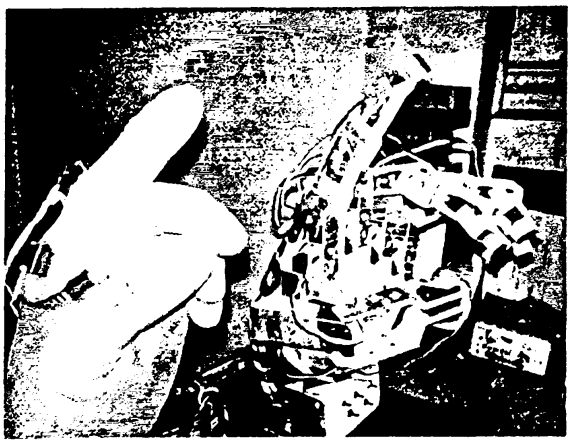
Ανάπτυξη Ρομποτικής Άκρας Χειρός με χρήση μικροελεγκτών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωστούλας Ελευθέριος

Τεχνολόγος Μηχανικός Αυτοματισμών



Επίβλεψη: Ιωάννης Ευαγγέλου, Αναπληρωτής Καθηγητής

Εργαστήριο Φυσικής Υψηλών Ενεργειών

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΣΤΙΣ ΣΥΓΧΡΟΝΕΣ ΗΛΕΚΤΡΟΝΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ

ΙΩΑΝΝΙΝΑ, ΙΟΥΝΙΟΣ 2014

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μίας ανθρωπόμορφης πλήρως - οδηγούμενης ρομποτικής άκρας χείρας. Η ρομποτική χείρα θα μπορούσε να φανεί πολύ χρήσιμη σε χώρους με ραδιενέργεια, εργαστήρια χημείας, χώρους με υψηλή θερμοκρασία και γενικότερα σε περιβάλλοντα επικίνδυνα και απαγορευτικά για τον άνθρωπο. Η ρομποτική χείρα που κατασκευάστηκε έχει δεκαεννέα βαθμούς ελευθερίας και μιμείται τις κινήσεις του ανθρώπινου χεριού μέσω διάταξης αισθητήρων κάμψης. Η διάταξη αισθητήρων προσαρμόζεται πάνω σε ανθρώπινο χέρι και συνδέεται από απόσταση μέσω καλωδίων με το σύστημα ελέγχου της ρομποτικής χείρας. Κάθε αισθητήρας κάμψης ελέγχει και μία άρθρωση - κινητήρα της ρομποτικής χείρας. Για την υλοποίηση του συστήματος ελέγχου χρησιμοποιήθηκαν δύο μικροελεγκτές ATmega2560 8-bit. Η ρομποτική χείρα που κατασκευάστηκε είναι σε θέση να κινεί τα δάχτυλα της, να κρατάει αντικείμενα και γενικότερα να εκτελεί λειτουργίες του ανθρώπινου χεριού ανάλογα με τις κινήσεις των δαχτύλων του χεριού του χρήστη με καλή ακρίβεια. Με την παρούσα διάταξη, ο χρήστης μπορεί να ελέγχει την ρομποτική χείρα από απόσταση δύο μέτρων. Η απόσταση αυτή όμως μπορεί να επεκταθεί ανάλογα με τις ανάγκες της εφαρμογής.

Abstract

The objective of the present thesis is the development of an anthropomorphic, fully-actuated robotic hand. This robotic hand can be used into radioactive places, chemistry labs, high temperature places and generally in environments, dangerous for humans. The robotic hand consists of eighteen degrees of freedom and imitates the movements of the human hand with the use of multiple bend sensors. These sensors are adapted on the human hand using a glove which is connected from a distance, through wires, to the control system of the robotic hand. Each bend sensor controls one joint-motor of the robotic hand. Two 8-bit ATmega2560 microcontrollers were used for the development of the robotic hand. The robotic hand built, is capable of moving its fingers, holding several things and generally to execute the functions of the human hand according to the movement of the user's fingers accurately. With the pattern presented, the user is able to control the robotic hand from a distance of two meters. This distance can be expanded according to the needs of the application.

Ευχαριστίες

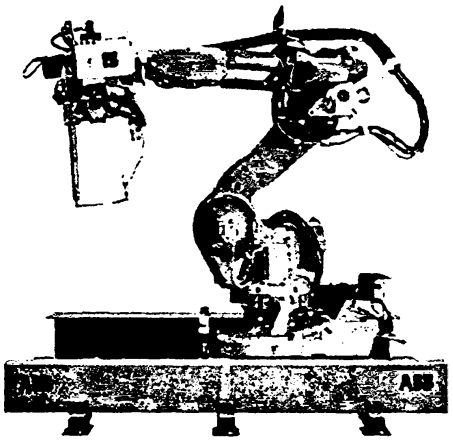
Σε αυτό το σημείο θα ήθελα να ευχαριστήσω τον άνθρωπο, που με τις συμβουλές και τις παρατηρήσεις του, συνέβαλε στην πραγματοποίηση αυτής της εργασίας.

Τον επιβλέποντα της διπλωματικής μου εργασίας Αναπληρωτή Καθηγητή κ. Ι. Ευαγγέλου, Μέλος του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων, για την αμέριστη βοήθεια και καθοδήγηση καθ'όλη τη διάρκεια εκπόνησης της εργασίας μου.

Περιεχόμενα

Εισαγωγή

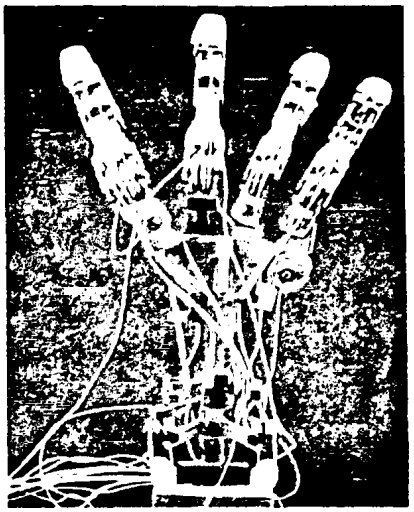
Με την ραγδαία ανάπτυξη των υψηλών τεχνολογιών όπως η μικρό-ηλεκτρονική και η μικρό-μηχανική ακριβείας, οι ρομποτικές διατάξεις δεν περιορίζονται πλέον μόνο σε βιομηχανικές εφαρμογές και περιβάλλοντα. Έχουν όμως εξαπλωθεί και σε πολλούς κλάδους των επιστημών, σε στρατιωτικές εφαρμογές, στην έρευνα, τη διάσωση, αλλά και στην καθημερινότητα όπου η ζωή των ανθρώπων γίνεται ευκολότερη και ασφαλέστερη βοηθώντας στην πραγματοποίηση συγκεκριμένων αναγκών. Ο κλάδος ο οποίος ασχολείται με την δημιουργία τέτοιων εφαρμογών ονομάζεται Ρομποτική. Η ρομποτική συνδέει τους κλάδους των ηλεκτρονικών και της μηχανικής με απώτερο σκοπό την μίμηση των λειτουργιών των διαφόρων μηχανισμών της φύσης και των έμβιων όντων, όπως ο άνθρωπος, αλλά και οπουδήποτε αυτό είναι επιθυμητό και αναγκαίο. Σε πολλές από αυτές τις εφαρμογές τα ρομπότ σχεδιάζονται ώστε να εκτελέσουν μία συγκεκριμένη λειτουργία. Ωστόσο, καθώς η χρήση των ρομπότ αυξάνεται συνεχώς, το ίδιο αυξάνεται και η ανάγκη για την αλληλεπίδραση τους με διάφορα αντικείμενα στο περιβάλλον τους. Ο σχεδιασμός ρομποτικών άκρων τα οποία μπορούν να σηκώσουν και να κρατήσουν διάφορα αντικείμενα με σκοπό να τα χρησιμοποιήσουν ως εργαλεία για κάποια εργασία είναι μία σημαντική πρόκληση στην ανάπτυξη ρομπότ αυτού του είδους. Ρομποτικά άκρα απλών λειτουργιών όπως η κλασσική αρπάγη (εικόνα 1) στην οποία χρησιμοποιούνται δύο παράλληλοι σιαγώνες, οδηγοί εργαλείων και ειδικές λαβές είναι κοινό φαινόμενο σε ρομποτικά συστήματα στις μέρες μας. Αυτά τα ρομποτικά άκρα έχουν το πλεονέκτημα της απλότητας στο σχεδιασμό και την κίνηση δίνοντας έτσι αυξημένη εμπιστοσύνη στη λειτουργία τους και το μειωμένο κόστος συντήρησης. Ωστόσο, για να επιτευχθούν πολλαπλές λειτουργίες ένα απλό ρομπότ θα χρειαζόταν ή πολλαπλούς ρομποτικούς βραχίονες με ένα διαφορετικό άκρο στον καθένα ή την ικανότητα αλλαγής των άκρων ανάλογα με την εκάστοτε εργασία.



Εικόνα 1: Βιομηχανικός βραχίονας με αρπάγη

Γι αυτό το λόγο καταστάσεις οι οποίες απαιτούν μεγαλύτερη ευελιξία προϋποθέτουν και τη χρήση μιας γενικότερης μεθόδου αλληλεπίδρασης με το περιβάλλον.

Πολλές εταιρίες ρομποτικής καθώς και κάποια εκπαιδευτικά ιδρύματα αλλά και ιδιώτες έχουν κατασκευάσει τα τελευταία χρόνια διάφορες ρομποτικές διατάξεις οι οποίες προσομοιώνουν με διάφορες μεθόδους και προσεγγίσεις το ανθρώπινο χέρι (εικόνα 2), καθώς αυτό είναι σε θέση να εκτελεί πλήθος διαφορετικών κινήσεων προσαρμόζοντας την λαβή του ανάλογα με το αντικείμενο που πρέπει να κρατήσει κάθε φορά. Έτσι ήταν πρόκληση να κατασκευαστεί ένα ρομποτικό χέρι με τα ίδια λειτουργικά χαρακτηριστικά, όπως για παράδειγμα η πλάγια κίνηση μεταξύ των ρομποτικών δαχτύλων καθώς και η ανεξάρτητη κίνηση μεταξύ των αρθρώσεων του κάθε ρομποτικού δαχτύλου, αλλά με πολύ λιγότερα χρήματα από αυτά που έχουν δαπανηθεί για άλλες τέτοιες κατασκευές έως τώρα.

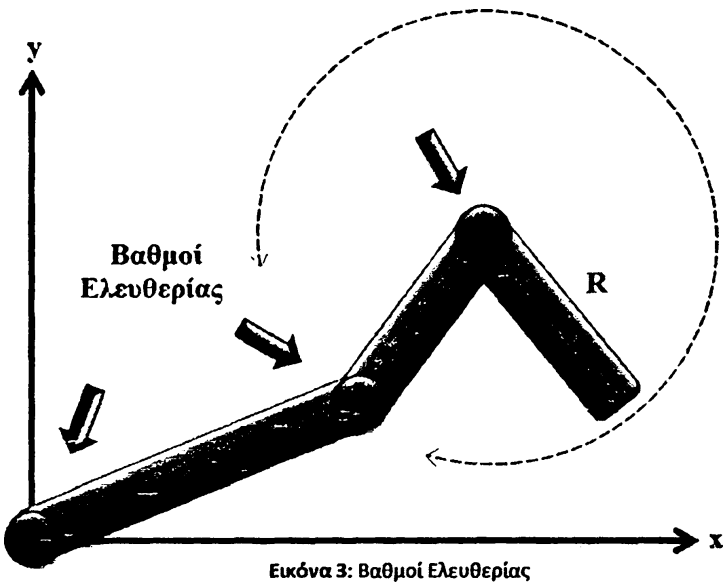

Εικόνα 2: Υπο-οδηγούμενή ρομποτική χειρα

Στην παρούσα διπλωματική εργασία υλοποιήθηκε ένα ρομποτικό χέρι με πλήρη ανεξαρτησία κινήσεων των αρθρώσεων του κάθε δαχτύλου το οποίο αντιγράφει τις κινήσεις ενός πραγματικού ανθρώπινου χεριού μέσω μίας διάταξης αισθητήρων από απομακρυσμένη θέση. Η παρουσία του θα μπορούσε να φανεί πολύ χρήσιμη σε χώρους με υψηλά επίπεδα ραδιενέργειας, εργαστήρια χημείας, χώρους με υψηλή θερμοκρασία και γενικότερα σε περιβάλλοντα επικίνδυνα και απαγορευτικά για τον άνθρωπο. Επίσης η ρομποτική χείρα θα μπορούσε να προσαρμοστεί σε μία μεγαλύτερη ρομποτική διάταξη όπως ένα ρομποτικό βραχίονα και να έχει μεγαλύτερη ευελιξία. Ενδιαφέρον παρουσιάζει ο ρεαλισμός που προσδίδει η μέθοδος με την οποία κατασκευάσθηκε η ρομποτική χείρα σε συνδυασμό με τα υλικά σχετικά χαμηλού κόστους από τα οποία αποτελείται, συγκρινόμενα πάντα με το κόστος άλλων τέτοιων συστημάτων. Επιπλέον, η κατασκευή ξεκίνησε από το μηδέν και συμπεριελάμβανε το σχεδιασμό και κατασκευή πλακετών PCB καθώς και την ανάπτυξη λογισμικού σε γλώσσα embedded C. Επίσης έγινε η μελέτη και κατασκευή

μεταλλικών μικρό-μηχανικών διατάξεων για τη στήριξη και λειτουργία της ρομποτικής χείρας και την ολοκλήρωση του συστήματος.

