροφοδοσία του σε volt (V).
- Το είδος της απαιτούμενης τάσης, συνεχές ή εναλλασσόμενο ρεύμα (DC ή AC) και στη 2η περίπτωση, μονοφασικό (1PH) ή τριφασικό (3PH).
- Η συχνότητα του εναλλασσόμενου ρεύματος, εφόσον πρόκειται για ηλεκτροκινητήρα AC (Hertz).
- Η ισχύς του κινητήρα (W ή HP)
- Η ένταση του ρεύματος σε αμπέρ που διαρρέει τον κινητήρα, και
- Η αποκτώμενη ταχύτητα περιστροφής του άξονα του κινητήρα σε στροφές ανά λεπτό (rpm).

Όλα τα παραπάνω στοιχεία φέρονται χαραγμένα, από τους κατασκευαστές, σε ειδική ενσωματωμένη στον ηλεκτροκινητήρα πινακίδα, καθώς και ο αριθμός της έγκρισης του Υπουργείου Βιομηχανίας για εμπορική διάθεση ή άλλα σύμβολα πιστοποίησης ασφαλούς λειτουργίας.



Εικόνα 2.17: Πινακίδα ηλεκτρικού κινητήρα.

Οι ηλεκτρικοί κινητήρες χωρίζονται στα παρακάτω είδη:
-Κινητήρες συνεχούς ρεύματος (DC)
(παράλληλης διέγερσης, διέγερσης σειράς, σύνθετης διέγερσης, ξένης διέγερσης)
-Κινητήρες εναλλασσόμενου ρεύματος (AC)
(Σύγχρονοι κινητήρες, επαγωγικοί η ασύγχρονοι κινητήρες)

Ηλεκτρικοί κινητήρες dc

Οι ηλεκτρικοί κινητήρες συνεχόμενου ρεύματος αποτελούνται από το στάτη και το δρομέα.

Ο στάτης αποτελείται από:
α)Το ζύγωμα, μια κυλινδρική στεφάνη από συμπαγή σίδηρο που ενώνει μηχανικά και μαγνητικά τους μαγνητικούς πόλους.
β)Τους μαγνητικούς πόλους, σιδηροπυρήνες από ελάσματα μαλακού σιδήρου και τα τυλίγματα διέγερσής τους.
γ)Τα καλύμματα.
δ)Τον ψηκτροφορέα με τις ψήκτρες(καρβουνάκια).
ε)Το κιβώτιο ακροδεκτών , όπου καταλήγουν οι ακροδέκτες των βασικών τυλιγμάτων της μηχανής.

Ο δρομέας αποτελείται από:
α) Τον άξονα που είναι κατασκευασμένος από ατσάλι και πάνω σε αυτόν στερεώνεται το επαγωγικό τύμπανο.
β)Το επαγωγικό τύμπανο που περιλαμβάνει το επαγωγικό τύλιγμα και τον πυρήνα.
γ)Το συλλέκτη.
ε)Τον ανεμιστήρα.

Η αλλαγή φοράς περιστροφής στους κινητήρες συνεχούς ρεύματος επιτυγχάνεται αλλάζοντας τη διεύθυνση του ρεύματος στο επαγωγικό τύλιγμα του τυμπάνου και κρατώντας σταθερή τη διεύθυνση του ρεύματος διέγερσης η το αντίθετο.

Μπορούμε να ρυθμίσουμε την ταχύτητα περιστροφής ενός dc κινητήρα με τους εξής τρόπους:
α)με ρυθμιστική αντίσταση στο τύλιγμα διέγερσης.
β)με ρυθμιστική αντίσταση στο επαγωγικό τύμπανο.
γ) με μεταβολή της τάσης τροφοδότησης του επαγωγικού τυμπάνου.

Τη στιγμή της εκκίνησης της περιστροφής ενός κινητήρα συνεχούς ρεύματος η ταχύτητα περιστροφής είναι μηδενική, έτσι είναι μηδενική και η τάση που αναπτύσσεται στο εσωτερικό του. Συνεπώς το ρεύμα του οπλισμού παίρνει πολύ μεγάλη τιμή και έτσι υπάρχει αυξημένη πιθανότητα βλάβης του κινητήρα. Γι αυτό το λόγο εισάγεται στο σύστημα και σε σειρά με το τύλιγμα οπλισμού μια αντίσταση εκκίνησης.

Κινητήρας παράλληλης διέγερσης

Σε αυτούς τους κινητήρες το τύλιγμα διέγερσης συνδέεται παράλληλα με το τύλιγμα του κινητήρα.

Κινητήρας διέγερσης σειράς

Στους κινητήρες αυτούς το τύλιγμα διέγερσης συνδέεται σε σειρά με το τύλιγμα του κινητήρα.

Κινητήρας σύνθετης διέγερσης

Στους κινητήρες σύνθετης διέγερσης χρησιμοποιούνται δυο τυλίγματα διέγερσης, ένα σε σειρά και ένα σε παραλληλία.

Κινητήρας ξένης(ανεξάρτητης) διέγερσης

Σε αυτούς τους κινητήρες το κύκλωμα διέγερσης τροφοδοτείται από μια ανεξάρτητη πηγή συνεχούς τάσης.

Ηλεκτρικοί κινητήρες ac

Κινητήρας εναλλασσόμενου ρεύματος είναι κάθε μηχανή, η οποία μετατρέπει την ηλεκτρική ενέργεια σε μηχανική, με την προϋπόθεση ότι συνδέεται σε εναλλασσόμενο δίκτυο.

Ανάλογα με την κατασκευή και με τη λειτουργία τους οι κινητήρες εναλλασσόμενου ρεύματος χωρίζονται σε δύο μεγάλες κατηγορίες:

α) τους σύγχρονους, και

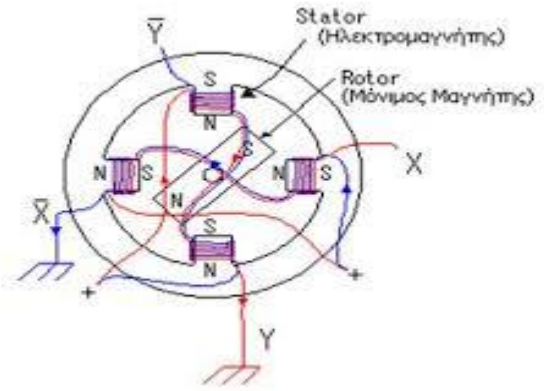
β) τους ασύγχρονους.

Σύγχρονοι ηλεκτρικοί κινητήρες

Σύγχρονη μηχανή ονομάζουμε τη μηχανή αυτή στην οποία η σχέση ανάμεσα στην ταχύτητα περιστροφής **n** του δρομέα και στη συχνότητα της ηλεκτρικής ισχύος **f** είναι σταθερή.

Οι σύγχρονοι κινητήρες έχουν ένα πολυφασικό τύλιγμα στο στάτη και ένα τύλιγμα διέγερσης που διαρρέεται από συνεχές ρεύμα και βρίσκεται στο δρομέα.

Η λειτουργία ενός σύγχρονου κινητήρα στηρίζεται στις δυνάμεις που ασκούνται μεταξύ ετερώνυμων μαγνητικών πόλων. Σε ένα σύγχρονο κινητήρα δύο πόλων, υπάρχει μαγνητικό πεδίο στο δρομέα το όποιο παράγεται από το ρεύμα διέγερσης. Επίσης στο στάτη της μηχανής εφαρμόζεται ένα τριφασικό σύστημα ρευμάτων, το οποίο παράγει στο εσωτερικό της περιστρεφόμενο ομογενές μαγνητικό πεδίο. Έτσι, στο εσωτερικό του κινητήρα υφίστανται δυο πεδία που τείνουν να ευθυγραμμιστούν.



Εικόνα 2.18: Συνδεσμολογία στάτη και ρότορα.

Επειδή, όμως, το πεδίο του στάτη περιστρέφεται συνεχώς, το πεδίο του δρομέα και ο ίδιος ο δρομέας προσπαθεί συνεχώς να τον ακολουθεί. Όσο μεγαλύτερη είναι η γωνία μεταξύ των δύο πεδίων, τόσο μεγαλύτερη είναι η ροπή που ασκεί στο δρομέα το μαγνητικό πεδίο του.

Ένας σύγχρονος ηλεκτρικός κινητήρας περιστρέφεται πάντα με σταθερή ταχύτητα την οποία ονομάζουμε σύγχρονη ταχύτητα περιστροφής και δίνεται από τον παρακάτω τύπο:

$$n_s = 60\frac{f}{p}\left(\sigma\tau\rho/\min\right)$$

όπου: **n** η ταχύτητα σε **rpm**

f η συχνότητα του εναλλασσόμενου ρεύματος

p το σύνολο ζευγών των πόλων

Οι σύγχρονες μηχανές χρησιμοποιούνται κυρίως σαν γεννήτριες για την παραγωγή ηλεκτρικής ενέργειας και σπάνια σαν κινητήρες για να κινούν ορισμένα μηχανήματα ή για να διορθώνουν το συντελεστή ισχύος μιας εγκατάστασης.

<u>Ασύγχρονοι ή επαγωγικοί ηλεκτρικοί κινητήρες</u>

Ονομάζονται ασύγχρονοι κινητήρες, επειδή δεν κινούνται με τη σύγχρονη ταχύτητα περιστροφής.

Η κίνηση στους επαγωγικούς κινητήρες επιτυγχάνεται από τα επαγωγικά ρεύματα που αναπτύσσονται στο δρομέα τους. Αυτά τα επαγωγικά ρεύματα

δημιουργούνται από τη διαφορά της ταχύτητας μεταξύ του στρεφόμενου πεδίου και του δρομέα.

Εάν η ταχύτητα του δρομέα φτάσει την ταχύτητα του πεδίου, τότε δεν έχουμε εμφάνιση επαγωγικών ρευμάτων ούτε δυνάμεων Laplace, και κατά συνέπεια δεν υπάρχει κίνηση. Συνεπώς, ο δρομέας δε στρέφεται ποτέ με τη σύγχρονη ταχύτητα περιστροφής, δηλαδή με την ταχύτητα του στρεφόμενου μαγνητικού πεδίου αλλά πάντοτε με ταχύτητα μικρότερη απ' αυτήν. Η διαφορά αυτή μεταξύ σύγχρονης και ασύγχρονης ταχύτητας ονομάζεται ολίσθηση και δίνεται από την παρακάτω σχέση:

$$S = \frac{n_s - n}{n_s}$$

όπου: n_S = στρ/min του στρεφόμενου μαγνητικού πεδίου.
n = στρ/min του δρομέα.

Οι ασύγχρονοι (επαγωγικοί) κινητήρες διακρίνονται στους:

α) δακτυλιοφόρους (μονοφασικούς ή πολυφασικούς) κινητήρες και

β)στους κινητήρες βραχυκυκλωμένου δρομέα ή κινητήρες κλωβού (μονοφασικοί ή πολυφασικοί).