Γενικά, υπάρχουν δύο κατηγορίες/τύποι ρομποτικών χειρών. Η πρώτη κατηγορία είναι οι υπό-οδηγούμενες ρομποτικές χείρες (under-actuated) και η δεύτερη κατηγορία είναι οι πλήρως οδηγούμενες ρομποτικές χείρες (fully-actuated). Οι κατηγορίες αυτές επεξηγούνται στις επόμενες παραγράφους.

Εδώ, είναι σημαντικό να αναφερθεί πως **βαθμός ελευθερίας** (degree of freedom) είναι η ικανότητα ενός αντικειμένου (π.χ. μία ράβδου) να κινείται γύρω από ένα άξονα. Έτσι, κάθε βαθμός ελευθερίας ισοδυναμεί με μία άρθρωση ενός απλού ρομποτικού βραχίονα που μπορεί να κινείται γύρω από ένα άξονα όπως αυτό στην εικόνα 3. Στην περίπτωση της ρομποτικής χείρας ένας βαθμός ελευθερίας αντιστοιχεί στην κάθε άρθρωση του κάθε δαχτύλου.



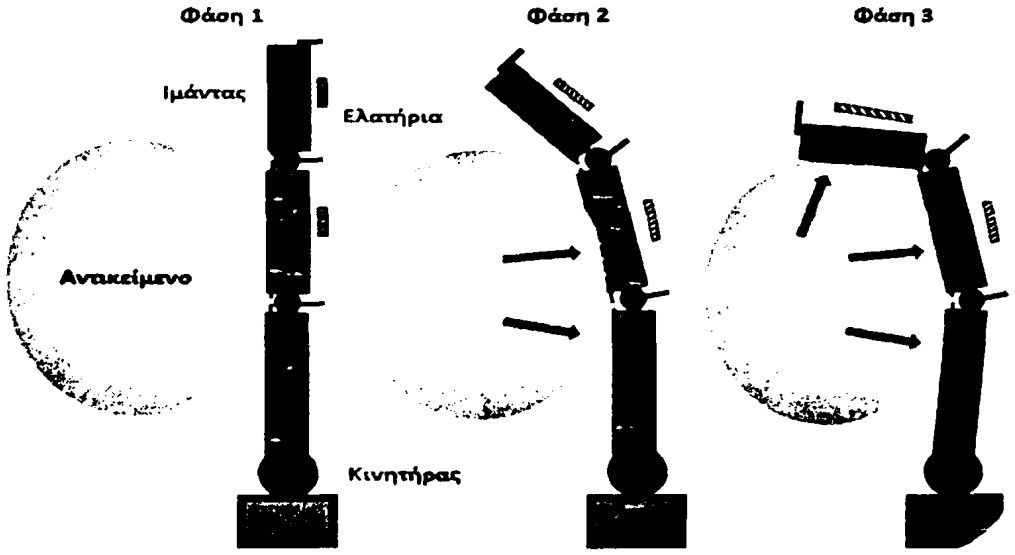
Εικόνα 3: Βαθμοί Ελευθερίας

Ένας **υπό-οδηγούμενος** [1] ρομποτικός μηχανισμός είναι αυτός ο οποίος διαθέτει λιγότερους κινητήρες μετάδοσης της κίνησης από ότι είναι οι βαθμοί ελευθερίας του (Degrees of Freedom, DoF).

Όταν αυτή η μέθοδος εφαρμόζεται σε ρομποτικά δάχτυλα, η έννοια της υπό-οδήγησης οδηγεί στην αυτόματη προσαρμογή. Τα ρομποτικά δάχτυλα θα περικλείουν το αντικείμενο που πρόκειται να πιάσουν και αυτόματα θα προσαρμόζονται στο σχήμα του αντικειμένου με χρήση μόνο ενός κινητήρα (π.χ.) χωρίς σύνθετες στρατηγικές ελέγχου των ρομποτικών δαχτύλων. Από την άλλη πλευρά όμως η ροπή

ενός κινητήρα πρέπει να τροφοδοτήσει περισσότερες από μία αρθρώσεις.

Αυτό σημαίνει πως η παραγόμενη δύναμη μοιράζεται στις φάλαγγες του δαχτύλου και το κράτημα ενός αντικειμένου γίνεται πιο δύσκολα. Για να δημιουργηθεί μία τέτοια τοπολογία θα πρέπει να χρησιμοποιηθούν στο σύστημα ελαστικά στοιχεία (π.χ. ελατήρια) ή ιμάντες με τροχαλίες, και αισθητήρες μηχανικών ορίων. Κατά τη διάρκεια που ένα τέτοιο δάχτυλο κλείνει ένα αντικείμενο, η λειτουργία του δαχτύλου κάθε χρονική στιγμή εξαρτάται από τους εξωτερικούς περιορισμούς που ορίζονται από το αντικείμενο που πρόκειται να κρατηθεί. Όταν το αντικείμενο κρατηθεί πλήρως η δύναμη που ασκείται από το αντικείμενο στον κινητήρα, αποδίδεται στις φάλαγγες του ρομποτικού δαχτύλου. Μια διαδικασία συγκράτησης ενός αντικειμένου φαίνεται στην εικόνα 4. Τα βέλη αντιπροσωπεύουν τις δυνάμεις που ασκούνται στις φάλαγγες του ρομποτικού δαχτύλου.

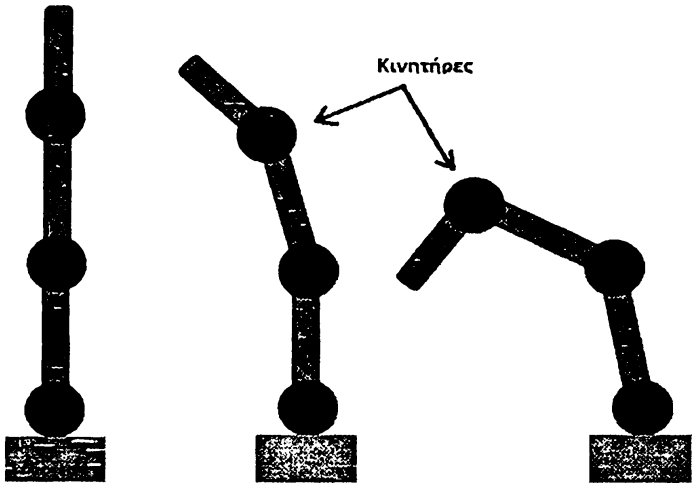


Εικόνα 4: Λειτουργία υπο-βοηθούμενου ρομποτικού δαχτύλου

Το ρομποτικό δάχτυλο κινείται μέσω ενός κινητήρα που βρίσκεται τοποθετημένος στο χαμηλότερο τμήμα του δαχτύλου. Εφόσον υπάρχουν τρείς βαθμοί ελευθερίας και ένας κινητήρας στο δάχτυλο (τρείς DoF – ένας κινητήρας), θα πρέπει να χρησιμοποιηθούν δύο ελαστικά στοιχεία στις υπόλοιπες δύο αρθρώσεις όπως είναι τα ελατήρια του παραδείγματος τα οποία τείνουν να κρατήσουν τις φάλαγγες του δαχτύλου σε πλήρη έκταση. Για την κάμψη των φαλάγγων είναι τοποθετημένος ιμάντας του οποίου η μία του άκρη συγκρατείται στην πάνω άκρη του ρομποτικού δαχτύλου και η άλλη του άκρη είναι δεμένη στον άξονα του κινητήρα. Έτσι λοιπόν όταν ο κινητήρας γυρίζει τον άξονα περιστροφής του ο ιμάντας έλκεται και

αντίστοιχα έλκονται και οι φάλαγγες του δαχτύλου. Είναι αυτονόητο πως η δύναμη που ασκείται από τον ιμάντα πάνω στις φάλαγγες πρέπει να είναι μεγαλύτερη από την δύναμη των ελατηρίων στο πίσω μέρος του δαχτύλου που τείνουν να κρατήσουν τις φάλαγγες σε πλήρη έκταση.

Ένας **πλήρως – οδηγούμενος** [2] ρομποτικός μηχανισμός είναι αυτός ο οποίος διαθέτει ίσο αριθμό κινητήρων μετάδοσης της κίνησης με τους βαθμούς ελευθερίας του. Όταν αυτή η μέθοδος εφαρμόζεται σε ρομποτικά δάχτυλα, η έννοια της πλήρους - οδήγησης οδηγεί σε ανεξάρτητη προσαρμογή. Αυτό σημαίνει πως οι φάλαγγες ενός ρομποτικού δαχτύλου θα είναι σε θέση να κινούνται ανεξάρτητα η μία από την άλλη. Αυτό δίνει την δυνατότητα της κάμψης των φαλάγγων όχι μόνο όταν ασκηθεί σε αυτές αντίθετη δύναμη από κάποιο αντικείμενο κατά το κράτημα αυτού, όπως παρουσιάζεται στην εικόνα 3, αλλά και όταν δεν υπάρχουν αντίθετες δυνάμεις, όταν δηλαδή οι φάλαγγες δεν έρχονται σε επαφή με κάποιο αντικείμενο. Στην επόμενη εικόνα αυτό γίνεται καλύτερα κατανοητό.



Εικόνα 5 : Λειτουργία πλήρους- οδηγούμενης ρομποτικής χείρας

Μηχανολογικά αυτή η μέθοδος έχει μικρότερο βαθμό δυσκολίας καθώς δεν απαιτούνται ελαστικά μέρη όπως ελατήρια (ή σε άλλες περιπτώσεις ιμάντες και τροχαλίες).

Επιπλέον αυτή η υλοποίηση απαιτεί πολύ πιο σύνθετες στρατηγικές ελέγχου των ρομποτικών δαχτύλων και προγραμματισμός για την κίνηση τριών ή τεσσάρων κινητήρων. Ένα ακόμη μειονέκτημα της μεθόδου αυτής είναι η κατανάλωση ισχύος. Η ισχύς που καταναλώνεται οφείλεται στους κινητήρες που αντικαθιστούν τις παθητικές διατάξεις. Δηλαδή, εκεί που υπήρχε ένας κινητήρας για τρείς αρθρώσεις

τώρα υπάρχουν τρείς και άρα το σύστημα τροφοδοτείται με τριπλάσιο ρεύμα και πολλαπλάσια κατανάλωση ισχύος.

Όσων αφορά τα πλεονεκτήματα αυτής της μεθόδου, το πρώτο είναι ο ρεαλισμός της κίνησης καθώς οι κινητήρες κινούν ανεξάρτητα τις αρθρώσεις των ρομποτικών δαχτύλων και φαίνονται περισσότερο φυσικές προσομοιώνοντας καλύτερα το ανθρώπινο χέρι. Το δεύτερο είναι η μεγάλη δύναμη που ασκούν οι φάλαγγες καθώς η ροπή κάθε ενός από τους κινητήρες συγκεντρώνεται και στην αντίστοιχη φάλαγγα του κάθε ρομποτικού δαχτύλου ενώ στις under-actuated διατάξεις η ροπή (και άρα η δύναμη) του κινητήρα μοιράζεται σε περισσότερες από μία αρθρώσεις και κατ' επέκταση φάλαγγες.

Λαμβάνοντας υπόψη τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μεθόδου όπως παρουσιάστηκαν στις προηγούμενες παραγράφους η μέθοδος που επιλέχθηκε για την κατασκευή της ρομποτικής χείρας της παρούσης διπλωματικής εργασίας είναι αυτή των πλήρως – οδηγούμενων ρομποτικών χειρών καθώς κύριοι στόχοι ήταν :

- *η ανεξάρτητη κίνηση των αρθρώσεων των δαχτύλων*
- *η αυξημένη δύναμη η οποία θα ασκείται από τις φάλαγγες των δαχτύλων του ρομποτικού χεριού σε σχέση με την άλλη μέθοδο (under-actuated)*
- *η απλότητα της κατασκευής (όσο αυτό ήταν δυνατό) καθώς η κατασκευή υλοποιήθηκε από μηδενική βάση.*
- *η φυσικότητα της κίνησης καθώς το ρομποτικό χέρι θα πρέπει να αντιγράφει κατά το καλύτερο δυνατό τις κινήσεις του πραγματικού χεριού μέσω διάταξης αισθητήρων.*

Η ρομποτική χείρα που κατασκευάστηκε αποτελείται από 19 βαθμούς ελευθερίας, ένας βαθμός ελευθερίας για κάθε άρθρωση της ρομποτικής χείρας.

Πιο συγκεκριμένα :

- *Τέσσερεις κινητήρες ανά δάχτυλο στα τέσσερα δάχτυλα: Δείκτη, Μέσο, Παράμεσο και Μικρό δάχτυλο. Οι τρείς κινητήρες θα πραγματοποιούν την κίνηση της έκτασης και της κάμψης και ο τέταρτος κινητήρας θα κινεί το κάθε δάχτυλο πλάγια (δεξιά αριστερά) σε σχέση με το διπλανό του δάχτυλο.*

- *Τρείς κινητήρες στον αντίχειρα. Δύο για την έκταση και κάμψη και ένας κινητήρας για το ανέβασμα και κατέβασμα του αντίχειρα.*

Εδώ επίσης θα πρέπει να αναφερθεί πως ο κάθε κινητήρας της ρομποτικής χείρας που κατασκευάσθηκε οδηγείται και από ένα αισθητήριο κάμψης. Συνολικά χρησιμοποιήθηκαν 17 αισθητήρια κάμψης. Καθώς ο αισθητήρας κάμψης της ακραίας φάλαγγας, του μικρού δαχτύλου, ήταν πολύ δύσκολο να τοποθετηθεί επάνω στη διάταξη αισθητήρων (Γάντι) λόγω των διαστάσεων του αφαιρέθηκε και η ακραία φάλαγγα ελέγχεται από το αισθητήριο της μεσαίας φάλαγγας του μικρού δαχτύλου. Επίσης, ο κινητήρας πλάγιας κίνησης του μέσου δαχτύλου της χείρας ελέγχεται έμμεσα από τα δύο αισθητήρια κάμψης που ελέγχουν τα διπλανά δάχτυλα δείκτη και παράμεσο.

Κεφάλαιο 1°

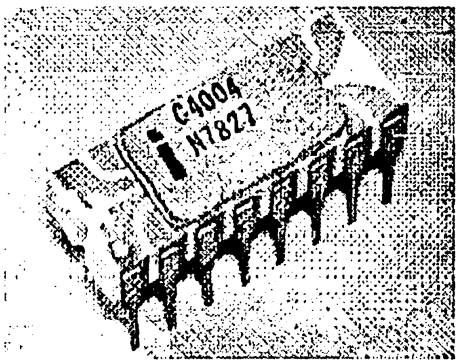
Μικροελεγκτές και Μικροεπεξεργαστές

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται ένα πολύ σημαντικό στοιχείο της διπλωματικής εργασίας που είναι οι μικροεπεξεργαστές και οι μικροελεγκτές, καθώς, χωρίς την χρήση τους η υλοποίηση του ελέγχου της κατασκευής με τον τρόπο που επιλέχθηκε να γίνει δεν θα ήταν δυνατή. Επίσης παρουσιάζονται συνοπτικά οι δύο διαφορετικές αρχιτεκτονικές με τις οποίες κατασκευάζονται οι σημερινοί μικροελεγκτές και αναφέρονται τα πλεονεκτήματα της κάθε μίας.

1.2 Τι είναι ο μικροεπεξεργαστής

Ένας μικροεπεξεργαστής [3] περιλαμβάνει τις περισσότερες ή όλες τις λειτουργίες μιας κεντρικής μονάδας επεξεργασίας (KME - CPU) ενός ηλεκτρονικού υπολογιστή σε ένα ενιαίο ολοκληρωμένο κύκλωμα (IC). Ο μικροεπεξεργαστής είναι ο πυρήνας του υπολογιστή, ελέγχει και κατευθύνει όλες τις εργασίες, κάνει υπολογισμούς και παίρνει αποφάσεις. Εάν ένας υπολογιστής μπορούσε να παρομοιασθεί με έναν άνθρωπο, ο μικροεπεξεργαστής θα ήταν ο εγκέφαλός του. Ο Intel 4004 ήταν ο πρώτος μικροεπεξεργαστής (εικόνα 5). Δημιουργήθηκε από τον Ted Hoff και το συνεργάτη του Stan Mazor και παρουσιάστηκε το 1971. Ο Intel 4004 ήταν ένας 4bit επεξεργαστής (ο μικροεπεξεργαστής λαμβάνει 4 bit από την μνήμη κάθε φορά με σκοπό να τα επεξεργαστεί), που αποτελούταν από περίπου 2300 τρανζίστορ με συχνότητα ρολογιού 108 KHz. Επιπλέον, εκτελούσε 60000 πράξεις το δευτερόλεπτο και μπορούσε να « δει » 640 bytes μνήμης. Η αρχική του εφαρμογή ήταν η δημιουργία αριθμομηχανών.



Εικόνα 5: Ο μικροεπεξεργαστής Intel 4004

1.2.1 Από τι αποτελείται ένας μικροεπεξεργαστής

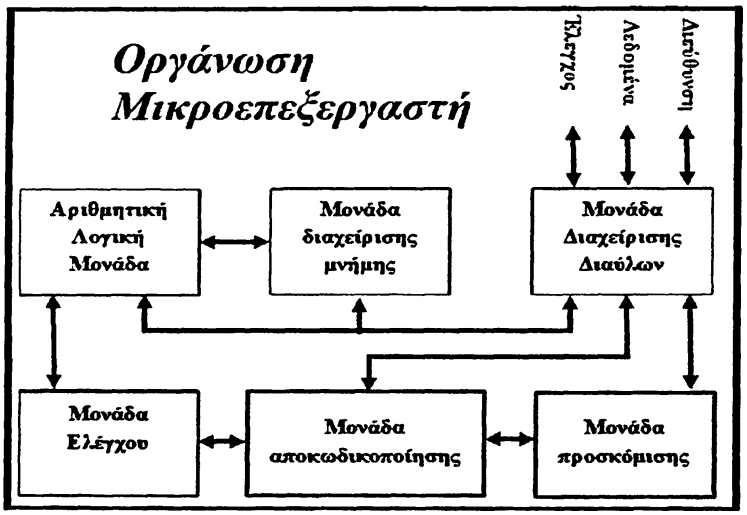
Ένας σύγχρονος μικροεπεξεργαστής αποτελείται από τις ακόλουθες μονάδες.

- **Αριθμητική και Λογική Μονάδα** (Arithmetic and Logic Unit, ALU): Η μονάδα στην οποία εκτελούνται μία προς μία οι αριθμητικές ή λογικές πράξεις, όπως υπαγορεύονται από τις εντολές που έχουν δοθεί στον υπολογιστή.
- **Μονάδα Διαχείρισης Μνήμης** (Memory Unit): Η μονάδα αυτή περιέχει μία προσωρινή μνήμη RAM πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller)
- **Μονάδα αποκωδικοποίησης** (Decoding Unit): Η μονάδα η οποία αποκωδικοποιεί ή μεταφράζει σύνθετες εντολές γλώσσας μηχανής σε απλούστερη μορφή τέτοια ώστε να γίνεται κατανοητή από την Αριθμητική – Λογική Μονάδα και τους καταχωρητές. Με τον τρόπο αυτό η επεξεργασία των δεδομένων γίνεται πιο αποδοτική.
- **Καταχωρητές** (Registers): Μικρά κελιά μνήμης στο εσωτερικό του επεξεργαστή, που χρησιμοποιούνται για την προσωρινή αποθήκευση των δεδομένων, καθώς αυτά υφίστανται επεξεργασία. Οι καταχωρητές διαφέρουν ανάλογα με τον τύπο του επεξεργαστή και τον κατασκευαστή, τόσο ως προς την οργάνωση όσο και ως προς τη χωρητικότητά τους.
- **Μονάδα ελέγχου** (Control Unit): Ελέγχει τη ροή δεδομένων από και προς την ALU, τους καταχωρητές, τη μνήμη και τις περιφερειακές μονάδες εισόδου/εξόδου.
- **Μονάδα προσκόμισης** (Fetch Unit): Μεταφέρει τις εντολές από τη μνήμη στον επεξεργαστή.
- **Μονάδα Διαχείρισης Διαύλων** (Bus Unit): Η μονάδα αυτή είναι υπεύθυνη για την μεταφορά δεδομένων από και προς τα διάφορα μέρη του επεξεργαστή.
- **Δίαυλος διευθύνσεων** (Address Bus) : Είναι μια ομάδα από καλώδια ή γραμμές που χρησιμοποιούνται για τη μεταφορά των διευθύνσεων της μνήμης ή I/O συσκευών.
- **Δίαυλος Δεδομένων** (Data Bus): Χρησιμοποιείται για τη μεταφορά δεδομένων εντός του μικροεπεξεργαστή και μνήμης/συσκευών εισόδου ή

εξόδου. Είναι αμφίδρομη καθώς ο Μικροεπεξεργαστής απαιτεί για την αποστολή ή/και την λήψη δεδομένων.

- **Δίαυλος ελέγχου** (Control Bus): Ο μικροεπεξεργαστής χρησιμοποιεί τον δίαυλο ελέγχου για να επεξεργαστεί δεδομένα, δηλαδή τι πρέπει να κάνει με την επιλεγμένη θέση μνήμης. Ορισμένα σήματα ελέγχου, read, write, Opcode fetch κ.α. Διάφορες εργασίες εκτελούνται από τον μικροεπεξεργαστή με τη βοήθεια του διαύλου ελέγχου. Αυτός είναι ένας ειδικός δίαυλος, διότι όλα τα σήματα χρονισμού παράγονται σύμφωνα με το σήμα ελέγχου.

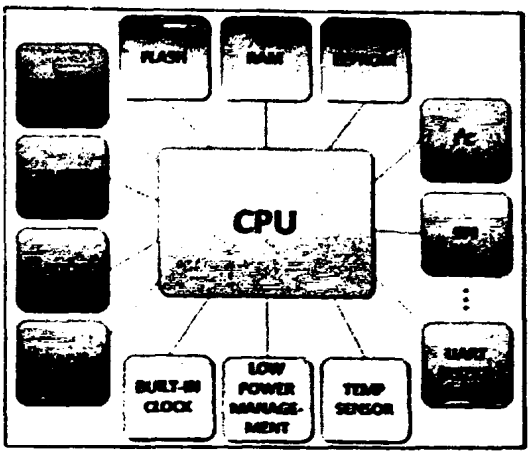
Στην εικόνα 6 παρουσιάζεται το σχηματικό διάγραμμα του εσωτερικού ενός μικροεπεξεργαστή.



Εικόνα 6: Οργάνωση Μικροεπεξεργαστή

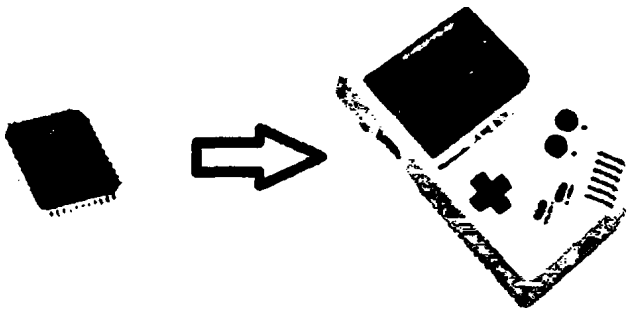
1.3 Τι είναι ο Μικροελεγκτής (uC)

Ο μικροελεγκτής (*microcontroller*) είναι ένα ολοκληρωμένο κύκλωμα το οποίο περιέχει τον Μικροεπεξεργαστή ως κεντρικό του στοιχείο αλλά επιπλέον έχει ενσωματωμένα πολλά περιφερειακά κυκλώματα καθιστώντας τον ικανό να λειτουργεί με ελάχιστα εξωτερικά εξαρτήματα. Θα μπορούσαμε να πούμε πως ο μικροελεγκτής είναι μία επέκταση του μικροεπεξεργαστή προορισμένος όμως για συγκριμένες εργασίες οι οποίες καθορίζονται σαφώς μέσω του προγράμματος το οποίο υλοποιείται από τον σχεδιαστή –προγραμματιστή μίας εφαρμογής.



Εικόνα 7: Οργάνωση Μικροελεγκτή

Οι μικροελεγκτές χρησιμοποιούνται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems), δηλαδή τα συστήματα που κατασκευάζονται με σκοπό να εκτελούν συγκεκριμένες εργασίες ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και οπουδήποτε υπάρχει η παρουσία ψηφιακής λογικής (εικόνα 8).



Εικόνα 8: Χρήση μικροελεγκτή στην κλασσική παιχνιδομηχανή Gameboy

1.3.1 Περιφερειακά ενός Μικροελεγκτή

Για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά υποσυστήματα ή περιφερειακά τα οποία είναι τοποθετημένα μέσα στο ίδιο ολοκληρωμένο (IC). Τέτοια είναι:

- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στην αρτηρία δεδομένων (bus) του επεξεργαστή.

- Μνήμη προγράμματος (FLASH, EPROM) η οποία περιέχει το λογισμικό του συστήματος και η οποία διατηρεί τα περιεχόμενα της και μετά την αφαίρεση της τροφοδοσίας (non-volatile). Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μνήμη SRAM (Static RAM), στην οποία αποθηκεύονται προσωρινά παράμετροι και δεδομένα που προκύπτουν από τη λειτουργία του προγράμματος, όπως το αποτέλεσμα μίας πρόσθεσης ή η ανάθεση μιας τιμής σε κάποια μεταβλητή. Με την αφαίρεση της τροφοδοσίας τα δεδομένα χάνονται (volatile).
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται στον πυρήνα του μικροελεγκτή κρατώντας τα δεδομένα και μετά την αφαίρεση της τροφοδοσίας (non-volatile). Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα ανεκτά όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (stall).
- Τοπικό ταλαντωτή (κρύσταλλο) για την παροχή παλμών χρονισμού (clock).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output)

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Έτσι, υπάρχουν:

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως *ROM-less*. Αυτοί παρέχουν πάντοτε ένα παράλληλο δίαυλο (bus) δεδομένων, πάνω στην οποία συνδέονται εξωτερικές μνήμες προγράμματος και RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM, η οποία κατασκευάζεται με το λογισμικό της (Mask ROM) ή γράφεται μόνο μια φορά (One Time Programmable, OTP). Παρέχουν τη δυνατότητα πολύ χαμηλού κόστους, όταν αγοράζονται σε πολύ μεγάλες ποσότητες.
- Μικροελεγκτές με μνήμη FLASH, η οποία μπορεί να προγραμματιστεί πάρα πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming, ISP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από το ειδικό τζαμάκι).