2.4.3 Κινητήρες universal

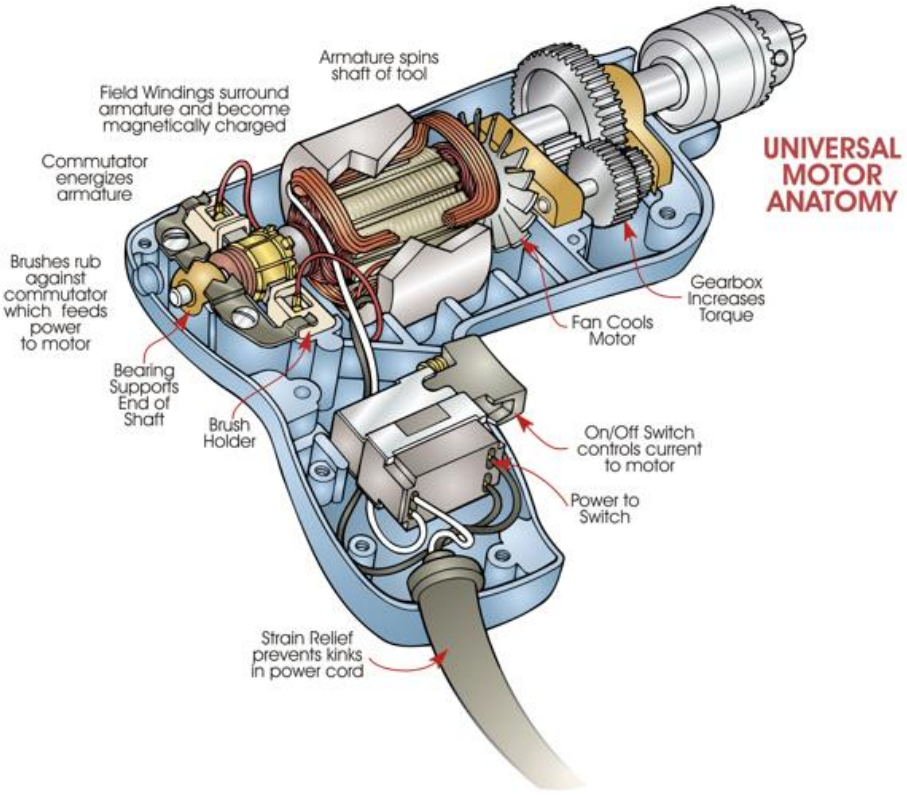
Στα προηγούμενα υποκεφάλαια έγινε αναφορά γενικά στους ηλεκτρικούς κινητήρες, στους κινητήρες συνεχούς και εναλλασσόμενου ρεύματος, στον τρόπο που αυτοί δουλεύουν, στα βασικά τους μέρη καθώς και στη χρήση τους.

Ωστόσο για τη συγκεκριμένη πτυχιακή εργασία χρειαζόμασταν ένα είδος κινητήρα πιο εύχρηστο, πιο ευρέως χρησιμοποιούμενο και πιο οικονομικό. Έτσι χρησιμοποιήσαμε το δραπανοκατάβιδο verto 50G505 (500W) το οποίο αποτελεί έναν universal κινητήρα και ένα παράδειγμα της ευρύτατης χρήσης που αυτοί τυγχάνουν την σήμερον ημέρα.

Εικόνα 2.19: Δραπανοκατσάβιδο verto 50G505 (500W).

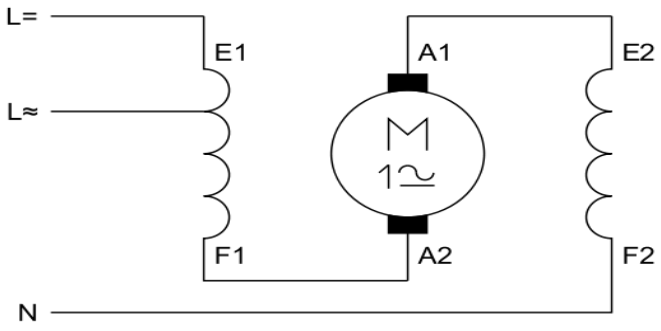
Οι κινητήρες universal είναι κινητήρες διέγερσης σειράς οι οποίοι είναι ικανοί να λειτουργήσουν και σε συνεχές και σε εναλλασσόμενο ρεύμα. Σε εναλλασσόμενο ρεύμα λειτουργούν σε συχνότητες f=25~60 Hz και συνφ=0,5~0,8. Στο συνεχές ρεύμα οι κινητήρες αυτοί αναπτύσσουν ταχύτητα κατά 15% μεγαλύτερη από ‘τι στο εναλλασσόμενο. Οι universal κινητήρες αναπτύσσουν μεγάλη ροπή εκκίνησης και η ταχύτητα τους εξαρτάται από το φορτίο. Όπως είπαμε και παραπάνω η χρήση τους είναι ευρεία, κυρίως σε μικρά εργαλεία και οικιακές συσκευές.

Όταν οι universal κινητήρες λειτουργούν στο συνεχές ρεύμα λειτουργούν όπως και οι κινητήρες συνεχούς ρεύματος διέγερσης σειράς. Όταν όμως λειτουργούν σε εναλλασσόμενο ρεύμα στηρίζονται στο γεγονός ότι αν αλλάξουμε τη διεύθυνση του ρεύματος στο επαγωγικό τύλιγμα αλλά και στο τύλιγμα διέγερσης σειράς τότε η διεύθυνση περιστροφής δεν θα αλλάξει. Έτσι σε κάθε εναλλαγή ημιπεριόδου θα αλλάζει ταυτόχρονα η διεύθυνση του ρεύματος και στα δυο τυλίγματα με αποτέλεσμα ο δρομέας να περιστρέφεται πάντα προς τη μια κατεύθυνση.



Εικόνα 2.20: Ανατομία ενός κινητήρα Universal.

Στους κινητήρες universal το ζύγωμα και οι μαγνητικοί πόλοι είναι φτιαγμένοι από ελάσματα μονωμένα μεταξύ τους για τον περιορισμό των δινορευμάτων και της μαγνητικής υστέρησης. Κατά τα άλλα η κατασκευή τους μοιάζει αρκετά με αυτή των απλών κινητήρων με διέγερση σειράς που τροφοδοτούνται με εναλλασσόμενο ρεύμα (υπάρχουν στο στάτη το βοηθητικό τύλιγμα και το τύλιγμα αντιστάθμισης για την αποφυγή σπινθηρισμών σε ψήκτρες και συλλέκτη) με τη διαφορά ότι έχουν ένα πρόσθετο ακροδέκτη στο τύλιγμα διέγερσης.



Εικόνα 2.21: Διάγραμμα λειτουργίας ενός κινητήρα Universal.

Όταν ο κινητήρας τίθεται υπό συνεχές ρεύμα τότε συνδέεται όλο το τύλιγμα διέγερσης, ενώ όταν χρησιμοποιούμε εναλλασσόμενο ρεύμα συνδέουμε το τύλιγμα διέγερσης όπως στο παραπάνω σχήμα.

Σε ένα κινητήρα universal μπορούμε να αλλάξουμε τη φορά περιστροφής απλά αλλάζοντας τη διεύθυνση του ρεύματος στο τύλιγμα της διέγερσης σειράς και κρατώντας σταθερή τη διεύθυνση του ρεύματος στο επαγωγικό τύλιγμα (ή το αντίθετο).

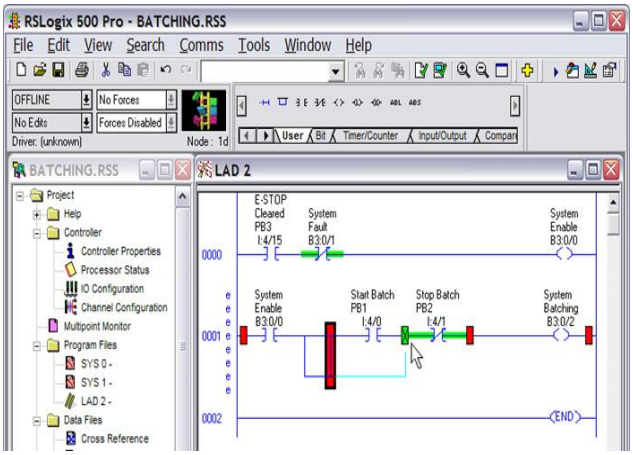
Τέλος μπορούμε να έχουμε βηματικό έλεγχο της ταχύτητας ενός τέτοιου κινητήρα χρησιμοποιώντας ένα κύκλωμα θυρίστορ όπως εξάλλου γίνεται και με πολλές οικιακές συσκευές.

ΚΕΦΑΛΑΙΟ 3 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

3.1 Εισαγωγή

Όταν μιλάμε για προγραμματισμό υπολογιστών εννοούμε το σύνολο των διαδικασιών που κάνουμε για να συντάξουμε ένα υπολογιστικό πρόγραμμα. Το πρόγραμμα αυτό μας βοηθάει συνήθως στην αυτοματοποιημένη εκτέλεση εργασιών ή την επίλυση κάποιου υπολογιστικού προβλήματος από έναν υπολογιστή, στον έλεγχο του προγράμματος για την επαλήθευση της ακρίβειας και της ορθότητάς του (αποσφαλμάτωση) και στην προπαρασκευή των οδηγιών με τις οποίες ένας υπολογιστής θα εκτελέσει τις εντολές που καθορίζονται στις προδιαγραφές του προγράμματος. Θεμελιώδη ρόλο στον υπολογιστικό προγραμματισμό διαδραματίζουν οι χιλιάδες διαφορετικές γλώσσες προγραμματισμού για τις οποίες θα μιλήσουμε όμως περισσότερο παρακάτω.

Όπως μπορούμε να καταλάβουμε πλέον ο προγραμματισμός υπάρχει παντού μέσα στη ζωή μας. Καταρχήν ο προγραμματισμός είναι αναγκαίος για την σωστή χρήση οποιουδήποτε υπολογιστή, smart phone, smart tv αλλά και των περισσότερων ηλεκτρονικών συσκευών τελευταίας γεννιάς. Δεν χρησιμοποιούμε όμως τον προγραμματισμό μόνο σε οικιακή χρήση, η πιο σημαντική χρήση του προγραμματισμού είναι στον επαγγελματικό τομέα. Όλες οι επιχειρήσεις χρησιμοποιούν προγράμματα για την καλύτερη λειτουργία τους, τα οποία εξαγοράζονται μάλιστα σε πολύ υψηλές τιμές. Και οι κυβερνήσεις των περισσότερων ανεπτυγμένων κρατών όμως κάνουν χρήση τέτοιων προγραμμάτων. Στο δικό μας επάγγελμα, αυτό της ηλεκτρολογίας χρησιμοποιούμε αυτές τις γνώσεις για να δουλέψουμε σωστά με τα plc, για τη χρήση των μικροελεγκτών κ.α.



Εικόνα 3.1: προγραμματισμός ενός plc

Γλώσσα προγραμματισμού

Γλώσσα προγραμματισμού λέγεται μια τεχνητή γλώσσα που μπορεί να χρησιμοποιηθεί για τον έλεγχο μιας μηχανής, συνήθως ενός υπολογιστή. Οι γλώσσες προγραμματισμού χρησιμοποιούνται για να διευκολύνουν την οργάνωση και διαχείριση πληροφοριών, αλλά και για την ακριβή διατύπωση αλγορίθμων.
Οι γλώσσες προγραμματισμού μπορούν να κατηγοριοποιηθούν με βάση:

Α) τον τρόπο οργάνωσης του προγράμματος

Β) τον στόχο που έχει η γλώσσα

Γ) τον τρόπο που περιγράφουν το ζητούμενο αποτέλεσμα

Στην πρώτη περίπτωση προκύπτουν κατηγορίες όπως:

- Διαδικαστικές γλώσσες (procedural) όπου το πρόγραμμα είναι οργανωμένο σε διαδικασίες, που αποτελούνται από σειρές εντολών που περιγράφουν αλγορίθμους.
- Αντικειμενοστραφείς γλώσσες (object-oriented) όπου το πρόγραμμα είναι οργανωμένο σε αντικείμενα. Ένα αντικείμενο είναι μια μονάδα που αποτελείται από την περιγραφή κάποιων δεδομένων και την περιγραφή των αλγορίθμων που τα επεξεργάζονται. Ένα αντικειμενοστραφές πρόγραμμα αποτελείται από διάφορα αντικείμενα που αλληλεπιδρούν μεταξύ τους.
- Συναρτησιακές γλώσσες (functional) όπου οι υπολογισμοί εκφράζονται ως εφαρμογές μαθηματικών συναρτήσεων, σε αντίθεση με τα άλλα είδη προγραμματισμού όπου οι υπολογισμοί εκφράζονται ως σειρές εντολών.

Στην περίπτωση που η κατηγοριοποίηση των γλωσσών προγραμματισμού γίνει με βάση το στόχο που έχει η γλώσσα, υπάρχουν οι παρακάτω κατηγορίες:

1. Γλώσσες γενικής χρήσης. Σε αυτήν την κατηγορία ταξινομούνται γλώσσες που δημιουργήθηκαν για τον προγραμματισμό γενικών εφαρμογών.
2. Γλώσσες προγραμματισμού συστημάτων, που χρησιμοποιούνται συνήθως για τον προγραμματισμό λειτουργικών συστημάτων ή οδηγών (drivers) υλικού.
3. Γλώσσες σεναρίων (scripting). Αυτές οι γλώσσες χρησιμοποιούνται συνήθως για τη γρήγορη ανάπτυξη μικρών προγραμμάτων.
4. Γλώσσες ειδικών εφαρμογών. Σε αυτή την κατηγορία ανήκουν γλώσσες που αναπτύχθηκαν ειδικά για μια συγκεκριμένη εφαρμογή.