1.3.2. Πρόσθετες λειτουργίες και περιφερειακά

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (πχ I^2C, SPI, Ethernet).
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών

ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.

- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, π.χ. κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

1.4 Διαδεδομένες κατηγορίες μικροελεγκτών

- Μικροελεγκτές (8-bit και 16-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I^2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό.
- Μικροελεγκτές (32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM.
- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

1.5 Γλώσσες προγραμματισμού και εργαλεία ανάπτυξης

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία

εκσφαλμάτωσης (debuggers). Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα η μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

1.6 Ομοιότητες και διαφορές μεταξύ Μικροεπεξεργαστή και Μικροελεγκτή

Σε αυτό το σημείο θα γίνει μία συνοπτική περιγραφή των ομοιοτήτων και των διαφορών των δύο αυτών ολοκληρωμένων κυκλωμάτων καθώς πολλές φορές οι όροι μικροεπεξεργαστής και μικροελεγκτής προκαλούν σύγχυση όσων αφορά τη λειτουργία τους και τον τρόπο χρήσης τους.

Στους σύγχρονους μικροεπεξεργαστές (πχ τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη για μία μόνο εργασία.

Αντίθετα, στους μικροελεγκτές, οι οποίοι έχουν μικρότερες δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, καθώς σε αυτούς δίνεται ιδιαίτερο βάρος στον μεγάλο αριθμό προ-εγκατεστημένων περιφερειακών συσκευών η ευελιξία είναι περιορισμένη, όπως και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση. Δηλαδή χρησιμοποιούνται για την εκτέλεση συγκεκριμένων εργασιών .

Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.

Πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν καθιστώντας τους έτσι τελείως αυτόνομους. Δηλαδή εκεί που θα χρειαζόμασταν δέκα ολοκληρωμένα κυκλώματα για την κατασκευή ενός συστήματος με τον μικροελεγκτή χρειαζόμαστε μόνο ένα .

Το κόστος τους είναι ιδιαίτερα χαμηλό σε σχέση με τους μικροεπεξεργαστές και η αξιοπιστία που προσδίδουν είναι μεγαλύτερη λόγω των λιγότερων εξωτερικών διασυνδέσεων . Επίσης υπάρχουν περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.

Τέλος θα πρέπει να αναφερθεί πως η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους χρησιμοποιείται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (πχ οι σειρές AVR της Atmel και PIC της Microchip). Στους κοινούς μικροεπεξ/τές συνηθίζεται η ενιαία διάταξη μνήμης τύπου Von Neumann.

1.7 Αρχιτεκτονικές επεξεργαστών

Στο τέλος της προηγούμενης ενότητας έγινε μία μικρή αναφορά στις δύο διαφορετικές αρχιτεκτονικές που διέπουν τους επεξεργαστές. Σε αυτό το κεφάλαιο θα αναλυθούν αυτές οι δύο αρχιτεκτονικές [3][4] παρουσιάζοντας την λειτουργία τους καθώς και τα πλεονεκτήματα της μίας έναντι της άλλης.

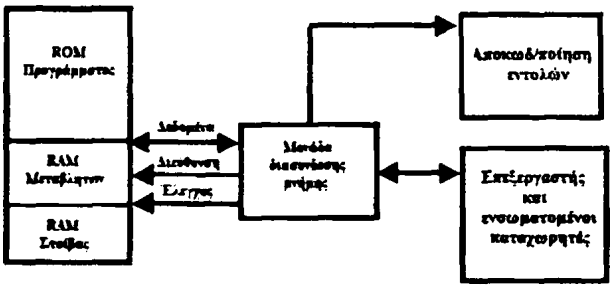
1.7.1 Harvard και Princeton

Πολλά χρόνια πρίν, η κυβέρνηση των Η.Π.Α. ζήτησε από τα πανεπιστήμια Harvard και Princeton να δημιουργήσουν μία αρχιτεκτονική υπολογιστή, η οποία θα χρησιμοποιούνταν για τον υπολογισμό πινάκων με τις αποστάσεις βολής του Πολεμικού Ναυτικού για μεταβαλλόμενες κλίσεις και καιρικές συνθήκες. Η απάντηση του Princeton ήταν ένας υπολογιστής με συνηθισμένη μνήμη για αποθήκευση του προγράμματος ελέγχου, διαφόρων μεταβλητών και άλλων δομών δεδομένων. Έγινε γνωστός με το όμομα του επικεφαλή της ομάδας "Von Neumann".

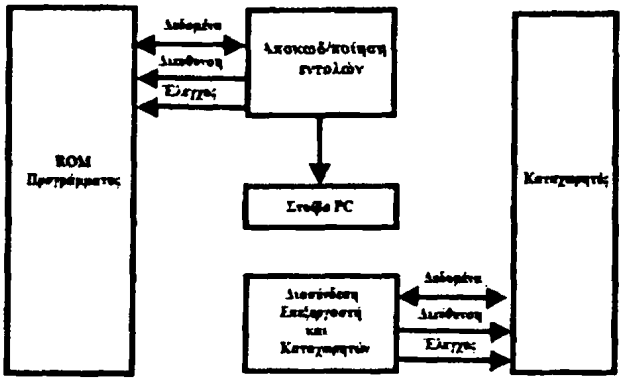
Η μονάδα διασύνδεσης της μνήμης (memory inteface unit) ευθύνεται για την κατανομή της προσπέλασης στο χώρο της μνήμης, τόσο για την ανάγνωση των

εντολών, όσο και για τη διέλευση των δεδομένων μεταξύ του επεξεργαστή και των εσωτερικών καταχωρητών του. Μπορεί αρχικά να φαίνεται ότι η μονάδα διασύνδεσης μνήμης αποτελεί μία στενωπό μεταξύ του επεξεργαστή και του χώρου μεταβλητών και RAM – ιδιαίτερα με την απαίτηση για ταυτόχρονη ανάκλιση εντολών. Ωστόσο σε πολλούς επεξεργαστές, αρχιτεκτονικής Princeton, δεν συμβαίνει το ίδιο, γιατί ο χρόνος που απαιτείται για την εκτέλεση μίας εντολής μπορεί να χρησιμοποιηθεί για την ανάκληση της επόμενης εντολής (λειτουργία pre-fetching) και αποτελεί χαρακτηριστικών πολλών επεξεργαστών, τύπου Princeton (σχήμα 1).

Αντίθετα, η απάντηση του Harvard ήταν μία σχεδίαση που χρησιμοποιούσε χωριστές περιοχές μνήμης για την αποθήκευση του προγράμματος, τη στοίβα (stack) του επεξεργαστή και τη RAM των μεταβλητών (σχήμα 2). Η αρχιτεκτονική Princeton κέρδισε τον ανταγωνισμό γιατί ήταν η πιο κατάλληλη για την τεχνολογία της εποχής εκείνης. Ήταν προτιμότερο να χρησιμοποιηθεί μία μόνο μνήμη, λόγω της αναξιοπιστίας των ηλεκτρονικών εκείνης της εποχής (δεν είχαν διαδοθεί ακόμη τα τρανζίστορ). Μία μόνο βαθμίδα διασύνδεσης μνήμης παρουσίαζε λιγότερες πιθανότητες σφάλματος.



Σχήμα 1. Von Neumann Block διάγραμμα



Σχήμα 2. Harvard Block διάγραμμα

Η αρχιτεκτονική Harvard αγνοήθηκε μέχρι τα τέλη της δεκαετίας του΄70, όταν οι κατασκευαστές μικροελεγκτών κατάλαβαν ότι αυτή η αρχιτεκτονική είχε

πλεονεκτήματα, τα οποία μπορούσαν να εκεμεταλλευτούν οι διατάξεις που τώρα πια κατασκεύαζαν.

1.7.2 Πλεονεκτήματα Αρχιτεκτονικών

Το μεγαλύτερο πλεομέκτημα της αρχιτεκτονικής Von Neumann είναι ότι απλοποιεί τη σχεδίαση του ολοκληρωμένου του μικροελεγκτή, αφού δίνει προσπέλαση σε μόνο μία μνήμη. Για τους μικροελεγκτές το μεγαλύτερο πλεονέκτημα είναι ότι η RAM μπορεί να χρησιμοποιηθεί τόσο για την αποθήκευση μεταβλητών όσο και για την αποθήκευση εντολών του προγράμματος. Ένα πλεονέκτημα σε μερικές εφαρμογές, είναι ότι το πρόγραμμα μπορεί να προσπελάσει τα περιεχόμενα της στοίβας του μετρητή προγράμματος (Program Counter). Αυτό δίνει μεγαλύτερη ευελιξία για την ανάπτυξη λογισμικού κυρίως στα συστήματα πραγματικού χρόνου.

Η αρχιτεκτονική Harvard από την άλλη πλευρά, εκτελεί εντολές σε λιγότερους κύκλους μηχανής απ'ότι η αρχιτεκτονική Von Neumann. Αυτό είναι δυνατό γιατί στην αρχιτεκτονική Harvard μπορούμε να έχουμε μεγαλύτερο παραλληλισμό εντολών. Η λειτουργία αυτή έγγειται στην ανάκληση της επόμενης εντολής κατά τη διάρκεια της τρέχουσας εντολής χωρίς η εκτέλεση του προγράμματος να χρειάζεται να περιμένει για ένα «νεκρό κύκλο» για την ολοκλήρωση της εντολής. Η λειτουργία αυτή καλείται επίσης και Pipeline. Οι λόγοι γι'αυτό θα αναλυθούν στην επόμενη παράγραφο.

1.8 Τεχνική Pipeline

Η τεχνική διοχέτευσης / παραλληλισμού [5] χρησιμοποιείται στο σχεδιασμό των υπολογιστών έτσι ώστε να αυξάνεται ο αριθμός εντολών που μπορεί να εκτελεστεί στη μονάδα του χρόνου (throughput). Ο βασικός κύκλος για την ολοκλήρωση μιας εντολής χωρίζεται σε επιμέρους στάδια. Αυτή η διαδικασία ονομάζεται pipeline.Οι παλαιότεροι επεξεργαστές εκτελούσαν τις εντολές τελείως σειριακά, η πρώτη εντολή ξεκινούσε την εκτέλεση, ολοκληρωνόταν και μετά άρχιζε να εκτελείται η επόμενη. Το πρόβλημα με αυτό είναι ότι είναι μη αποδοτικό, καθώς η εκτέλεση γινόταν σε βήματα. Έπρεπε να ολοκληρωθούν όλα τα βήματα της εντολής 1 για να ξεκινήσει η εκτέλεση της εντολής 2, σαν σε μια γραμμή συναρμολόγησης ένας εργάτης στην αρχή της γραμμής να πρέπει να περιμένει να τελειώσει ο εργάτης στο

τέλος της γραμμής έτσι ώστε να ξεκινήσει να κάνει κάτι άλλο. Σε κάθε χρονική στιγμή όλοι οι εργάτες της γραμμής συναρμολόγησης θα κάθονται πλην ενός.

Φυσικά σε μια τέτοια γραμμή τα πράγματα δεν γίνονται ακριβώς έτσι. Κάθε εργάτης μόλις ολοκληρώσει ένα τμήμα το δίνει στον αμέσως επόμενό του και ξεκινάει να δημιουργεί ένα καινούργιο. Συνεπώς όλοι οι εργάτες δουλεύουν συνεχώς. Οι σύγχρονοι επεξεργαστές κάνουν ακριβώς την ίδια λειτουργία με τις εντολές.

Το πρώτο βήμα της εκτέλεσης μιας εντολής εκτελείται και όταν η εντολή περάσει στο επόμενο βήμα, αρχίζει μια καινούρια εντολή. Αυτή η διαδικασία ονομάζεται **διοχέτευση**. Τα βήματα στη διοχέτευση συχνά αποκαλούνται και **στάδια (stages)**.

Η διοχέτευση οδηγεί σε μεγάλη αύξηση της απόδοσης του συστήματος, συγκρινόμενη με τη σειριακή εκτέλεση εντολών, στην οποία πολλά κυκλώματα του επεξεργαστή παρέμεναν ανενεργά. Σε όσα περισσότερα στάδια διαχωριστεί η διοχέτευση, τόση μεγαλύτερη, θεωρητικά, ταχύτητα θα έχει το σύστημα μας.
Η τεχνική διοχέτευσης αυξάνει το throughput αλλά δεν μειώνει τον χρόνο για να ολοκληρωθεί μία απλή εντολή από την αρχή μέχρι το τέλος της εφόσον θα πρέπει υποχρεωτικά να περάσει από όλα τα επιμέρους βήματα.
Οι πραγματικές φάσεις εκτέλεσης ποικίλλουν από επεξεργαστή σε επεξεργαστή, με κάποιους πιο προοδευμένους να χρησιμοποιούν μικρότερα βήματα από άλλους. Ωστόσο, όλοι ακολουθούν τα ίδια βασικά βήματα.