5. Παράλληλες γλώσσες. Στη συγκεκριμένη κατηγορία ταξινομούνται γλώσσες που επιτρέπουν τη ανάπτυξη παράλληλων προγραμμάτων, όπου πολλές εντολές εκτελούνται ταυτόχρονα σε πολλούς υπολογιστές, έτσι ώστε το τελικό αποτέλεσμα να προκύψει γρηγορότερα.

Τέλος, στην περίπτωση που η κατηγοριοποίηση γίνεται με βάση τον τρόπο που περιγράφεται το ζητούμενο, υπάρχουν οι παρακάτω κατηγορίες:

1. Προστακτικές γλώσσες προγραμματισμού (imperative).
2. Δηλωτικές γλώσσες προγραμματισμού (declarative).

Στη δικιά μας εφαρμογή θα χρησιμοποιήσουμε πολλές γλώσσες προγραμματισμού έτσι ώστε να καταφέρουμε το επιθυμητό αποτέλεσμα. Οι γλώσσες αυτές περιλαμβάνουν την ajax, html, java, php και visual basic.

3.2 VISUAL BASIC

3.2.1 Μεταβλητές

Οι μεταβλητές είναι οι θέσεις στην κεντρική μνήμη του ηλεκτρονικού υπολογιστή στις οποίες μπορούν να αποθηκευτούν δεδομένα. Τα δεδομένα αυτά μπορούν να αλλάζουν ανάλογα με τις απαιτήσεις μας. Μπορούμε να ορίσουμε το όνομα της μεταβλητής με την εντολή **DIM** ακολουθώντας όμως τους εξής κανόνες:

- Το όνομα αποτελείται από λατινικούς χαρακτήρες.
- Ο πρώτος χαρακτήρας είναι πάντα γραμμα.
- Δεν μπορούμε να χρησιμοποιήσουμε το κενό.

Η visual basic υποστηρίζει τις εξής κατηγορίες μεταβλητών:

1. **Boolean,** η οποία δέχεται δυο εναλλακτικές τιμές(true ή false), και δεσμεύει δυο bytes.
2. **Char,** δέχεται τιμές από το 0 έως το 65535. Με βάση την κωδικοποίηση Unicode(κωδικοποιημένη αναπαράσταση χαρακτήρων σε δυαδική μορφή) οι τιμές αυτές μπορούν να αντιστοιχούν σε κάποιο χαρακτήρα η σύμβολο, και δεσμεύουν 2 bytes.
3. **Short,** που δέχεται ακέραιους αριθμούς από το -32.768 έως το 32.767 και δεσμεύει 2 bytes.

4. **Integer,** στην οποία αποθηκεύονται ακέραιοι αριθμοί από το -2.147.483.648 έως το 2.147.483.647 και δεσμεύει 4 bytes.
5. **Byte,** αποθηκεύει φυσικούς αριθμούς από 0 έως το 255 και καταλαμβάνει στη μνήμη 1 byte.
6. **Long,** που αποθηκεύονται ακέραιοι αριθμοί από το -9223372036854775808 έως το 9223372036854775807, καταλαμβάνοντας 8 bytes.
7. **Single,** στην οποία αποθηκεύονται δεκαδικοί αριθμοί, δεσμεύοντας 4 bytes.
8. **Double,** που αποθηκεύονται δεκαδικοί αριθμοί δεσμεύοντας 8 bytes.
9. **Decimal,** όπου αποθηκεύονται δεκαδικοί αριθμοί δεσμεύοντας 14 bytes.
10. **Date,** για την αποθήκευση ημερομηνιών, δεσμεύει 8 bytes.
11. **String,** που μπορούμε να αποθηκεύσουμε αλφαριθμητικά δεδομένα και καταλαμβάνουν μνήμη ανάλογη του αριθμού των χαρακτήρων που χρησιμοποιούμε.
12. **Object,** που χρησιμοποιούνται για την αποθήκευση διευθύνσεων που αφορούν αντικείμενα και καταλαμβάνουν 4 bytes.

Παράδειγμα δήλωσης μεταβλητών:
DIM name as String
(δήλωση αλφαριθμητικής μεταβλητής)
DIM x, y as Integer
(δήλωση του x και του y ως ακέραιους αριθμούς)

Στη visual basic μπορούμε επίσης να φτιάξουμε τον δικό μας τύπο μεταβλητής ανάλογα με τους ήδη υπάρχοντες τύπους. Για να κάνουμε κάτι τέτοιο θα πρέπει να χρησιμοποιήσουμε την εντολή type.
π.χ.

```
type diastaseis_ogkou

x as single
y as single
z as single

end type
```

Αφού ορίσουμε τον τύπο της μεταβλητής που θέλουμε, μπορούμε να δηλώσουμε μια μεταβλητή με την εντολή DIM.

Dim sxhma as diastaseis_ogkou

Τέλος με την εντολή **const** μπορούμε να αποθηκεύσουμε μια τιμή η οποία θα παραμένει σταθερή όσο εκτελείται το πρόγραμμα.

π.χ.

const pi as single=3,14

3.2.2 τελεστές

αριθμητικοί τελεστές

Χρησιμοποιούμε σε ένα πρόγραμμα τους αριθμητικούς τελεστές για να επιτελέσουμε αριθμητικές πράξεις και να κατασκευάσουμε αριθμητικές παραστάσεις. Η σειρά υπολογισμού των πράξεων γίνεται όπως και στα μαθηματικά. Οι αριθμητικοί τελεστές στη visual basic είναι οι εξής:

τελεστής	**Αριθμητική πράξη**
^	Ύψωση σε δύναμη
*	πολλαπλασιασμός
/	διαίρεση
\	Ακέραια διαίρεση
+	πρόσθεση
-	αφαίρεση
mod	Υπόλοιπο ακέραιας διαίρεσης

Τελεστές σύγκρισης

Οι τελεστές σύγκρισης χρησιμοποιούνται στον προγραμματισμό για να συγκρίνουμε σταθερές και μεταβλητές. Οι τελεστές σύγκρισης στη visual basic είναι οι εξής:

τελεστής	**Λογική σχέση**
<	Μικρότερο από
>	Μεγαλύτερο από
=	ίσο
>=	Μεγαλύτερο από ή ίσο
<=	Μικρότερο από ή ίσο
<>	Διάφορο από

Λογικοί τελεστές

Οι λογικοί τελεστές χρησιμοποιούνται για να κάνουμε λογικές πράξεις. Οι πράξεις αυτές μπορεί να έχουν δυο αποτελέσματα, αληθής ή ψευδής. Οι λογικές πράξεις στη visual basic είναι οι εξής:

τελεστής	**Λογική πράξη**
and	and
or	or

not	not
Hor	xor

Επίσης στη visual basic υπάρχουν οι λογικές πράξεις orelse και andalso.Στην andalso έχουμε αληθές αποτέλεσμα μόνο όταν και οι δυο λογικές μας προτάσεις είναι αληθείς, ενώ στην orelse έχουμε ψευδές αποτέλεσμα μόνο όταν και οι δυο λογικές μας προτάσεις είναι ψευδείς.

Τελεστές σύνθεσης αλφαριθμητικών δεδομένων

Οι τελεστές & και + είναι οι τελεστές που χρησιμοποιούμε για να προσθέσουμε αλφαριθμητικά δεδομένα.

π.χ.

```
dim A, B, C as string

A=τει
B=πειραια

C=A&B
```

Στο παραπάνω παράδειγμα προσθέσαμε δυο μεταβλητές έτσι ώστε να σχηματίσουμε τη φράση τει 'πειραια'.

Τελεστές ανάθεσης

Οι τελεστές ανάθεσης χρησιμοποιούνται για την αποθήκευση τιμών σε μια μεταβλητή. Η visual basic υποστηρίζει τους εξής τελεστές ανάθεσης:

1. **=**, υπολογίζει την τιμή μιας παράστασης και την καταχωρίζει στην μεταβλητή.
2. **^=**, υπολογίζει την τιμή μιας παράστασης, υψώνει τη μεταβλητή στην τιμή της παράστασης και καταχωρεί το αποτέλεσμα στη μεταβλητή.
3. ***=**, πολλαπλασιάζει την τιμή μιας παράστασης με μια μεταβλητή και καταχωρεί το αποτέλεσμα στη μεταβλητή.
4. **/=**, διαιρεί την τιμή μιας παράστασης με μια μεταβλητή και καταχωρεί το αποτέλεσμα στη μεταβλητή.
5. **+=**, προσθέτει την τιμή μιας παράστασης με την τιμή μιας μεταβλητής και καταχωρεί το αποτέλεσμα στη μεταβλητή.
6. **-=**, αφαιρεί την τιμή μιας παράστασης από την τιμή μιας μεταβλητής και καταχωρεί το αποτέλεσμα στη μεταβλητή.
7. **&=**, προσθέτει την τιμή μιας παράστασης και μιας μεταβλητής και καταχωρεί το αποτέλεσμα στη μεταβλητή. Ισχύει μόνο για αλφαριθμητικά δεδομένα.

π.χ.
```
dim X as single=5.0
dim Y as single=10.0
X+=Y
```
Στο παραπάνω παράδειγμα το X παίρνει την τιμή 15.

3.2.3 Εντολές

Option compare [binary/text]

Η εντολή αυτή χρησιμοποιείται για να συγκρίνουμε strings, όταν γράφουμε option compare binary η σύγκριση γίνεται σε επίπεδο bit, ενώ όταν χρησιμοποιούμε το option compare text γίνεται σύγκριση σε επίπεδο χαρακτήρων.

Let

Με την εντολή αυτή μπορούμε να αναθέτουμε τιμές σε μεταβλητές. Μπορούμε επίσης αν θέλουμε να παραλείπουμε τη λέξη let.

π.χ.
```
let X=Y+1*2
```
ή αλλιώς
```
X=Y+1*2
```
Loop

Στη visual basic υπάρχουν τρεις εντολές επανάληψης, η for/next, η while/wend και η do/loop.Απο αυτές τις εντολές η πιο χρήσιμη είναι η do/loop αφού υπερκαλύπτει όλες τις περιπτώσεις εφαρμογής των υπόλοιπων επαναληπτικών εντολών και μπορούμε με τη βοήθεια της να αντιμετωπίσουμε ακόμα και τις πιο σύνθετες περιπτώσεις αλγορίθμων με επαναληπτικές διαδικασίες. Η εντολή αυτή έχει τέσσερις διαφορετικές μορφές για να μας βοηθήσει να διαμορφώσουμε το πρόγραμμά μας ανάλογα με τις ανάγκες μας.

1. **Do while/loop**. Οσο ικανοποιείται η συνθήκη που ακολουθεί τη κωδική λέξη while εκτελούνται οι εντολές που βρίσκονται μέσα στο βρόγχο do while/loop.Υπαρχει ωστόσο περίπτωση να μην εκτελεστούν ποτέ οι εντολές, αν δεν ικανοποιείται η συνθήκη.

2. **Do/loop while**. Όσο ικανοποιείται η συνθήκη που ακολουθεί τη κωδική λέξη while εκτελούνται οι εντολές που βρίσκονται μέσα στο βρόγχο do/loop while.Οι εντολές αυτές θα εκτελεστούν τουλάχιστον μια φορά.

3. **Do until/loop**. Εκτελούνται οι εντολές του βρόγχου do until/loop μέχρι να ικανοποιηθεί η συνθήκη που βρίσκεται μετά την κωδική λέξη until. Υπάρχει η περίπτωση να μην εκτελεστούν ποτέ οι εντολές.

4. **Do/loop until**. Εκτελούνται οι εντολές του βρόγχου do/loop until μέχρι να ικανοποιηθεί η συνθήκη που ακολουθεί τη κωδική λέξη until.Οι εντολές θα εκτελεστούν τουλάχιστον μια φορά.

π.χ.

```
X=0
Do while X<5
X=X+1
Loop
```

ή

```
X=0
Do
X=X+1
Loop while X<5
```

If

Η εντολή if είναι η κύρια εντολή ελέγχου στη visual basic.Με αυτή την εντολή μπορούμε να προγραμματίσουμε τον ηλεκτρονικό υπολογιστή έτσι ώστε να ελέγχει την ικανοποίηση ή όχι κάποιων συνθηκών και να εκτελεί τις εντολές που θέλουμε, ανάλογα με την περίπτωση. Οι δυο πιο χρήσιμες μορφές της εντολής if είναι οι εξής:

1. **If συνθήκη then εντολή**

π.χ.

```
if bathmos >=9 then message = ‘’μπραβο μου’’
```

Στο παραπάνω παράδειγμα ο υπολογιστής ελέγχει τη συνθήκη, αν αυτή αληθεύει τότε εμφανίζει το μήνυμα, ενώ αν η συνθήκη είναι ψευδής το πρόγραμμα συνεχίζει στην επόμενη εντολή.

2. **If συνθήκη 1 then**

Εντολές

Else if συνθήκη 2 then

Εντολές

Else if συνθήκη 3 then

Εντολές

Else

Εντολές

End if

π.χ.

```
if bath >= 8 then
message = ‘’Άριστα’’
else if bath >= 5 and bath <8 then
message = ‘’προβιβάζεται’’
else
message = ‘’χαχα κοπηκες’’
end if
```

Στο παραπάνω παράδειγμα ο υπολογιστής ελέγχει την κάθε συνθήκη, τη μια μετά την άλλη, εμφανίζοντας το κατάλληλο μήνυμα.

Rem

Η εντολή rem χρησιμοποιείται στη visual basic για να γράψουμε σχόλια μέσα στο πρόγραμμα, τα σχόλια αυτά αγνοούνται από το πρόγραμμα. Μπορούμε επίσης να παραλείψουμε την κωδική λέξη rem και να χρησιμοποιήσουμε το μόνο εισαγωγικό.
π.χ.

```
rem σχόλιαααααα
```

ή

```
‘σχόλιαααααααααα
```

Συναρτήσεις

Οι συναρτήσεις στη visual basic είναι ένα σύνολο ήδη προγραμματισμένων εφαρμογών που είναι ενσωματωμένες στη γλώσσα προγραμματισμού. Μπορούμε να καλέσουμε αυτές τις συναρτήσεις γράφοντας απλά το όνομά τους. Κάποιες από τις πιο συχνές συναρτήσεις που μπορούμε να συναντήσουμε είναι οι εξής:

1) *System.math.abs(τιμή).*Επιστρέφει την απόλυτη τιμή της τιμής.
2) *System.math.cos(τιμή).*Επιστρέφει το συνημίτονο της τιμής.
3) *System.math.sin(τιμή).*Επιστρέφει το ημίτονο της τιμής.
4) *System.math.tan(τιμή).*Επιστρέφει την εφαπτομένη της τιμής.
5) *System.math.log(τιμή).*Επιστρέφει το λογάριθμο της τιμής.

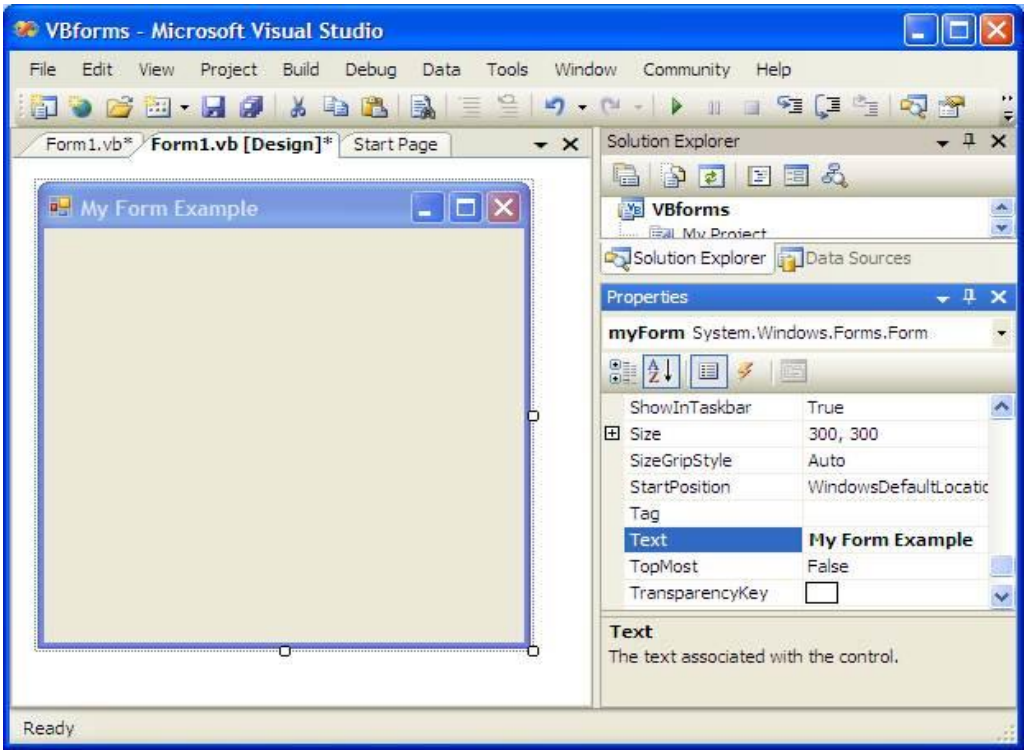
6) Συναρτήσεις μετατροπής. Χρησιμοποιούνται για τη μετατροπή ενός τύπου δεδομένων σε ένα άλλο.π.χ. *cbool* επιστρέφει Boolean, *cbyte* επιστρέφει byte, *cint* επιστρέφει integer.

7) *Msgbox().*Εμφάνιση παράθυρου με μήνυμα και πλήκτρα.

8) *Inputbox().*Εμφάνιση παράθυρου με πλαίσιο κειμένου.

3.2.4 Φόρμα, ελεγκτήρια και συμβάντα

Με την εντολή class μπορούμε να ορίσουμε μια τάξη, μέσα στην οποία ορίζουμε τις μεταβλητές και τις λειτουργίες της σε μορφή υπορουτίνων. Στη visual basic κάθε φόρμα(form) είναι και μια τάξη(και ένα παράθυρο της εφαρμογής).



Εικόνα 3.2: Ένα form σε περιβάλλον visual basic.

Στην παραπάνω εικόνα μπορούμε να δούμε ένα παράδειγμα μιας φόρμας σε περιβάλλον visual basic.Μέσα σε αυτό το παράθυρο μπορούμε να βάλουμε τα ελεγκτήρια(ετικέτες, buttons κλπ) που θέλουμε. Τα ελεγκτήρια αυτά είναι στην ουσία κάποια αντικείμενα(σε μορφή υπορουτίνων) που μας παρέχει το περιβάλλον της visual basic. Στη visual basic υπάρχουν πολλά ελεγκτήρια χρήσιμα για τις εφαρμογές μας, κάποια από τα οποία είναι: τα πλήκτρα εντολών(button), οι ετικέτες(label), τα πλαίσια κειμένου(textbox), τα πλαίσια εικόνας(picturebox) κα.Ο

κώδικας των ελεγκτηρίων μοιάζει συνήθως κάπως έτσι:

```
PrivateSub αντικείμενο_συμβάν(byVal sender As System.object,ByVal e As
System.EventArgs)Handles αντικείμενο. Συμβάν
…
Εντολές
…
End Sub
```

Στον παραπάνω τρόπο σύνταξης κώδικα βλέπουμε ότι πρέπει πρώτα να γράψουμε ποιο είναι το αντικείμενο που θέλουμε να χρησιμοποιήσουμε(π.χ. button) και μετά να ακολουθήσει το συμβάν του. Τα συμβάντα στη visual basic είναι υπορουτίνες ενσωματωμένες στην τάξη του παραθύρου στο οποίο εμπεριέχονται τα αντικείμενα. Όπως γίνεται κατανοητό υπάρχουν πάρα πολλά τέτοια συμβάντα, κάποια από τα πιο συχνά χρησιμοποιούμενα σε όλα τα ελεγκτήρια είναι τα παρακάτω:

1) *Click*.Αυτο το συμβάν περιλαμβάνει τις ενέργειες που πρέπει να γίνουν όταν ο χρήστης πιέσει με το ποντίκι το αντικείμενο.

2) *Double click*.Περιλαμβάνει τις ενέργειες που πρέπει να γίνουν όταν ο χρήστης πιέσει δυο γρήγορες φορές το πλήκτρο του ποντικιού.

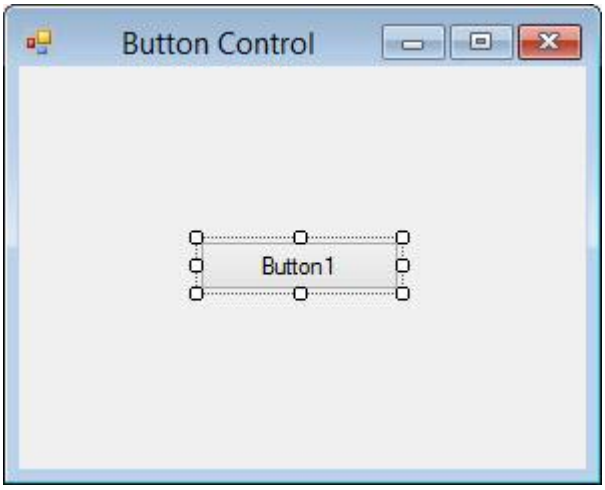
3) *Keypress*.Περιλαμβάνει τις ενέργειες που πρέπει να γίνουν όταν ο χρήστης πιέσει κάποιο πλήκτρο στο πληκτρολόγιό του.

4) *Keyup*.Περιλαμβάνει τις ενέργειες που πρέπει να γίνουν όταν ο χρήστης σταματήσει να πιέζει το πλήκτρο στο πληκτρολόγιο του.

5) *Keydown*.Περιλαμβάνει τις ενέργειες που θα πρέπει να γίνουν όταν ο χρήστης πιέζει κάποιο πλήκτρο στο πληκτρολόγιό του.

6) *Enter*.Περιλαμβάνει τις ενέργειες που θα πρέπει να γίνουν όταν ο χρήστης πιέσει το enter.

Στο παρακάτω παράδειγμα έχουμε ένα ελεγκτήριο πλήκτρο(button) το οποίο έχουμε προγραμματίσει έτσι ώστε να εκτελούνται κάποιες εντολές όταν κάνουμε click σε αυτό.



Εικονα 3.3: Ένα button σε περιβαλλον visual basic.

```
Private sub button1_click(byval sender as system.object,byval e as
system.eventargs)handles button1.click
…
Εντολές
…
End sub
```

Προγραμματιστικές μονάδες(modules)

Όταν ορίζουμε μεταβλητές και εντάσσουμε υπορουτίνες σε μια φόρμα, τότε όλα αυτά ισχύουν μόνο για τη συγκεκριμένη φόρμα μας. Υπάρχει όμως η πιθανότητα να θέλουμε να κατασκευάσουμε μια εφαρμογή η οποία θα αποτελείται από πολλές φόρμες, και να θέλουμε να χρησιμοποιήσουμε τις ίδιες μεταβλητές και υπορουτίνες σε όλες τις φόρμες. Αυτό μπορούμε να το κάνουμε φτιάχνοντας μια προγραμματιστική μονάδα. Ένα module δηλαδή μπορεί να έχει ενταγμένες μέσα του όλες τις υπορουτίνες και τις μεταβλητές που χρησιμοποιούμε σε πολλές διαφορετικές φόρμες.

3.3 Η ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ HTML

3.3.1 Εισαγωγή

Η html είναι η κύρια περιγραφική γλώσσα (γλώσσα σήμανσης ή αλλιώς markup language, δεν θεωρείται γλώσσα προγραμματισμού) που χρησιμοποιείται για τη οικοδόμηση των ιστιοσελίδων (συνήθως καθορίζουν τον τρόπο εμφάνισης των στοιχείων μιας ιστοσελίδας και τον τρόπο κλήσης άλλων αρχείων ή εφαρμογών). Η html (HyperText Markup Language) γράφεται υπό τη μορφή στοιχείων. Τα στοιχεία αυτά αποτελούνται από ετικέτες (tags) , οι οποίες λειτουργούν συνήθως ανά ζεύγη , έχουμε δηλαδή μια ετικέτα έναρξης και μια ετικέτα λήξης (για παράδειγμα <h1> … <h1>). Ανάμεσα σε αυτές τις ετικέτες τοποθετείται το κείμενο , η εικόνα , ο πίνακας ή ότι άλλο χρειάζεται. Εμείς στο δικό μας πρόγραμμα χρησιμοποιούμε την html για

να ταξινομήσουμε με τον κατάλληλο τρόπο τα button της ιστιοσελίδας που θα χρησιμοποιήσουμε για τον ασύρματο χειρισμό της συσκευής μας αλλά και για τη διακόσμηση της.