1. Προσκόμιση της επόμενης εντολής από τη μνήμη και αποθήκευση της στον καταχωρητή εντολών (IR).
2. Προσδιορισμός του τύπου της εντολής που προσκομίστηκε.
3. Ανάκτηση των δεδομένων, αν χρειάζεται και αποθήκευση τους σε εσωτερικούς καταχωρητές του επεξεργαστή.
4. Εκτέλεση της εντολής.
5. Αποθήκευση των αποτελεσμάτων στην κατάλληλη θέση.

Η ακολουθία των βημάτων είναι γνωστή ως **(fetch - decode – execute)**.

Στην εικόνα 9, βλέπουμε σχηματικά τη λειτουργία της τεχνικής διοχέτευσης που χρησιμοποιείται από ένα επεξεργαστή με όλα τα επιμέρους βήματα στα οποία

τμηματοποιείται μία εντολή. Με αυτόν τον τρόπο πραγματοποιείται η λειτουργία του παραλληλισμού που αναφέρθηκε στην προηγούμενη ενότητα στην αρχιτεκτονική Harvard.

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Εικόνα 9: Pipeline 5 επιπέδων

Basic five-stage pipeline in a RISC machine (**IF** = Instruction Fetch, **ID** = Instruction Decode, **EX** = Execute, **MEM** = Memory access, **WB** = Register write back). In the fourth clock cycle (the green column), the earliest instruction is in MEM stage, and the latest instruction has not yet entered the pipeline

Αυτή την τεχνική χρησιμοποιούν και οι μικροελεγκτές AVR της Atmel οι οποίοι θα χρησιμοποιηθούν για την υλοποίηση του συστήματος της διπλωματικής εργασίας.

Έτσι, με σκοπό τη μεγιστοποίηση της εκτέλεσης εντολών ανά μονάδα χρόνου στους μικροελεγκτές AVR χρησιμοποιείται τεχνική διοχέτευσης δύο σταδίων ή δύο επιπέδων. Αυτό σημαίνει πως ενώ μία εντολή εκτελείται ($1^{ο}$ στάδιο) ταυτόχρονα μεταφέρεται και η επόμενη εντολή από τη μνήμη ($2^{ο}$ στάδιο) η οποία εκτελείται με τη σειρά της στον επόμενο κύκλο ρολογιού κ.ο.κ.

Συμπεραίνοντας , σε ένα σύστημα RISC, οι εντολές είναι όσο το δυνατόν λιγότερες, έτσι ώστε να επιτρέπει στον χρήστη, και όχι στον σχεδιαστή, να σχεδιάζει τις λειτουργίες που θέλει.

1.9 Αρχιτεκτονική RISC των μικροελεγκτών

Από το 1960 σε όλους τους μικροελεγκτές οι σχεδιαστές τοποθετούσαν όσο το δυνατό περισσότερες εντολές μπορούσαν μέσα στη CPU. Μερικές από αυτές τις εντολές εκτελούσαν πολύπλοκες λειτουργίες. Ένα παράδειγμα είναι η πρόσθεση δεδομένων διευθύνσεων μνήμης και η αποθήκευση του αποτελέσματος σε μία μνήμη.

Οι σχεδιαστές των μικροεπεξεργαστών ακολούθησαν αυτή τη μέθοδο. Επειδή αυτοί οι μικροεπεξεργαστές χρησιμοποιούσαν ένα μεγάλο αριθμό εντολών, πολλές από τις οποίες εκτελούσαν πολύ πολύπλοκες λειτουργίες, έγιναν γνωστοί ως CISC (Complex Instruction Set Computer) επεξεργαστές.

Με το πέρασμα των ετών όμως παρατηρήθηκε πως πολλές από αυτές τις πολύπλοκες εντολές, που ήταν εγκατεστημένες στη CPU, δεν χρησιμοποιούνταν ποτέ από τους προγραμματιστές και τους compilers. Το μεγάλο κόστος της τοποθέτησης αυτών των εντολών στον μικροεπεξεργαστή συν το γεγονός ότι ένα μεγάλο μέρος των τρανζίστορ στο ολοκληρωμένο χρησιμοποιούνταν από τον αποκωδικοποιητή εντολών οδήγησε κάποιους σχεδιαστές να σκεφτούν την απλοποίηση και μείωση του αριθμού των εντολών. Καθώς αυτό το σενάριο υλοποιήθηκε τελικά , οι μικροεπεξεργαστές που προέκυψαν έγιναν γνωστοί ως RISC (Reduced Instruction Set Computer)

1.9.1 Χαρακτηριστικά μικροελεγκτών RISC

Κάποια από τα σημαντικότερα χαρακτηριστικά είναι τα εξής:

1. Οι επεξεργαστές RISC έχουν συγκεκριμένο μέγεθος εντολών 2 bytes με ελάχιστες εξαιρέσεις εντολών με μέγεθος 4- bytes.
2. Ο μεγάλος αριθμός καταχωρητών. Όλες οι αρχιτεκτονικές RISC έχουν τουλάχιστον 32 καταχωρητές. Από αυτούς λίγοι μόνο είναι συνδεδεμένοι με μία συγκεκριμένη λειτουργία.
3. Ο μικρός αριθμός εντολών. Οι RISC επεξεργαστές έχουν μόνο βασικές εντολές όπως Add, Sub, Mul, Load, Store, For, And, Or, Call, Jump κ.α.
4. Στους RISC μικροελεγκτές το 95% των εντολών εκτελείται σε ένα κύκλο ρολογιού σε αντίθεση με τους CISC.
5. Οι RISC μικροελεγκτές έχουν διαφορετικούς διαύλους για δεδομένα και κώδικα σε αντίθεση με τους CISC όπου για να προσπελαστεί ένα κομμάτι της μνήμης άσχετα με το αν περιέχει κώδικα η δεδομένα, θα πρέπει να χρησιμοποιηθούν ο ίδιος δίαυλος διευθύνσεων και δεδομένων.

1.10 Ανακεφαλαίωση

Σε αυτό το κεφάλαιο έγινε μία γενική περιγραφή του μικροεπεξεργαστή και του μικροελεγκτή , των μερών από τα οποία δομούνται. Επίσης, παρουσιάστηκαν συνοπτικά οι διαφορές τα πλεονεκτήματα και τα μειονεκτήματα του ενός ως προς το άλλο. Στη συνέχεια, παρουσιάστηκαν οι δύο αρχιτεκτονικές υλοποίησης των μικροελεγκτών και επισημάνθηκαν οι ομοιότητες και διαφορές τους. Τέλος, αναλύθηκε η τεχνική pipeline που διέπει τους σύγχρονους μικροελεγκτές καθώς και η αρχιτεκτονική RISC των μικροελεγκτών (όπως αυτός που θα χρησιμοποιηθεί για την υλοποίηση της διπλωματικής εργασίας) έναντι της αρχιτεκτονικής CISC.

ΚΕΦΑΛΑΙΟ 2^{ο}

Ο Μικροελεγκτής της Ρομποτικής Χείρας

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει αναφορά στον μικροελεγκτή που επιλέχθηκε, για τον έλεγχο της ρομποτικής χείρας, και στους λόγους για τους οποίους επιλέχθηκε καθώς και στα περιφερειακά που τον διέπουν και είναι απαραίτητα στοιχεία για την υλοποίηση του ελέγχου του συστήματος. Επίσης θα αναλυθούν τα πιο σημαντικά λειτουργικά χαρακτηριστικά του μικροελεγκτή όπως η Αριθμητική και Λογική Μονάδα (ALU), ο καταχωρητής κατάστασης (SREG), ο μετρητής προγράμματος (Program Counter) και ο δείκτης στοίβας (Stack Pointer).

2.2 Έλεγχος της ρομποτικής χείρας

Μετά από σχετική έρευνα, επιλέχθηκε, ως καταλληλότερος, ο μικροελεγκτής AVR ATmega 2560 καθώς έχει διπλάσια κανάλια ADC (16) και πολλαπλάσια σήματα διαμόρφωσης εύρους παλμών (PWM) από οποιοδήποτε άλλο μικροελεγκτή της κατηγορίας του. Πιο συγκεκριμένα, διαθέτει:

- Μέχρι 12 σήματα **16-bit** PWM, 3 σήματα PWM / χρονιστή
- Μέχρι 16 κανάλια ADC (single – ended)
- Χρήση κρυστάλλου 16MHz
- Προγραμματισμό σε γλώσσα embedded C [6].
- Μεταφορά του κώδικα (.hex) μέσω Programmer ISP [7].

Για τον έλεγχο της ρομποτικής χείρας χρειάζονται **19** σήματα PWM και **17** κανάλια ADC τουλάχιστον. Όπως γίνεται φανερό ο μικροελεγκτής αυτός [8], παρέχει επτά σήματα PWM και ένα κανάλι ADC λιγότερα από αυτά που χρειάζονται για την υλοποίηση του συστήματος.

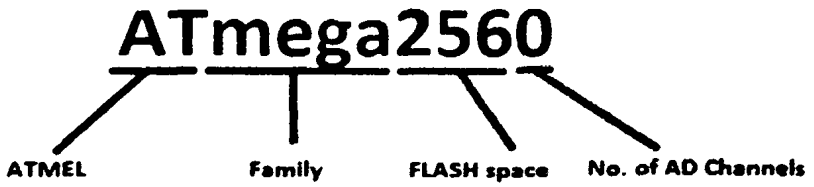
Έτσι έπρεπε ή να χρησιμοποιηθεί μία διάταξη με ένα μικροελεγκτή και μία τοπολογία η οποία θα παρήγαγε έξι επιπλέον κανάλια PWM ή, η χρήση δύο μικροελεγκτών.

Υλοποιήθηκαν και οι δύο μέθοδοι και αφού εξετάστηκαν και δοκιμάστηκαν λεπτομερώς αποφασίστηκε η υλοποίηση του συστήματος ελέγχου με χρήση δύο μικροελεγκτών. Οι λόγοι για τους οποίους αποφασίστηκε αυτή η μέθοδος αναλύεται στο παράρτημα που βρίσκεται στο παράρτημα Π6 του τεύχους.

2.2.1 Ο μικροελεγκτής ATmega2560

Ο μικροελεγκτής ATmega2560 ανήκει στην κατηγορία AVR, 8-bit, τεχνολογίας CMOS και κατασκευάζεται από την εταιρία Atmel. AVR είναι η ονομασία που έχει δοθεί στους μικροελεγκτές της Atmel και αποτελείται από πολλές επιμέρους οικογένειες, ανάλογα με τα χαρακτηριστικά τους. Στην ίδια οικογένεια με τον ATmega2560 βρίσκονται και οι μικροελεγκτές ATmega1280 και ATmega640.

Η κύρια και σημαντικότερη διαφορά αυτών των τριών μικροελεγκτών είναι το μέγεθος της μνήμης FLASH στην οποία και μεταφέρεται / αποθηκεύεται το πρόγραμμα (.hex) που έχει γραφθεί σε γλώσσα embedded C. Στην εικόνα 10, αναλύεται η σημασία της ονομασίας του μικροελεγκτή και στην εικόνα 11, παρουσιάζονται οι διαφορές μεταξύ των τριών αυτών μικροελεγκτών της ίδιας κατηγορίας.



Εικόνα 10: Κωδικός μικροελεγκτή

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

Εικόνα 11: Βασικές διαφορές της ίδιας οικογένειας

Οι αριθμοί 640, 1280 και 2560 που αναγράφονται στην ονομασία των μικροελεγκτών υποδηλώνουν το μέγεθος της μνήμης FLASH σε kilobytes καθώς και των αριθμό των καναλιών μετατροπής Analog to Digital (AD) . Δηλαδή οι αριθμοί 64, 128 και 256 μας δείχνουν τα kilobytes της μνήμης ενώ ο αριθμός 0/1 στο τέλος υποδηλώνει πως ο μικροελεγκτής έχει 16/8 κανάλια μετατροπής AD .

2.2.2 Χαρακτηριστικά μικροελεγκτή

Ο μικροελεγκτής ATmega2560 είναι εφοδιασμένος με κάποια πολύ σημαντικά χαρακτηριστικά που τον καθιστούν ιδιαίτερα χρήσιμο στην ανάπτυξη ρομποτικών εφαρμογών όπως την υλοποίηση της ρομποτικής χείρας. Τα χαρακτηριστικά αυτά είναι τα παρακάτω:

- Αρχιτεκτονική RISC με χρήση pipeline δύο επιπέδων
  - Έως 16 MIPS (Mega Instructions/Second) με χρήση κρυστάλου 16MHz
  - 32 Καταχωρητές 8-bit γενικής χρήσης
  - On Chip πολλαπλασιαστής 2 κύκλων
  - 135 εντολές όπου οι περισσότερες απαιτούν μόνο 1 κύκλο μηχανής για την ολοκλήρωση τους
- Μνήμη (non volatile) προγράμματος και δεδομένων
  - 4 KBytes μνήμη EEPROM
  - 8 Kbytes μνήμη SRAM
  - 256 Kbytes μνήμη FLASH
- Προγραμματισμός μικροελεγκτή μέσω πρωτοκόλλων επικοινωνίας SPI και JTAG (για δυνατότητα debugging μέσω του Atmel Studio 6)
- Ειδικά Χαρακτηριστικά
  - Εσωτερικό ρολόι
  - Ειδικός Καταχωρητής διακρίβωσης του εσωτερικού ρολογιού
  - Αυτόματο Reset κατά την τροφοδότηση
  - Ταχύτητα έως και 16 MHz
- Διακοπές (interrupts)
  - 17 εσωτερικές διακοπές
  - 12 εξωτερικές διακοπές
- Περιφερειακά
  - 3 timers/counters 8-bit με δύο ανεξάρτητους συγκριτές ο καθένας

- 4 timers /counters 16-bit
- 12 κανάλια PWM 16 –bit
- 16 κανάλια μετατροπής AD ανάλυσης 10 – bit
- Σειριακή θύρα με δυνατότητα UART-USART
- Watchdog timer τριών καταστάσεων λειτουργίας
- Πρωτόκολλο επικοινωνίας I2C
- Αναλογικός Συγκριτής με δυνατότητα πολύπλεξης εισόδων

❖ 6 sleep modes για εξοικονόμηση ενέργειας

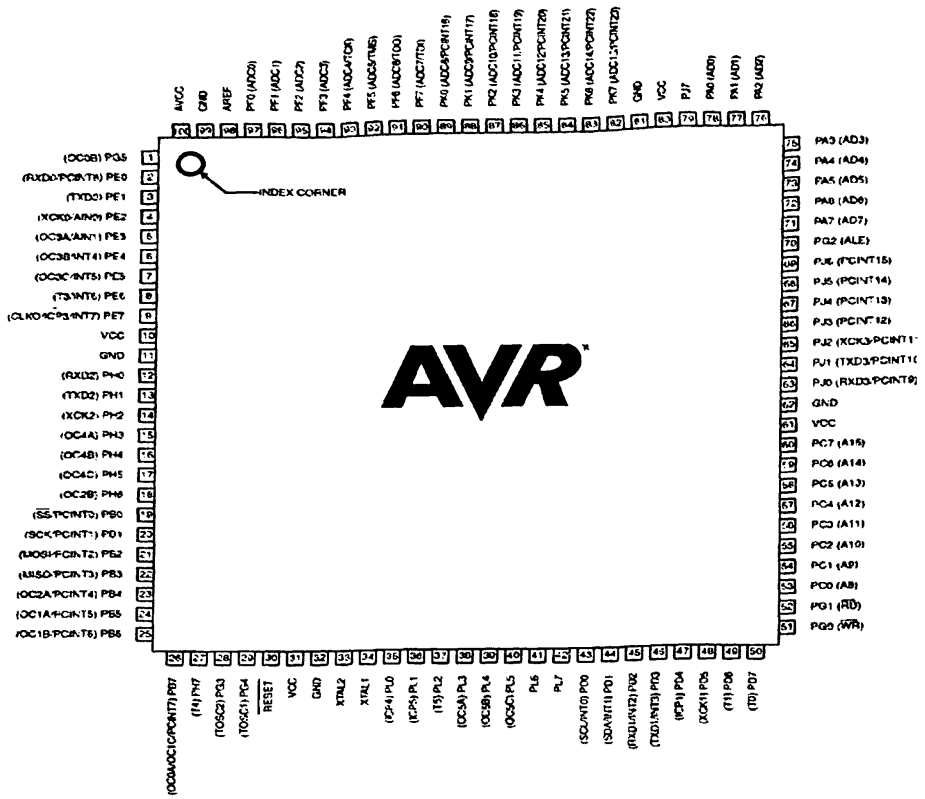
❖ Πολύ μικρή κατανάλωση (1.8V : 500μA) σε ταχύτητα 1MHz

2.2.3 Τα στοιχεία του μικροελεγκτή που χρησιμοποιήθηκαν

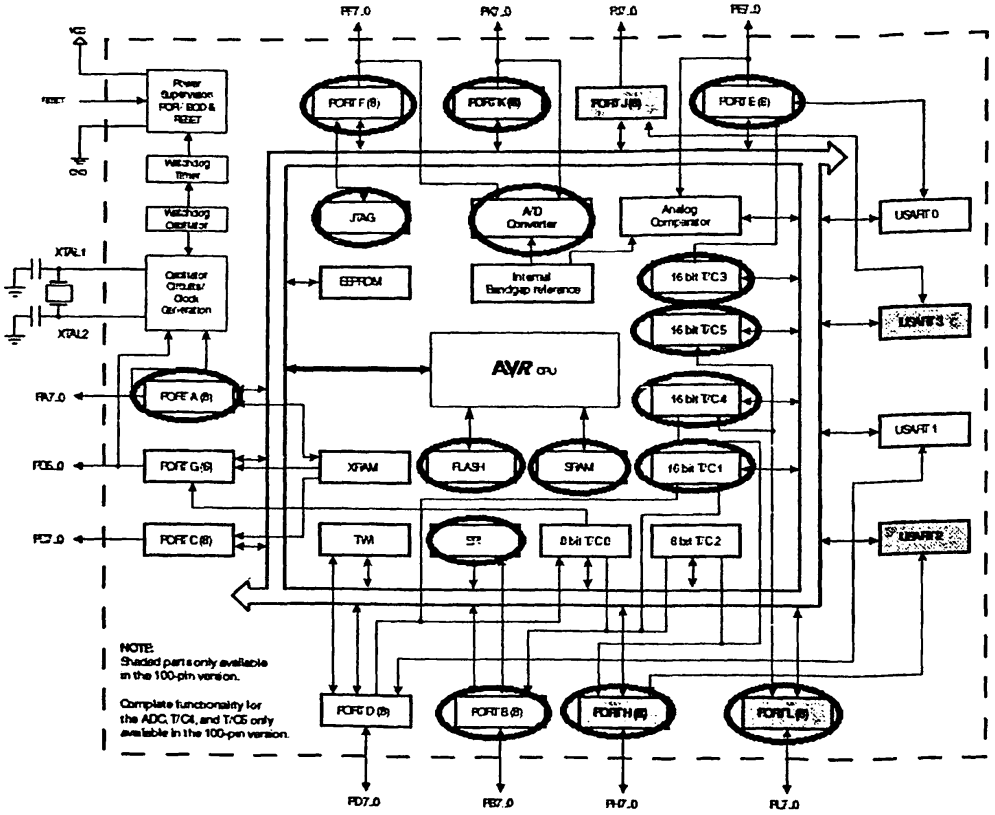
Τα χαρακτηριστικά του μικροελεγκτή που χρησιμοποιήθηκαν [8] για την υλοποίηση του ελέγχου της ρομποτικής χείρας είναι :

1) Κρύσταλλος 16MHz
2) Η μεταφορά του κώδικα στον μικροελεγκτή μέσω ISP
3) Τα 16 κανάλια μετατροπής AD
4) Οι 4 χρονιστές (timers) – 16 bit
5) Οι διακοπές (interrupts) των χρονιστών
6) Τα 12 PWM κανάλια των χρονιστών
7) Τρία GPIO ports του μικροελεγκτή για γενική χρήση

Στις εικόνες 12α και 12β, της επόμενης σελίδας , φαίνονται σε απλοποιημένη μορφή οι ακροδέκτες του μικροελεγκτή και οι λειτουργίες του αντίστοιχα ενώ σε κύκλο είναι οι λειτουργίες και τα ports που χρησιμοποιήθηκαν για το project.



Εικόνα 12α: Ακροδέκτες του ATmega2560



Εικόνα 12β: Στοιχεία του ATmega2560 που χρησιμοποιήθηκαν

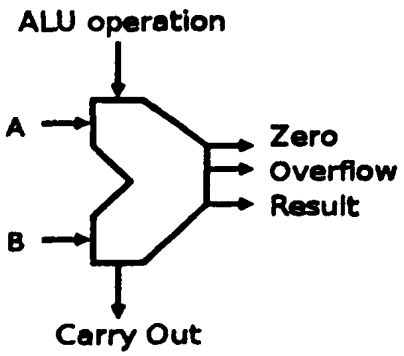
2.3 Αρχιτεκτονική Harvard στους AVR

Οι AVR [8] έχουν ένα εσωτερικό διάδρομο 8-bit για τη μεταφορά δεδομένων ανάμεσα στην ALU (Arithmetic Logic Unit) , τον SREG (Status Register), τον PC (Program Counter), την SRAM (Static Random Access Memory) και τα διάφορα περιφερειακά. Όλοι οι AVR χρησιμοποιούν μια παραλλαγή της αρχιτεκτονικής Harvard. Με την παραλλαγή αυτή, τα περιεχόμενα της μνήμης εντολών προσπελαύνονται ως απλά δεδομένα με τη χρήση ειδικών εντολών. Επίσης το πρόγραμμα και τα δεδομένα αποθηκεύονται σε ξεχωριστά συστήματα φυσικής μνήμης. Το πρόγραμμα, μια σειρά από εντολές οι οποίες εκτελούνται στην ALU, μεταφέρεται από μία διεύθυνση στη μνήμη FLASH, η οποία υποδεικνύεται κάθε φορά από τον Program Counter.

2.3.1 Η ALU των AVR

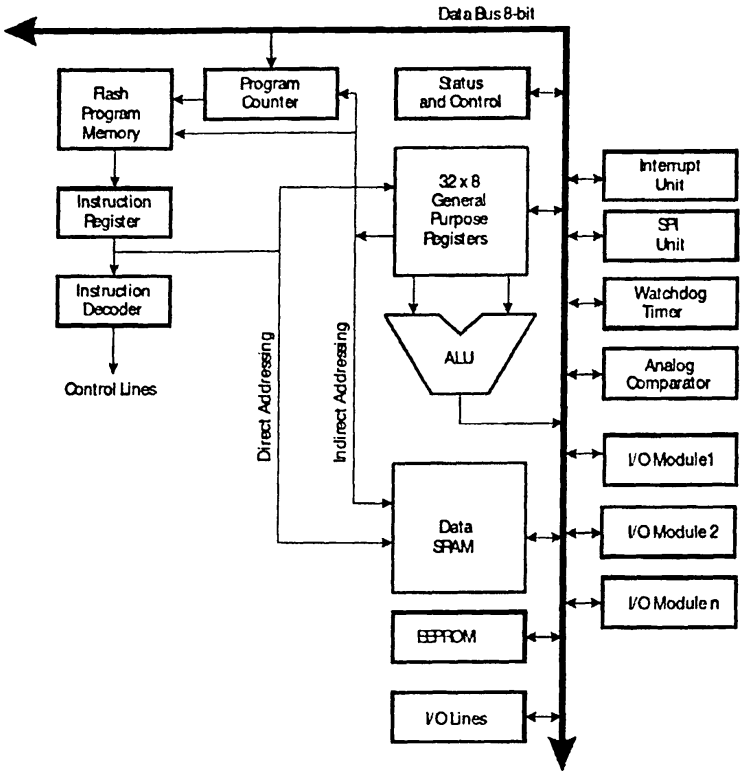
Η ALU [6,8] (εικόνα 13) διαθέτει 32 Καταχωρητές γενικής χρήσης (εικόνα 17) οι οποίοι λειτουργούν ως όροι πράξεων κατά την εκτέλεση των διαφόρων εντολών. Έξι από τους 32 καταχωρητές μπορούν να χρησιμοποιηθούν έμμεσα ως τρείς 16-bit δείκτες διευθύνσεων καταχωρητών για την διευθυνσιοδότηση των δεδομένων , δίνοντας τη δυνατότητα γρήγορης προσπέλασης των διευθύνσεων που βρίσκονται στη μνήμη. Η ALU δίνει την δυνατότητα αριθμητικών και λογικών πράξεων (AND, OR, ...) μεταξύ δύο καταχωρητών ή ενός καταχωρητή και μίας σταθεράς.

Μετά από μία αριθμητική πράξη ο Status Register ανανεώνεται εμφανίζοντας τα αποτελέσματα της λειτουργίας. Σαν παράδειγμα αναφέρεται το bit του κρατουμένου Carry Out στον SREG το οποίο μπορεί να γίνει "1" μετά την ολοκλήρωση μιας αριθμητικής η λογικής πράξης.



Εικόνα 13: Διεργασίες της ALU

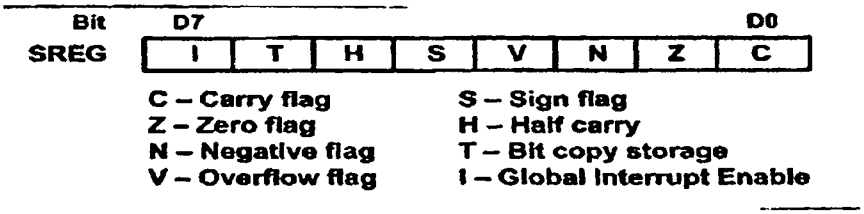
Στην εικόνα 14 φαίνεται το μπλοκ διάγραμμα της αρχιτεκτονικής των μικροελεγκτών AVR.