3.3.2 Στοιχεία και ετικέτες της html

Καταρχήν στην γλώσσα σήμανσης html όλα τα έγγραφα πρέπει να ξεκινάνε με την ετικέτα <!DOCTYPE html> .
Το κείμενο που γράφουμε ξεκινάει με την ετικέτα <html> και τελειώνει με την </html> .
Το εμφανές κομμάτι του εγγράφου μας (αυτά που βλέπουμε δηλαδή στην ιστοσελίδα) ξεκινάει με την ετικέτα <body> και τελειώνει με την </body> .
Με βάση τα παραπάνω ένα πρόγραμμα html θα έχει την εξής μορφή:

```
<!DOCTYPE html>
<html>
<body>
…
…
…
</body>
</html>
```

Δίπλα στην ετικέτα έναρξης html μπορούμε να δηλώσουμε τη γλώσσα της ιστοσελίδας μας ως εξής:

```
<html lang="en-US">
```

Αυτό έχει σημασία για τις μηχανές αναζήτησης.
Μπορούμε να δούμε το πρόγραμμα μιας ιστοσελίδας απλά πατώντας δεξί κλικ πάνω σε αυτήν και επιλέγοντας ‘’View Page Source’’ στο google chrome ή την αντίστοιχη επιλογή σε άλλους περιηγητές .

Τίτλοι, επικεφαλίδες

Οι επικεφαλίδες (τίτλοι) της γλώσσας ξεκινάνε με την ετικέτα <h1> και τελειώνουν με την </h1> (ξεκινάμε από <h1> και φτάνουμε έως και το <h6>).
Π.χ.

```
<h1>επικεφαλίδα 1</h1>
<h2>επικεφαλίδα 2</h2>
<h3>επικεφαλίδα 3</h3>
```

Οι επικεφαλίδες είναι πολύ σημαντικές γιατί με βάση αυτές βρίσκονται οι ιστοσελίδες μας από τις μηχανές αναζήτησης . Το <h1> είναι η κύρια επικεφαλίδα μας , το <h2> η δεύτερη σημαντικότερη , και φτάνουμε μέχρι την <h6> τη λιγότερο σημαντική.
Μπορούμε να τραβήξουμε μια οριζόντια γραμμή για να διαχωρίσουμε το

περιεχόμενο των κειμένων μας με το στοιχείο (εντολή , ετικέτα) <hr> .

Π.χ.
```
<h1>επικεφαλίδα 1</h1>
<hr>
<p>παράγραφος                    No                    1</p>
```

Παράγραφοι

Οι παράγραφοι κειμένου ξεκινάνε με την ετικέτα και τελειώνουν με την .
(Όσα κενά και αν βάλουμε ανάμεσα σε δυο λέξεις μέσα σε μια παράγραφο , στην ιστιοσελίδα θα φαίνεται σαν ένα κενό .)

Π.χ.
```
<p>παράγραφος                    No                    1</p>
<p>παράγραφος No 2</p>
```

Επειδή στην html όταν πατάμε το πλήκτρο enter αυτό δεν φαίνεται στην ιστοσελίδα μας , υπάρχει στη γλώσσα αυτή το στοιχείο
 το οποίο μας επιτρέπει να πηγαίνουμε στην επόμενη γραμμή (line break).

Π.χ.
```
<p>grammh<br>grammh<br>grammh</p>
```

Αν θέλουμε να γράψουμε ένα κείμενο στο οποίο να μπορούν να διατηρηθούν οι όλες οι γραμμές και τα κενά τα οποία προσθέσαμε , τότε πρέπει να χρησιμοποιήσουμε τις ετικέτες **<pre>** και **</pre>** (για αρχή και τέλος αντίστοιχα).

Π.χ.
```
<pre>
    leksi          leksi        grammh


Grammh
leksi
</pre>
```

Άλλα στοιχειά που μπορεί να μας χρησιμεύουν όταν γράφουμε ένα κείμενο είναι το στοιχείο <q> με το οποίο μπορούμε να βάλουμε ένα κείμενο σε εισαγωγικά, το <blockquote> με το οποίο μπορούμε να κάνουμε μια παραπομπή, το <abbr> που μας βοηθάει με τις συντομογραφίες και τα ακρώνυμα και το <address> με το οποίο μπορούμε να γράψουμε τα στοιχεία του συγγραφέα κάτω από ένα κείμενο.
Απλό κείμενο παραγράφου:

> Αυτό είναι ένα κείμενο που γράφτηκε με τη βοήθεια του στοιχείου της html <blockquote>, το κείμενο αυτό θα πρέπει να ξεκινάει με την ετικέτα έναρξης

<blockquote> και να τελειώνει με την ετικέτα λήξης </blockquote>.”Αυτό το κείμενο γράφτηκε με τη βοήθεια του στοιχείου <q>.”

Ακολουθεί ο κώδικας του παραδείγματος.

```
<p> Απλό κείμενο παραγράφου:</p>
<blockquote >
Αυτό είναι ένα κείμενο που γράφτηκε με τη βοήθεια του στοιχείου της html
<blockquote>, το κείμενο αυτό θα πρέπει να ξεκινάει με την ετικέτα έναρξης
<blockquote> και να τελειώνει με την ετικέτα λήξης </blockquote>. <q>Αυτό το
κείμενο γράφτηκε με τη βοήθεια του στοιχείου <q>. </q>
</blockquote>
```

Links

Αν θέλουμε να χρησιμοποιήσουμε links στην ιστιοσελίδα μας τότε θα πρέπει να χρησιμοποιήσουμε τις ετικέτες <a> και (για αρχή και τέλος αντίστοιχα). Τον προορισμό του link θα πρέπει να τον γράψουμε ως href=΄΄προορισμός΄΄ μέσα στην πρώτη ετικέτα.

Π.χ.

```
<a href="http://www.en.wikipedia.org/wiki/main_page">wikipedia</a>
```

Μπορούμε επίσης να χρησιμοποιήσουμε την παράμετρο target έτσι ώστε το link μας να ανοίξει σε μια άλλη σελίδα . Για να γίνει αυτό θα πρέπει να γράψουμε δίπλα από την παράμετρο τη λέξη blank .

Π.χ.

```
<a href="http://www.en.wikipedia.org/wiki/main_page"  target="_blank">wikipedia<
/a>
```

Εικόνες

Μπορούμε να βάλουμε εικόνες στην εφαρμογή μας με την ετικέτα <img> . Η ετικέτα αυτή έχει τις εξής παραμέτρους: src για την πηγή του αρχείου , alt για να γράψουμε κάποιο εναλλακτικό κείμενο για την εικόνα , width και height για τις διαστάσεις της εικόνας.

Π.χ.

```
<img src="kati.jpg" alt="kati" width="104" height="142">
```

Ένας πιο σίγουρος τρόπος για να γράψουμε τις διαστάσεις τις εικόνας στο πρόγραμμα μας είναι να χρησιμοποιήσουμε την παράμετρο **style** .

Π.χ.

```
<img src="kati.gif" alt="kati" style="width:128px;height:128px;">
```

Μπορούμε επίσης να χρησιμοποιήσουμε και μια εικόνα σαν link απλά βάζοντας μέσα

στο στοιχείο του link το στοιχείο της εικόνας .

Π.χ.

```
<a href="link.gr">
  <img src="eikona.gif" alt="eikona" style="width:42px;height:42px;border:0;">
</a>
```

Με το στοιχείο <cite> μπορούμε να προσθέσουμε στοιχεία της εικόνας κάτω από αυτή σε πλάγια γραφή.

Π.χ.

```
<img src="kati.gif" alt="kati" style="width:128px;height:128px;">
<p><cite>to kati</cite> by emena. Painted in 3000 pX.</p>
```

Μεταδεδομένα (meta data)

Τα μεταδεδομένα είναι τα δεδομένα που χρησιμοποιούμε στο πρόγραμμα αλλά δεν είναι εμφανή στην ιστοσελίδα μας. Ανάμεσα στις ετικέτες <html> και <body> βρίσκουμε συνήθως τα μεταδεδομένα τα οποία ξεκινάνε με την ετικέτα έναρξης <head> και τελειώνουν με την ετικέτα λήξης </head>. Στις καινούριες εκδόσεις html το στοιχείο αυτό μπορεί να παραλειφθεί , τα μεταδεδομένα θα είναι απλά ότι προσθέσουμε εμείς ανάμεσα στις ετικέτες αρχής <html> και <body> .
Το στοιχείο **<title>** ορίζει έναν τίτλο στην καρτέλα του προγράμματος περιήγησης , παρέχει ένα τίτλο στη σελίδα όταν αυτή προσθέτεται στα αγαπημένα και βοηθάει τις μηχανές αναζήτησης να βρουν τη ιστιοσελίδα .

Π.χ.

```
<!DOCTYPE html>
<html>
<title>titlos</title>

<body>
periexomeno                                                    selidas.....
</body>

</html>
```

Το στοιχείο **<meta>** χρησιμοποιείται για να προσδιορίσει μεταδεδομένα όπως είναι διάφορες λέξεις κλειδιά για τις μηχανές αναζήτησης , περιγραφές της ιστιοσελίδας , τον συγγραφέα της ιστιοσελίδας κ.λ.π. Στο παρακάτω παράδειγμα έχουμε κάνει χρήση του στοιχείου <meta> έτσι ώστε να γίνεται ανανέωση (refresh) της σελίδας μας κάθε 40 δευτερόλεπτα .

```
<meta http-equiv="refresh" content="30">
```

Μέσα σε μια ετικέτα μπορούμε να χρησιμοποιήσουμε την παράμετρο **style** έτσι ώστε να αλλάξουμε το χρώμα του κειμένου , να επιλέξουμε το μέγεθος ή τη γραμματοσειρά ενός κειμένου , να επιλέξουμε το χρώμα του φόντου και το είδος του κειμένου .

Παραδείγματα:

Αλλαγή χρώματος φόντου

```
<body style="background-color: blue;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
```

Αλλαγή χρώματος κειμένου

```
<h1 style="color:blue;">epikefalida</h1>
<p style="color:red;">paragrafos</p>
```

Αλλαγή μεγέθους κειμένου

```
<h1 style="font-size:300%;">epikefalida</h1>
<p style="font-size:160%;">paragrafos</p>
```

Αλλαγή χρώματος link (αλλαγή στο χρώμα ου link και του φόντου ανάλογα αν έχω επισκευθεί ή όχι το συγκεκριμένο link , αλλαγή του ποντικιού κ.λ.π.).

```
<style>
a:link    {color:green; background-color:transparent; text-decoration:none}
a:visited {color:pink; background-color:transparent; text-decoration:none}
a:hover   {color:red; background-color:transparent; text-decoration:underline}
a:active  {color:yellow; background-color:transparent; text-decoration:underline}
</style>
```

Χρώματα στην html

Είπαμε πριν πως μπορούμε να αλλάξουμε χρώμα σε ένα κείμενο , στο φόντο της ιστιοσελίδας κλπ. Ωστόσο τα χρώματα στην html μπορούν να λειτουργήσουν λίγο πιο περίπλοκα . Στις οθόνες των υπολογιστών τα χρώματα φτιάχνονται συνήθως με συνδυασμό των τριών βασικών χρωμάτων (κόκκινο , πράσινο , μπλε). Στην html μπορούμε να συνδυάσουμε αυτά τα τρία χρώματα για να επιλέξουμε το χρώμα που θέλουμε .

Όταν για παράδειγμα διαλέγουμε χρώμα για μια παράγραφο τότε μπορούμε να επιλέξουμε την ένταση των τριών βασικών μας χρωμάτων αν χρησιμοποιήσουμε την παράμετρο rgb(0,0,0). Όπως γίνεται κατανοητό το πρώτο μηδέν μας δίνει την ένταση του κόκκινου χρώματος (από τα αρχικά του red), το δεύτερο του πράσινου (green) και το τελευταίο του μπλε (blue). Η ένταση του κάθε χρώματος παίρνει τιμές από το 0 μέχρι το 255 με βάση το δεκαεξαδικό σύστημα .

```
<h2 style="background-color:rgb(255,0,0)">paragrafos</h2>
```
Κόκκινο χρώμα φόντου παραγράφου.
```
<h2 style="background-color:rgb(0,0,0);color:white">paragrafos</h2>
```
Μαύρο χρώμα φόντου παραγράφου με λευκά γράμματα.
Μπορούμε επίσης να χρησιμοποιήσουμε και το δεκαεξαδικό σύστημα , ως εξής:
```
<h2 style="background-color:#00FF00">paragrafos</h2>
```
Πράσινο χρώμα φόντου παραγράφου με βάση το δεκαεξαδικό σύστημα.

Λίστες

Ο πιο εύκολος τρόπος να φτιάξουμε λίστες στην html είναι χρησιμοποιώντας το στοιχείο <ul> και το στοιχείο <li> . Με την ετικέτα <ul> ξεκινάμε μια λίστα , η οποία τελειώνει με την ετικέτα . Η ετικέτα <li> και χρησιμοποιείται για την έναρξη και τη λήξη κάθε αντικειμένου που γράφεται στην λίστα .

Π.χ.
```
<ul>
  <li>gala</li>
  <li>zaxarh</li>
  <li>psomi</li>
</ul>
```

Μπορούμε επίσης να χρησιμοποιήσουμε την παράμετρο **style** για να αλλάξουμε την εμφάνιση των ΄΄σημαδιών΄΄ στα αντικείμενα μας (η default εμφάνιση είναι αυτή της κουκκίδας). Γράφοντας στο style **list-style-type:circle** κάνουμε τα σημάδια στην αρχή κάθε αντικειμένου κύκλους , γράφοντας **list-style-type:square** τα κάνουμε τετράγωνα , με το **list-style-type:none** τα αφαιρούμε τελείως και τέλος με το **list-style-type:disk** χρησιμοποιούμε την default επιλογή δηλαδή αυτή της κουκκίδας.

Π.χ.
```
<ul style="list-style-type:circle">
  <li>gala</li>
  <li>zaxarh</li>
  <li>psomi</li>
</ul>
```
Μπορούμε να φτιάξουμε αριθμημένες λίστες ή λίστες που αντί για ΄΄σημάδια΄΄ χρησιμοποιούν γράμματα με της ετικέτες <ol> και για έναρξη και λήξη αντίστοιχα. Τα αντικείμενα της λίστας θα προσδιορίζονται πάλι με το στοιχείο <li> .

π.χ.
```
<ol>
  <li>gala</li>
  <li>zaxarh</li>
  <li>psomi</li>
```

```
</ol>
```

(αριθμημένη λίστα)

Για να επιλέξουμε τον τύπο της αρίθμησης ή της γραμματοσειράς που θα χρησιμοποιεί η λίστα μας θα πρέπει να χρησιμοποιήσουμε την παράμετρο **type** . Γράφοντας **type="1"** αριθμούμε τη λίστα μας με τους αραβικούς αριθμούς , γράφοντας **type="A"** τα αντικείμενα της λίστας μας θα παρουσιάζονται με κεφαλαία γράμματα , ενώ γράφοντας **type="a"** με μικρά . Πληκτρολογώντας **type="I"** χρησιμοποιούμε λατινικούς αριθμούς (κεφαλαία) και **type="i"** πάλι λατινικούς αριθμούς (μικρά). Η default επιλογή εδώ είναι η αραβική αρίθμηση .

Π.χ.

```
<ol type="I">
  <li>gala</li>
  <li>zaxarh</li>
  <li>psomi</li>
</ol>
```

Χρησιμοποιώντας τα στοιχειά <dl>, <dt>, και <dd> έχουμε τη δυνατότητα να φτιάξουμε μια λίστα στην οποία να μπορούμε να περιγράψουμε κάθε αντικείμενο της . Η ετικέτα <dl> χρησιμοποιείται για την έναρξη της λίστας και η </dl> για τη λήξη της. Οι ετικέτες <dt> και </dt> χρησιμοποιούνται για να προσδιορίσουμε τα αντικείμενα της λίστας ενώ οι ετικέτες <dd> και </dd> χρησιμοποιούνται για την έναρξη και τη λήξη της περιγραφής μας.

Π.χ.

```
<dl>
  <dt>giaoyrti</dt>
  <dd>- apaxo, 0% lipara klp</dd>
  <dt>krema</dt>
  <dd>- gioths, topothethsh proiontos</dd>
</dl>
```

Μπορούμε επίσης να κάνουμε μια λίστα μέσα σε μια άλλη λίστα (listinception).

Π.χ.

```
<ul>
  <li>psomi</li>
  <li>laxanika
    <ul>
      <li>maroyli</li>
      <li>rapanaki</li>
    </ul>
  </li>
  <li>gala</li>
```

```
</ul>
```

Πίνακες

Χρησιμοποιώντας το στοιχείο <table> μπορούμε να κατασκευάσουμε πίνακες. Οι σειρές του πίνακα προσδιορίζονται με την ετικέτα <tr> για την έναρξη και </tr> για τη λήξη τους. Διαχωρίζουμε τον πίνακα σε στήλες με βάση τα δεδομένα τους, κάθε δεδομένο προσδιορίζεται με τις ετικέτες <td> και </td>. Τέλος οι επικεφαλίδες στο πίνακα κατασκευάζονται χρησιμοποιώντας το στοιχείο <th>.

Π.χ.

```
<table>
  <tr>
    <td>dimitris</td>
    <td>mandelias</td>
    <td>10</td>
  </tr>
  <tr>
    <td>kapoios</td>
    <td>allos</td>
    <td>5</td>
  </tr>
</table>
```

Μπορούμε να χρησιμοποιήσουμε την παράμετρο **border** και την **style** για να οριοθετήσουμε τον πίνακα. Ο αριθμός δίπλα από το border δηλώνει το πάχος της ¨γραμμής¨ οριοθέτησης (το width στο παρακάτω παράδειγμα κάνει μεγαλύτερα τα ¨τετράγωνα¨ του πίνακα ως προς το πλάτος).

Π.χ.

```
<table border="6" style="width:100%">
<tr>
    <td>dimitris</td>
    <td>mandelias</td>
    <td>10</td>
  </tr>
  <tr>
    <td>kapoios</td>
    <td>allos</td>
    <td>5</td>
  </tr>
</table>
```

Στο παραπάνω παράδειγμα είδαμε την παράμετρο border η οποία μας επιτρέπει να οριοθετήσουμε τον πίνακα μας, ωστόσο αν θέλουμε να οριοθετήσουμε τον πίνακα μας με μόνες γραμμές θα πρέπει να κάνουμε χρήση της παραμέτρου border-collapse καθώς η border έχει πάντα διπλή γραμμή οριοθέτησης. Η border-collapse χρησιμοποιείται μόνο με τη βοήθεια μιας άλλης γλώσσας προγραμματισμού, της css. Άλλες παράμετροι που λειτουργούν μόνο με τη βοήθεια της css είναι η παράμετρος id η οποία μας βοηθάει να προσδιορίσουμε με μοναδικό τρόπο ένα πίνακα, η border-spacing, η padding, η text-align κ.α.
Χρησιμοποιώντας τις παραμέτρους colspan και rowspan μπορούμε να κάνουμε ένα δεδομένο του πίνακα να εκτείνεται σε δυο ή περισσότερες στήλες ή γραμμές.

Π.χ.

```
<table>
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>shakira</td>
    <td>121 12 121</td>
    <td>123 12 123</td>
  </tr>
</table>
```

Μια επικεφαλίδα για ολόκληρο τον πίνακα (λεζάντα) προσδιορίζεται με τη βοήθεια του στοιχείου <caption>. Οι ετικέτες <caption> μπαίνουν συνήθως ακριβώς μετά από την ετικέτα έναρξης <table>.

Π.χ.

```
<table>
  <caption>bathmoi ptyxiakhs</caption>
  <tr>
    <th>onoma</th>
    <th>bathmos</th>
  </tr>
  <tr>
    <td>mandelias</td>
    <td>10 me tono</td>
  </tr>
  <tr>
    <td>allos</td>
    <td>oti na nai</td>
  </tr>
```

```
</table>
```

Επίσης μπορούμε να χρησιμοποιήσουμε τα στοιχεία **<colgroup>** και **<col>** τα οποία μας δίνουν τη δυνατότητα να επεξεργαστούμε μια ή περισσότερες στήλες του πίνακα (έτσι ώστε να μην χρειάζεται να επεξεργαζόμαστε κάθε δεδομένο ξεχωριστά).

Π.χ.

```
<table>
  <colgroup>
    <col span="2" style="background-color:red">
    <col style="background-color:yellow">
  </colgroup>
  <tr>
    <th>titlos</th>
    <th>titlos</th>
    <th>k allos titlos</th>
  </tr>
  <tr>
    <td>kati</td>
    <td>kati</td>
    <td>kai kati allo</td>
  </tr>
  <tr>
    <td>kati2</td>
    <td>kati2</td>
    <td>kai kati allo2</td>
  </tr>
</table>
```

Στο παραπάνω παράδειγμα οι δυο πρώτες στήλες έχουν κόκκινο χρώμα ενώ η τρίτη κίτρινο.

Τέλος υπάρχουν τα στοιχεία **<thead> , <tbody>** και **<tfoot>** που μας επιτρέπουν να ομαδοποιούμε τη σειρά των επικεφαλίδων, τις μεσαίες σειρές και την τελευταία σειρά αντίστοιχα.

Π.χ.

```
<table>
 <thead>
  <tr>
    <th>title</th>
    <th>title</th>
```

```
 </tr>
</thead>
<tfoot>
 <tr>
   <td>teleytaia seira</td>
   <td>teleytaia seira</td>
 </tr>
</tfoot>
<tbody>
 <tr>
   <td>mesh</td>
   <td>mesh</td>
 </tr>
 <tr>
   <td>mesh</td>
   <td>mesh</td>
 </tr>
</tbody>
</table>
```

Στο παραπάνω παράδειγμα έχουμε προσδιορίσει τη σειρά των τίτλων, τη μεσαία σειρά και την τελευταία σειρά, αν δεν είχαμε κάνει χρήση των <thead>, <tbody> και <tfoot> τότε οι μεσαίες και η τελευταία σειρά θα εμφανιζόντουσαν με αντίθετη σειρά στην ιστιοσελίδα. Για να αλλάξουμε το στυλ, το χρώμα κλπ στις σειρές θα έπρεπε να κάνουμε χρήση της css.

Κωδικοποίηση χαρακτήρων (set χαρακτηρων)

Για να απεικονιστεί σωστά μια ιστοσελίδα που έχουμε φτιάξει θα πρέπει να υπάρχει ένα set χαρακτήρων που να είναι γνωστό στον web browser. Μερικά τέτοια set χαρακτήρων είναι το ascII το οποίο ήταν το πρώτο που μπορούσε να χρησιμοποιηθεί στο internet και υποστήριζε αριθμούς από το 0 μέχρι το 9, το αγγλικό αλφάβητο και κάποια σύμβολα, το ansi, το iso-8859-1 και το utf-8 το οποίο καλύπτει σχεδόν όλους τους χαρακτήρες και τα σύμβολα στον κόσμο. Για να επιλέξουμε set χαρακτήρων στην html4 πρέπει να χρησιμοποιήσουμε την ετικέτα <meta> ως εξής:

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

Στην html5 είναι αρκετά πιο απλοποιημένο και χρειαζόμαστε απλά τη βοήθεια της παραμέτρου charset, ο κώδικας μας θα δείχνει κάπως έτσι:

```
<meta charset="UTF-8">
```

Στην html όμως υπάρχουν κάποιοι χαρακτήρες οι οποίοι είναι δεσμευμένοι όπως άλλωστε και σε κάθε γλώσσα προγραμματισμού. Οι δεσμευμένοι χαρακτήρες είναι χαρακτήρες που δεν μπορούν να χρησιμοποιηθούν συνήθως επειδή χρησιμοποιούνται από το πρόγραμμα όπως για παράδειγμα οι χαρακτήρες <, > οι

οποίοι αν χρησιμοποιηθούν στην html μπορεί να μπερδευτούν από το browser με ετικέτες του προγράμματος. Για να μπορούμε να χρησιμοποιήσουμε τέτοιους χαρακτήρες ή και χαρακτήρες που δεν υπάρχουν στο πληκτρολόγιο μας, πρέπει να κάνουμε χρήση των entities της html. Μπορούμε να καλέσουμε μια entity είτε γράφοντας το όνομα της, είτε γράφοντας τον αριθμό της.