Εικόνα 14: Αρχιτεκτονική AVR

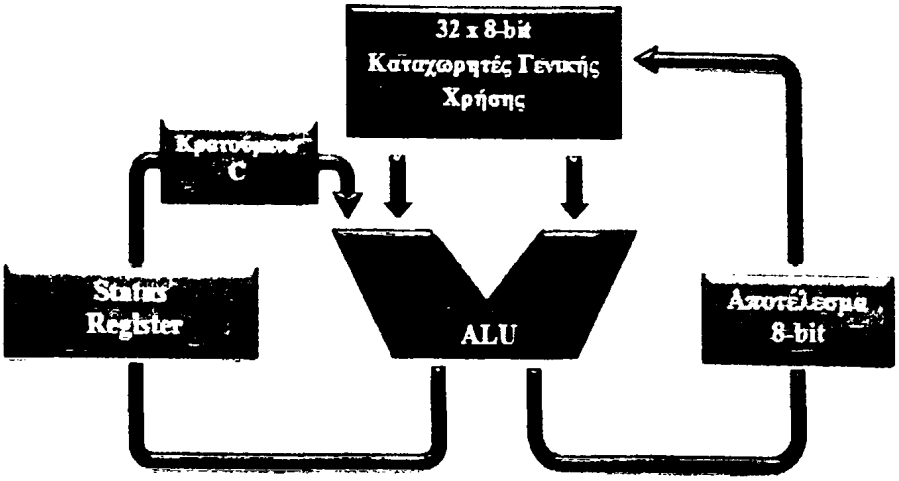
2.3.2 Καταχωρητής Κατάστασης Status Register (SREG)

Ο καταχωρητής κατάστασης [6,8] έχει μήκος 8-bit και περιέχει κάποιες πληροφορίες για το αποτέλεσμα της πιο πρόσφατης εκτελεσμένης αριθμητικής εντολής. Η πληροφορίες αυτές μπορούν να χρησιμοποιηθούν για την εναλλαγή της ροής του προγράμματος έτσι ώστε να εκτελεστούν λειτουργίες συνθηκών. Ο SREG ανανεώνεται εφόσον η ALU έχει τελειώσει όλες τις πράξεις της. Στην εικόνα 15 φαίνεται η διάταξη των bit στον Status Register (SREG) και των λειτουργιών τους.



Εικόνα 15: Τα bits του SREG

Στην εικόνα 16 φαίνεται η επικοινωνία του Status Register με την ALU και τους Καταχωρητές Γενικής Χρήσης.

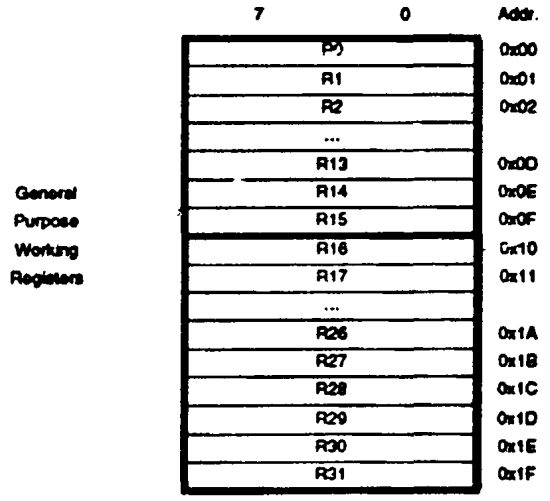


Εικόνα 16: Συνδεση μεταξύ ALU και Status Register

2.3.3 Καταχωρητές γενικής χρήσης των AVR

Οι καταχωρητές των AVR [6,8] (εικόνα 17), είναι κατασκευασμένοι έτσι ώστε να υποστηρίζουν το σετ εντολών RISC. Για να επιτευχθεί η επιθυμητή αποδοτικότητα και ευελιξία, τα επόμενα σχήματα εισόδων/ εξόδων υποστηρίζονται από το αρχείο καταχωρητών.

1. Μια έξοδος 8-bit και μία είσοδος 8-bit
2. Δύο έξοδοι 8-bit και δύο είσοδοι 8-bit
3. Δύο έξοδοι 8-bit και μία είσοδος 16-bit
4. Μία έξοδος 16-bit και μία είσοδος 16-bit



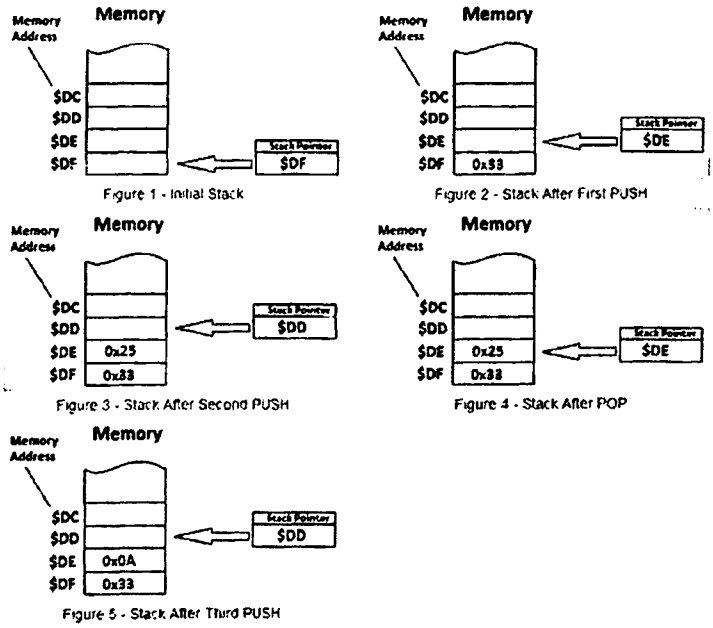
Εικόνα 17: Καταχωρητές Γενικής Χρήσης του AVR

2.3.4 Στοίβα (Stack) και Stack Pointer (SP) των AVR

Η στοίβα [6,8] χρησιμοποιείται για την αποθήκευση προσωρινών στοιχείων, τοπικών μεταβλητών και για την αποθήκευση διευθύνσεων επιστροφής μετά από την χρήση διακοπών και υπορουτίνων. Ο Stack Pointer δείχνει πάντα την κορυφή της στοίβας. Η στοίβα εφαρμόζεται καθώς αυξάνεται από τις υψηλότερες θέσεις μνήμης στις χαμηλότερες. Δηλαδή μία εντολή PUSH μειώνει τον δείκτη στοίβας. Ο Stack Pointer έχει μήκος 2 bytes τα οποία ελέγχονται από δύο 8-bit καταχωρητές. Ένα παράδειγμα της λειτουργίας της στοίβας δίνεται στις εικόνες 18 και 19 όπου τοποθετούμε στους καταχωρητές R16, R17 και R18 τρείς τυχαίες τιμές (hex), 0x33, 0x25,0x0A. Μετά τοποθετούμε τις δύο πρώτες στη στοίβα με την εντολής PUSH, επαναφέρουμε τη μία ξανά στον καταχωρητή R17 και αποθηκεύουμε στη θέση της την τιμή που περιέχει ο καταχωρητής R18.

```
.include "tn2313def.inc"

        ;Set up AVR ATTiny2313 stack
        LDI     R16, RAMEND
        OUT     SPL, R16

        LDI     R16, 0x33
        LDI     R17, 0x25
        LDI     R18, 0x0A

        PUSH    R16
        PUSH    R17
        POP     R17
        PUSH    R18

end:    RJMP    end
```

Εικόνα 18: Λειτουργία Στοίβας



Εικόνα 19 : Γράψιμο και Διάβασμα στη Στοιβα

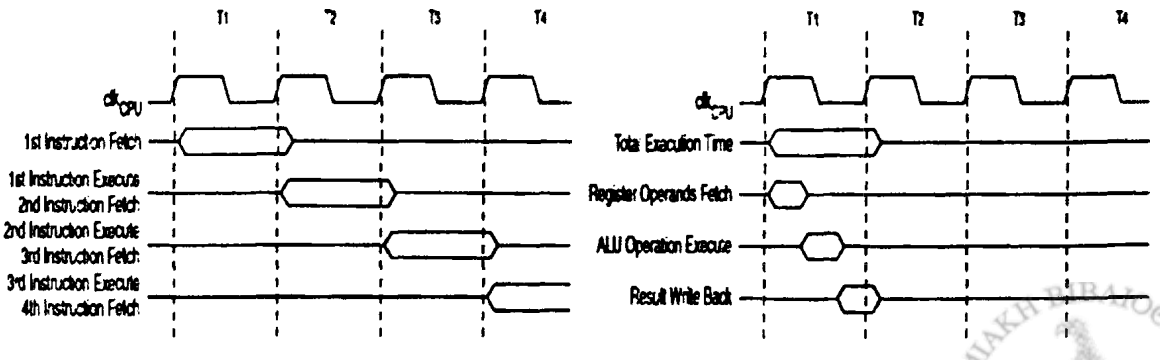
2.3.5 Ο Program Counter (PC) των AVR

Ο πιο σημαντικός καταχωρητής στους AVR είναι ο PC [6,8] ο οποίος χρησιμοποιείται από την CPU για να δείξει την διεύθυνση της επόμενης εντολής που πρόκειται να εκτελεστεί. Καθώς η CPU φέρνει την προηγούμενη εντολή από την μνήμη, ο PC αυξάνεται αυτόματα δείχνοντας την επόμενη εντολή. Όσο πιο μεγάλο το μέγεθος του PC τόσο περισσότερες θέσεις μνήμης μία CPU μπορεί να προσπελάσει.

Αυτό σημαίνει πως ένας PC 14-bit μπορεί να προσπελάσει 16384 τοποθεσίες μνήμης γιατί $2^{14} = 16384$. Ο PC μπορεί να έχει μέγεθος μέχρι και 22-bits. Το μέγεθος αυτό εξαρτάται από το μέγεθος της μνήμης FLASH του κάθε μικροελεγκτή. Αυτό σημαίνει πως ο AVR μπορεί να προσπελάσει θέσεις μνήμης 00h – 3FFFFFh, ένα σύνολο 4.194.304 θέσεων. Επειδή όμως κάθε τοποθεσία FLASH έχει μήκος 2 bytes θεωρητικά ο AVR μπορεί να έχει το πολύ 8,38 Mbytes (2 * 4.194.308) κώδικα. Στην πραγματικότητα όμως κανένας μικροελεγκτής AVR δεν διαθέτει τόσο μεγάλη μνήμη FLASH.

2.3.6 Χρόνοι εκτέλεσης των εντολών των AVR

Η CPU του AVR οδηγείται από ένα ρολόι CLK_{CPU} όπου βρίσκεται εγκατεστημένο μέσα στο ολοκληρωμένο και για το οποίο δεν χρησιμοποιείται διαιρέτης συχνότητας. Στην επόμενη εικόνα φαίνονται οι παράλληλες μεταφορές και εκτελέσεις εντολών (fetch - execution) που ενεργοποιούνται με την αρχιτεκτονική Harvard. Αυτή είναι και η βασική λειτουργία διοχέτευσης έτσι ώστε ο μικροελεγκτής να εκτελεί 1 Mega Instructions Per Second ανά MHz (1 MIPS/MHz). Επίσης φαίνονται η εκτέλεση συνεχόμενων εντολών με χρήση pipeline 2 σταδίων καθώς και τα στάδια εκτέλεσης μιας εντολής από την ALU ανά κύκλο ρολογιού [8].