Παραδείγματα entities

ΧΑΡΑΚΤΗΡΑΣ	**ΟΝΟΜΑ ENTITY**	**ΑΡΙΘΜΟΣ ENTITY**
<	<	<
>	>	>
&	&	&
€	€	€

Π.χ.

```
<p>50 ekatomuria  &euro;</p>
<p>50 ekatomuria  &#8364;</p>
```

Μορφοποίηση κειμένου

Μπορούμε να μορφοποιήσουμε το κείμενο της ιστιοσελίδας μας ανάλογα με τις ανάγκες μας, αν θέλουμε δηλαδή έντονη γραφή, πλάγια γραφή, υπογράμμιση κ.λ.π. Αυτό το κάνουμε με τη βοήθεια των παρακάτω στοιχείων:

<b> για κείμενο με έντονη γραφή.
<i> για κείμενο με πλάγια γραφή.
<mark> για μαρκαρισμένο κείμενο (με έμφαση).
<del> για διαγραφόμενο κείμενο.
<ins> για υπογραμμισμένο κείμενο.

Π.χ.

```
<p><b>keimeno</b>.</p>
<h2>titlos<mark>markarismenos titlos</mark> titlos</h2>
```

Σχόλια (μέσα στην html)

Μπορούμε επίσης να χρησιμοποιήσουμε σχόλια στην html τα οποία δεν είναι εμφανή στην ιστιοσελίδα μας αλλά μπορεί να μας είναι χρήσιμα (σημειώσεις μας ή κάποια υπενθύμιση). Τα σχόλιά μας ξεκινάνε με την ετικέτα **<!- -** και λήγουν με την **- ->**.

Π.χ.

```
<!—sxolioooooooooooo!!!!!!!!!!!!! -->
```

Σταθερές απεικόνισης στοιχείων html

Κάθε στοιχείο στην html έχει το δικό του τρόπο απεικόνισης στην ιστοσελίδα, υπάρχουν δυο κύριοι τρόποι απεικόνισης, η απεικόνιση τύπου block και η απεικόνιση τύπου inline. Τα block στοιχεία ξεκινάνε πάντα από την αρχή της

γραμμής και καταλαμβάνουν όλο διαθέσιμο σε αυτά χώρο, ενώ τα στοιχεία inline δεν χρειάζεται να ξεκινάνε από την αρχή της γραμμής και καταλαμβάνουν μόνο όσο χώρο χρειάζονται. Ένα παράδειγμα κλασσικού στοιχείου τύπου block είναι αυτό της παραγράφου, το στοιχείο δηλαδή <p>, ενώ ένα παράδειγμα inline τύπου στοιχείο είναι αυτό της εικόνας, το στοιχείο <img>.

Μπορούμε να χρησιμοποιήσουμε το στοιχείο ** ** για να μετατρέψουμε οποιοδήποτε στοιχείο θέλουμε σε στοιχείο τύπου block έτσι ώστε να καταλαμβάνει όλο το διαθέσιμο σε αυτό χώρο, και χρησιμοποιούμε το στοιχείο **** για να μετατρέψουμε όποιο στοιχείο θέλουμε σε στοιχείο τύπου inline.

Π.χ.

```
<div style="background-color:black; color:white; padding:20px;">
<h2>τίτλος-επικεφαλίδα</h2>
<p>Σε αυτό το παράδειγμα έχουμε χρησιμοποιήσει το στοιχείο div για να κάνουμε το φόντο μαύρο και τα γράμματα του κειμένου λευκά. Ολόκληρο το κείμενο και η επικεφαλίδα είναι ένα block.</p>
</div>

<h1>Μόνο <span style="color:red">αυτό</span> το κομμάτι του κειμένου είναι χρωματισμένο κόκκινο, χάρη στο στοιχείο span το αυτό είναι πλέον inline.</h1>
```

Μπορούμε στην html να κατηγοριοποιήσουμε-ταξινομήσουμε τα block και τα inline στοιχεία μας ως εξής:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.kathg {
   background-color:black;
   color:white;
}
</style>
</head>
<body>
<div class="kathg">
<p>Με αυτόν τον τρόπο κάθε φορά που θα γράφω δίπλα από το class το όνομα kathg δηλαδή κατηγοριοποίηση θα καλώ το συγκεκριμένο div που έχω φτιάξει πιο πάνω.</p>
</div>
<div class="kathg">
<p>Το ίδιο θα μπορούσα να κάνω και με το inline, δηλαδή θα χρησιμοποιούσα το
```

στοιχείο style, μέσα σε αυτό θα έγραφα span.onoma, θα έγραφα πως θέλω να εμφανίζεται, θα τελείωνα το style και μετά όποτε το χρειαζόμουν θα το καλούσα με span class=onoma.</p>

</body>
</html>

Iframes

Κάποιες φορές βλέπουμε να απεικονίζονται κάποιες ιστοσελίδες μέσα σε άλλες ιστοσελίδες. Μπορούμε να κάνουμε κάτι τέτοιο στην γλώσσα προγραμματισμού html με τη χρήση του προγραμματιστικού στοιχείου **<iframe>**. Όπως και με τις εικόνες μπορούμε να χρησιμοποιήσουμε τις παραμέτρους width και height για να επηρεάσουμε της διαστάσεις της εμφολιασμένης μας ιστιοσελίδας. Επίσης χρησιμοποιώντας την παράμετρο style και τη γλώσσα προγραμματισμού css μπορούμε να επιλέξουμε τον τύπο της συνοριοποίησης της ιστιοσελίδας μας ή και να την αφαιρέσουμε τελείως.

Π.χ.

```
<iframe src="url" width="200" height="200"></iframe>
```

Τέλος μπορούμε με τη χρήση των παραμέτρων **name** και **target** να βάλουμε ένα link στην ιστιοσελίδα μας το οποίο θα ανοίγει μια καινούρια ιστοσελίδα μέσα στο iframe μας. Για να το κάνουμε αυτό θα πρέπει να δώσουμε στο target το όνομα που έχουμε δώσει στο name.

Π.χ.

```
<iframe src="url" name="onoma"></iframe>
<p><a href="http//klp.com" target="onoma">link</a></p>
```

3.4 Η γλωσσα προγραμματισμου PHP

3.4.1 Εισαγωγή

Η php (hypertext preprocessor) είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία ιστοσελίδων και για διάφορες άλλες διαδικτυακές εφαρμογές. Είναι μια γλώσσα server side, δηλαδή την τρέχει ο

διακομιστής(server) έτσι ώστε να δούμε εμείς το αποτέλεσμα στην ιστοσελίδα μας. Μπορούμε να ενσωματώσουμε κώδικα php μέσα στον κώδικα μιας html ιστοσελίδας, με αυτόν τον τρόπο ο php κώδικας θα εκτελείται κάθε φορά που κάποιος επισκέπτεται την συγκεκριμένη ιστοσελίδα.

3.4.2 Στοιχεία κώδικα της php

Όπως είπαμε και στην εισαγωγή, ο κώδικας της php εκτελείται στον server έτσι ώστε το αποτέλεσμα να σταλεί στο πρόγραμμα περιήγησης(browser). Μπορούμε να εισάγουμε έναν κώδικα php οπουδήποτε μέσα σε έναν κώδικα http με τους εξής τρόπους:

- Ξεκινώντας με την ετικέτα **<?** και καταλήγοντας στην ετικέτα **?>**.
- Ξεκινώντας με την ετικέτα **<?php** και τελειώνοντας με την ετικέτα **?>.**
- Με την ετικέτα αρχής **<script language=php>** και την ετικέτα λήξης **</script>.**

Από όλους τους παραπάνω τρόπους ο δεύτερος είναι ο πιο διαδεδομένος και ο πιο σίγουρος. Σχόλια στην php μπορούμε να κάνουμε με πολλούς διαφορετικούς τρόπους.

Π.χ.

```
<!DOCTYPE html>
<html>
<body>

<?php
// σχολιο μιας μονο γραμμης
# κι άλλο σχολιο μονο μιας γραμμης
/*
σχολιο πολλων
γραμμωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωωω
ων
*/
?>
</body>
</html>
```

Μεταβλητές php

Οι μεταβλητές(variables) είναι στην ουσία οι χώροι αποθήκευσης που χρησιμοποιούμε για να αποθηκεύουμε τις πληροφορίες μας. Οι μεταβλητές γράφονται ως $ ακολουθούμενες από το όνομα που θα τους δώσουμε, για παράδειγμα:

```
<?php
$kmn = "kati egraca";
$x = 15;
$y = 30;
?>
```

Στο παραπάνω παράδειγμα έχουμε αποθηκεύσει τον αριθμό 15 στη μεταβλητή $x, τον αριθμό 30 στη μεταβλητή $y και ένα κείμενο στη μεταβλητή &kmn.

Όταν ονομάζουμε μεταβλητές θα πρέπει να ακολουθούνται οι εξής κανόνες:

1. Το όνομα της μεταβλητής θα πρέπει να ξεκινάει με γράμμα και ποτέ με αριθμό.
2. Δεν μπορούν να περιέχονται κενά στο όνομα μιας μεταβλητής.
3. Έχει σημασία αν θα χρησιμοποιήσουμε μικρά ή κεφάλαια γράμματα, $a και $A είναι δυο διαφορετικές μεταβλητές.

Οι μεταβλητές αυτές δεν είναι ορατές σε μας εκτός αν της κάνουμε εμείς ορατές, αυτό το επιτυγχάνουμε με τις παραμέτρους **echo** και **print**. Οι δυο αυτή παράμετροι κάνουν στην ουσία την ίδια δουλειά με πολύ μικρές διαφορές. Η echo για παράδειγμα είναι ελάχιστα πιο γρήγορη από την print.

Π.χ.

```
<?php
$kmn = "κείμενο PHP";
$x = 30;
$y = 15;

echo "<h2>$kmn</h2>";
echo "allo keimeno kai $kmn <br>";
echo $x + $y;
?>
```

Στο παραπάνω παράδειγμα αφού θέσαμε κάποιες μεταβλητές, εμφανίσαμε με την παράμετρο echo στην πρώτη γραμμή το κείμενο της μεταβλητής $kmn σαν επικεφαλίδα, η επόμενη γραμμή θα εμφανίσει ένα άλλο κείμενο στο οποίο μέσα θα είναι και το κείμενο της μεταβλητής $kmn και στην τελευταία γραμμή έχουμε το άθροισμα των μεταβλητών $x και $y.Παρατηρουμε ότι μπορούμε να χρησιμοποιήσουμε ταυτόχρονα και στοιχεία της html όπως αυτό της επικεφαλίδας και το break.

Αντί για τις μεταβλητές μπορούμε να χρησιμοποιήσουμε τα constants ως αποθηκευτικούς χώρους στο πρόγραμμά μας, η διαφορά με τις μεταβλητές είναι ότι τα constants είναι σταθερά και δεν αλλάζουν. Για να φτιάξουμε ένα constant θα πρέπει να χρησιμοποιήσουμε το define function(περισσότερα για τα functions παρακάτω).

Τύποι δεδομένων

Οι τύποι δεδομένων που μπορούμε να αποθηκεύσουμε μέσα σε μια μεταβλητή, και οι οποίοι έχουν διαφορετική χρήση και διαφορετικές ιδιότητες μέσα στο πρόγραμμά μας, είναι οι παρακάτω:

1. Τα Strings(αλφαριθμητικά) είναι οποιαδήποτε σειρά χαρακτήρων(κείμενα κλπ.), περικλείεται από εισαγωγικά.

   Π.χ.

   ```
   <?php
   $x = "βάλε δέκα";
   $y = "10";
   ?>
   ```

2. Οι integer αριθμοί, δηλαδή οι ακέραιοι αριθμοί σε δεκαδικό, δεκαεξαδικό ή οκταδικό σύστημα.

3. Οι αριθμοί κινητής υποδιαστολής(float point number), δηλαδή οι δεκαδικοί αριθμοί με υποδιαστολή.

4. Τα Boolean(λογικός τύπος δεδομένων) δεδομένα, τα δεδομένα που μπορεί να έχουν μόνο δυο καταστάσεις(true ή false).

5. Οι πίνακες(array).Τα array χρησιμοποιούνται για να αποθηκεύσουμε περισσότερες από μια τιμές σε μια μόνο μεταβλητή.

   ```
   <?php
   $foit = array("allos", "mandelias", "ki allos");
   echo "o ". $foit[1] . " einai o kaluteros foiththns ever.";
   ?>
   ```

   Μπορούμε επίσης να βάλουμε arrays μέσα σε arrays έτσι ώστε να έχουμε πίνακες με πολλές σειρές και στήλες.

6. Το null, ένα είδος δεδομένου που χρησιμοποιούμε κυρίως για να αδιάσουμε τις μεταβλητές μας.

Τελεστές(operators)

1. **Αριθμητικοί τελεστές**, οι τελεστές που χρησιμοποιούνται για να κάνουμε αριθμητικές πράξεις. Οι αριθμητικοί τελεστές είναι οι εξής:

+	πρόσθεση
-	Αφαίρεση
*	πολλαπλασιασμός
/	διαίρεση
%	ακέραιο υπόλοιπο
**	ύψωση σε δύναμη

π.χ.

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 10;
$y = 6;
echo $x % $y;
?>
</body>
</html>
```

Το αποτέλεσμα στην παραπάνω πράξη θα είναι 4.

2. **Τελεστές εκχώρησης.** Ο βασικός τελεστής εκχώρησης είναι το =, το οποίο μπορεί να δώσει μια τιμή σε μια μεταβλητή.

π.χ.

```
<?php
$x = 10;
?>
```

Το $x θα πάρει την τιμή 10.

Άλλοι τελεστές εκχώρησης είναι οι εξής: +=, -=, *=, /= και %=.

π.χ.

```
<?php
$x = 5;
$x += 10;
echo $x;
?>
```

Στο παραπάνω παράδειγμα το echo θα εμφανίσει τον αριθμό 15.

3. **Τελεστές σύγκρισης.** Τελεστές που χρησιμοποιούνται για να συγκρίνουν δυο τιμές ή μεταβλητές και να μας φέρουν αποτέλεσμα true ή false.

$a==$b	true αν το $a είναι ίσο με το $b.
$a===$b	true αν το $a είναι ίσο και ίδιου τύπου με το $b.
$a<$b	true αν το $a είναι μικρότερο του $b.
$a>$b	true αν το $a είναι μεγαλύτερο του $b.
$a<=$b	true αν το $a είναι μικρότερο ή ίσο του $b.
$a>=$b	true αν το $a είναι μεγαλύτερο ή ίσο του $b.
$a <>$b	true αν το $a δεν είναι ίσο με το $b.

4. **Τελεστές αύξησης/μείωσης.**

++$a	Αυξάνει το $a κατά ένα και μετά το επιστρέφει.
$a++	Επιστρέφει το $a και μετά το αυξάνει κατά ένα.
--$a	Μειώνει το $a κατά ένα και μετά το επιστρέφει.
$a--	Επιστρέφει το $a και μετά το αυξάνει κατά ένα.

5. **Λογικοί τελεστές.** Χρησιμοποιούνται για να κάνουμε λογικές πράξεις.

Λογικός τελεστής	λογική πράξη
and ή &&	and
or	or
xor	xor
!	not

Εντολή if…elseif…else

Κάποιες φορές όταν γράφουμε έναν κώδικα μπορεί να θέλουμε να τελέσει

διαφορετικές λειτουργίες ανάλογα με το αν αληθεύουν κάποιες συνθήκες. Για αυτό το λόγο υπάρχει στον προγραμματισμό η εντολή if.Στην html η εντολή if συντάσσεται ως εξής:

if (συνθήκη) {
κώδικας που εκτελείται αν η παραπάνω συνθήκη είναι αληθής;
} elseif (συνθήκη) {
κώδικας που εκτελείται αν η παραπάνω συνθήκη είναι αληθής;
} else {
κώδικας που εκτελείται αν όλες οι παραπάνω συνθήκες είναι ψευδείς;
}

επαναλήψεις(loop)

while loop

Η while loop εκτελεί τον κώδικα που της ανατέθηκε όσο η συνθήκη της είναι αληθής. Στο παρακάτω παράδειγμα η while loop ελέγχει αν η μεταβλητή που της δώσαμε είναι μικρότερη ή ίση με το 10, όσο η συνθήκη αυτή ισχύει ο κώδικας επαναλαμβάνεται και η μεταβλητή αυξάνεται κατά 1.

π.χ.

```
<?php
$x = 1;
while($x <= 10) {
   $x++;
}
?>
```

Do while loop

Η do while loop είναι στην ουσία σαν την while loop με τη διαφορά ότι πάντα θα εκτελεί τον κώδικα της μια φορά παραπάνω. Επίσης η do while loop θα εκτελεί τον κώδικα της πάντα τουλάχιστον μια φορά.

π.χ.

```
<?php
$x = 1;
do {
   echo "The number is: $x <br>";
   $x++;
} while ($x <= 10);
?>
```

For loop

Η for loop μας είναι χρήσιμη όταν ξέρουμε από πριν πόσες επαναλήψεις θέλουμε να κάνουμε. Στην for loop θα πρέπει να θέσουμε μια αρχική τιμή, μια συνθήκη και το loop counter.Στο παρακάτω παράδειγμα η αρχική τιμή είναι 0, η συνθήκη είναι ο αριθμός να είναι μικρότερος ή ίσος του 10 και το loop counter είναι 1.

Π.χ.

```
<?php
for ($x = 0; $x <= 10; $x++) {
   echo "γράφω κώδικα";
}
?>
```

Functions

Τα functions είναι στην php ότι γνωρίζουμε στον προγραμματισμό γενικότερα ως υποπρογράμματα, είναι δηλαδή κάποια κομμάτια κώδικα χρήσιμα για συγκεκριμένες εργασίες μέσα στο πρόγραμμά μας, τα οποία μπορούμε να καλέσουμε όσες φορές θέλουμε και οπότε θέλουμε. Αυτό είναι όπως γίνεται κατανοητό πολύ χρήσιμο για κάθε προγραμματιστή καθώς μπορεί έτσι να κερδίσει πολύ χρόνο. Η php λοιπόν έχει πολλά προκατασκευασμένα τέτοια υποπρογράμματα έτσι ώστε ο εκάστοτε προγραμματιστής να τα καλεί όταν τα χρειάζεται, μπορούμε φυσικά να φτιάξουμε και τα δικά μας υποπρογράμματα τα οποία θα ταιριάζουν απόλυτα στις ανάγκες μας. Ο κώδικας που εκτελείται μέσα σε ένα function θα πρέπει να είναι κλεισμένος μέσα σε αγκύλες ({, }), ενώ μπορούμε να καλέσουμε ένα function απλά γράφοντας το όνομά του. Τέλος θα πρέπει να πούμε ότι οι μεταβλητές που χρησιμοποιούνται μέσα σε ένα function δεν ισχύουν σε ολόκληρο το πρόγραμμα και το αντίστροφο. Για να αποκτήσουμε πρόσβαση σε μεταβλητές του προγράμματος μέσα από ένα υποπρόγραμμα θα πρέπει να χρησιμοποιήσουμε την παράμετρο global.

Παραδείγματα functions:

```
<?php
function katastash($on, $bath) {
   echo "$on βαθμός πτυχίου $bath <br>";
}
katastash ("ΜΑΝΔΕΛΙΑΣ", "10");
katastash ("άλλος", "7");
katastash ("άλλος", "5");
?>
```

Στο παραπάνω παράδειγμα κάθε φορά που θα καλούμε το υποπρόγραμμα θα πρέπει να γράφουμε και τις δυο παραμέτρους έτσι ώστε να μας εμφανίζει το όνομα του φοιτητή και το βαθμό του πτυχίου του.

```
<?php
function sum($x, $y) {
   $z = $x + $y;
    echo $z;
}
sum(5, 10)
sum(7, 13)
sum(2, 4);
?>
```

Σε αυτό το παράδειγμα χρησιμοποιούμε ένα απλό function για να κάνουμε την πρόσθεση δυο αριθμών και να εμφανίσουμε το αποτέλεσμα.

Συμπεράσματα

Όπως αναφέρθηκε σε προηγούμενα κεφάλαια η συγκεκριμένη πτυχιακή εργασία έχει σκοπό την απόκτηση γνώσεων και εμπειρίας πάνω στη κατασκευή ηλεκτρικών και υπολογιστικών κυκλωμάτων αλλά και τη συγκομιδή πληροφοριών πάνω στα γνωστικά πεδία που χρησιμοποιήθηκαν. Με την ελπίδα λοιπόν να επιτεύχθηκαν οι παραπάνω στόχοι άλλα και να χρησιμοποιηθεί το γνωστικό υλικό και η κατασκευή του κυκλώματος ως ερεθίσματα για την περαιτέρω ενασχόληση ακόμα περισσότερων ανθρώπων στον κλάδο της ηλεκτρολογίας και όχι μόνο με υπολογιστικά συστήματα θα πρέπει να κάνουμε μερικές τελευταίες παρατηρήσεις.

Το πρακτικό κομμάτι της πτυχιακής εργασίας λοιπόν, όπως γίνεται κατανοητό δεν έχει πρακτική χρήση στη συγκεκριμένη μορφή αλλά αποτελεί και χρησιμεύει ως παράδειγμα της χρήσης υπολογιστικών συστημάτων στην ηλεκτρολογία, έτσι για να δείξουμε καλύτερα τις δυνατότητες της κατασκευής μας η ηλεκτρική συσκευή που επιλέχτηκε ήταν μια ηλεκτρική μηχανή. Είπαμε επίσης σε άλλα κεφάλαια πως η συγκεκριμένη κατασκευή μπορεί να χρησιμοποιηθεί για τον έλεγχο οποιασδήποτε ηλεκτρικής συσκευής, ωστόσο ανάλογα με τη συσκευή θα πρέπει να αλλάξει τόσο το κύκλωμα όσο ο προγραμματισμός της κατασκευής. Τέλος θα πρέπει να αναφερθούμε στο γεγονός πως η επιλογή των περισσότερων υλικών και συσκευών που επιλέχτηκαν για την κατασκευή του κυκλώματος έγινε με κριτήριο κυρίως το οικονομικό κόστος.

Βιβλιογραφία

- Αθανάσιος Σπυριδάκος, «Αντικειμενοστραφεις προγραμματισμος σε περιβαλλον visual basic.Net», Συγχρονη εκδοτικη
- M.Morris Mano, Michael D.Ciletti, «Ψηφιακη σχεδιαση (Τεταρτη εκδοση)», Εκδοσεις Παπασωτηρίου
- Stephen J.Chapman, «Ηλεκτρικες μηχανες (Τεταρτη εκδοση)», Εκδοσεις Τζιόλα
- Ι.Α.Τεγοπουλος, «Ηλεκτρικες μηχανες», Εκδοσεις συμμετρια
- Δρ.Σταμάτης Αλατσαθιανός, « Μικροελεγκτες 8051»
- Elizabeth Castro, «Εισαγωγη στην HTML για τον παγκοσμιο ιστο», Εκδοσεις Κλειδάριθμος
- Shuyler, «Εφαρμοσμένα Ηλεκτρονικά», Εκδόσεις Τζιόλα
- Στέφανος Μανιάς, «Ηλεκτρονικά Ισχύος», Εκδοσεις Συμεών
- https://dspace.lib.uom.gr
- http://www.gr.circuitlib.com
- https://el.wikipedia.org
- http://www.epanorama.net
- http://www.w3schools.com
- https://portforward.com
- http://setuprouter.com
- http://www.electroniccircuits.gr