Μεταφορά των εντολών απο τη μνήμη και εκτέλεση (fetch - execute) Τμηματοποίηση εντολής σε μικρότερα μέρη (μέθοδος Pipeline)

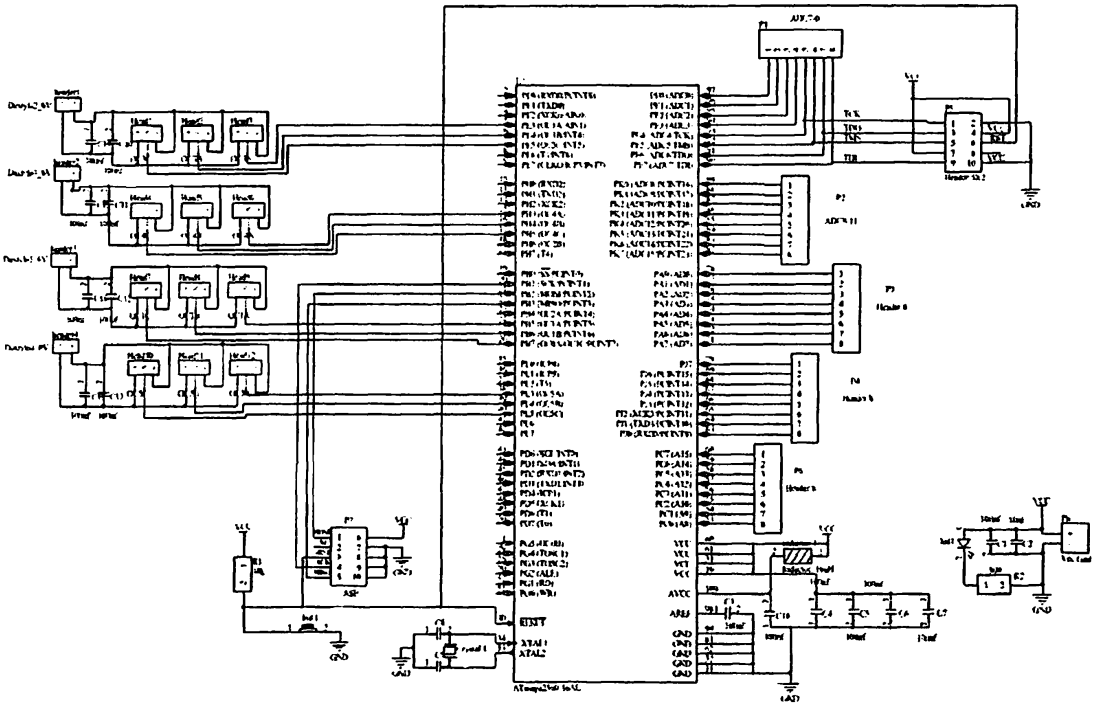
Εικόνα 20

2.4 Σχεδίαση και Κατασκευή Πλακέτας Μικροελεγκτή

Οι ακροδέκτες του μικροελεγκτή που χρησιμοποιήθηκαν είναι:

- Τα Ports F και K για τα κανάλια του ADC
- Τα Ports A, J και C ως ακροδέκτες γενικής χρήσης
- Τα 4 Vcc και Gnd για την ορθή λειτουργία του
- Τα Ports B,E,H και L ως έξοδοι των σημάτων PWM
- Το Port B για τον προγραμματισμό του μικροελεγκτή
- Οι ακροδέκτες του κρυστάλλου 16MHz και Reset

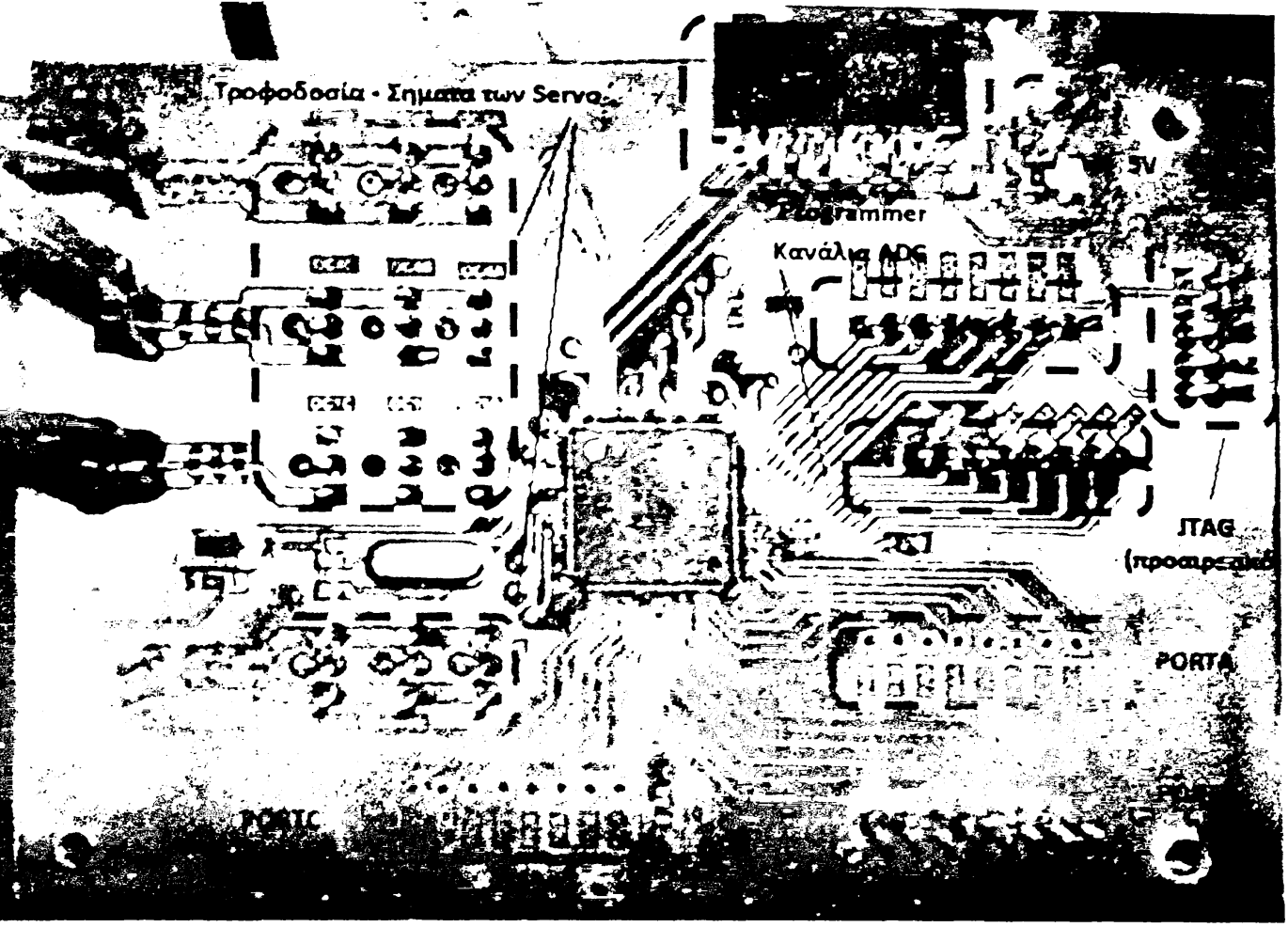
Στην εικόνα 21 φαίνεται το σχηματικό διάγραμμα του μικροελεγκτή.



Εικόνα 21: Συνδεσμολογία πλακέτας μικροελεγκτή

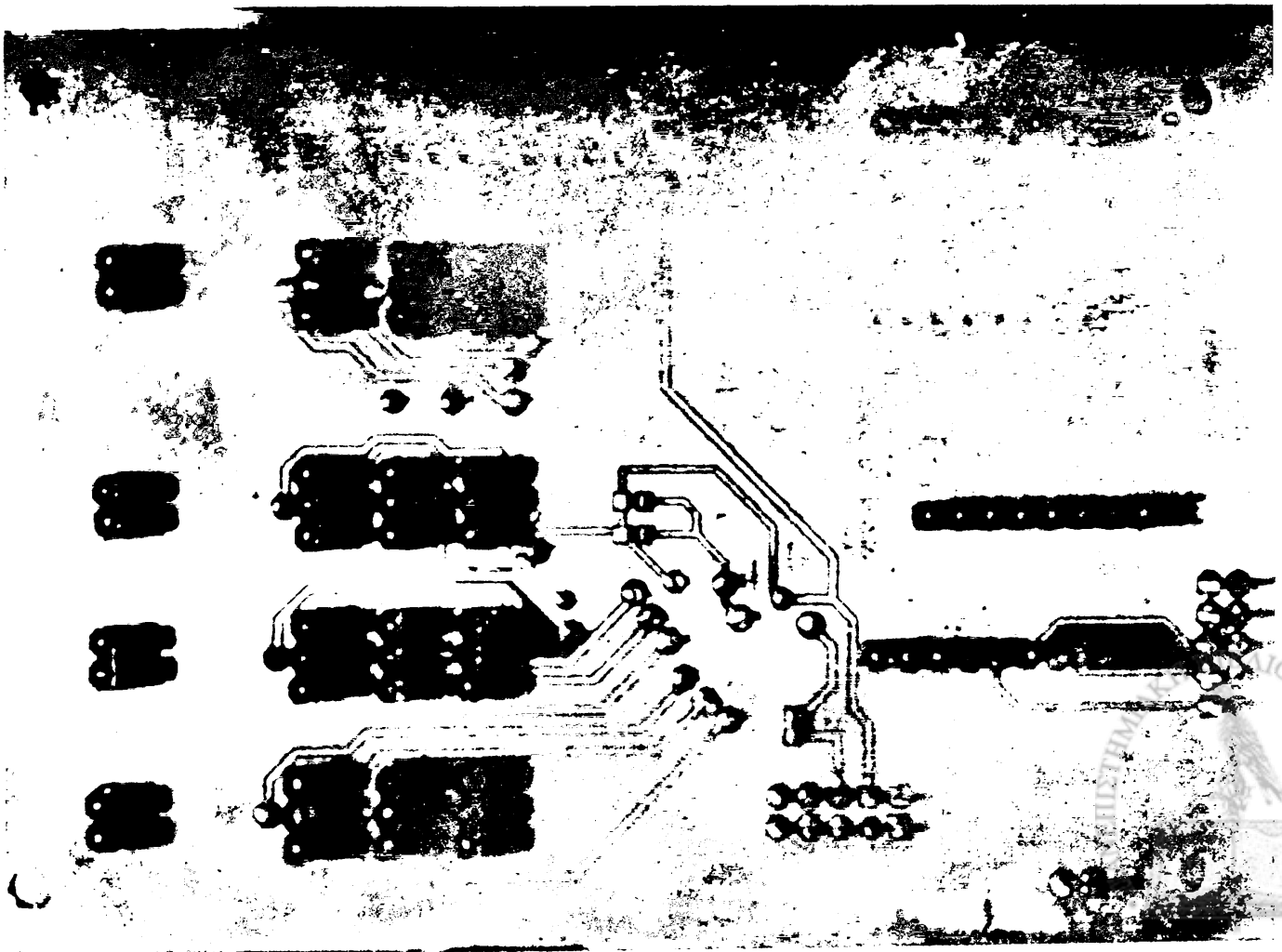
Στην εικόνα 22 φαίνεται η πλακέτα του μικροελεγκτή μετά την κατασκευή της.

Στο αριστερό μέρος της μπροστινής όψης συνδέονται οι ακροδέκτες των σέρβο-κινητήρων της ρομποτικής χείρας. Στο πάνω μέρος είναι οι ακροδέκτες για τον προγραμματισμό του μικροελεγκτή και στα δεξιά του μικροελεγκτή είναι τα κανάλια του ADC, PORT F/K για την λήψη των αναλογικών σημάτων από τους αισθητήρες. Στο κάτω μέρος είναι τα Ports γενικής χρήσης PORT J/A/C. Στα παράρτηματα Π3.3 & Π3.4 παρουσιάζονται τα layouts (σχέδια πλακετών) των μικροελεγκτών.



Εικόνα 22α: Πλακέτα Μικροελεγκτή - Συνδέσεις μπροστινής όψης

ω Όψη



ΚΕΦΑΛΑΙΟ 3^{ο}

Αισθητήρες της Ρομποτικής Χείρας

3.1 Αισθητήρια Κάμψης

Σε αυτό το κεφάλαιο παρουσιάζεται η διάταξη αισθητήρων μέσω της οποίας ο χρήστης θα έχει τον έλεγχο της ρομποτικής χείρας. Επίσης αναλύεται ο τρόπος με τον οποίο σχεδιάστηκε και κατασκευάστηκε η πλακέτα (PCB) των αισθητήρων μέσω της οποίας παράγονται τα επιθυμητά αναλογικά σήματα τα οποία τροφοδοτούν τους μετατροπείς AD των μικροελεγκτών. Πρώτα όμως θα παρουσιαστούν οι δύο τύποι αισθητήρων κάμψης οι οποίοι είναι υπεύθυνοι για την δημιουργία των αναλογικών σημάτων από τις κινήσεις του χεριού του χρήστη, πάνω στο οποίο και είναι τοποθετημένοι .

3.2 Λειτουργία Αισθητήρων

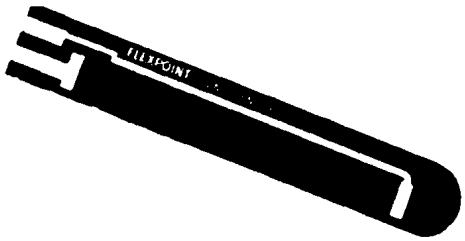
Οι αισθητήρες κάμψης (flex sensors, bend sensors) λειτουργούν ως μεταβλητές αντιστάσεις οι οποίοι μεταβάλλονται ανάλογα με την κάμψη που δέχονται και χωρίζονται σε δύο κατηγορίες:

1. Αγώγιμης ή Ωμικής Μελάνης (Resistive ink)
2. Ωμικού Άνθρακα (Resistive Carbon)

3.2.1 Αισθητήρια Κάμψης Αγώγιμης Μελάνης

Αυτά τα αισθητήρια [9] είναι στοιχεία παθητικής αντίστασης που κατασκευάζονται τοποθετώντας μια λωρίδα αγώγιμης μελάνης επάνω σε μία πλαστική επιφάνεια σχήματος λεπτής λωρίδας με μήκη μεταξύ μίας ίντσας και πέντε ιντσών. Τα αισθητήρια αυτά όταν βρίσκονται σε θέση ηρεμίας (δηλαδή σε έκταση) χαρακτηρίζονται από μία ενδογενή αντίσταση. Καθώς τα αισθητήρια κάμπτονται, η απόσταση μεταξύ των σωματιδίων της μελάνης μεγαλώνει. Ο λόγος είναι ότι τα λιγότερα διπλανά σωματίδια έρχονται σε επαφή αυξάνοντας με αυτό τον τρόπο την αντίσταση.

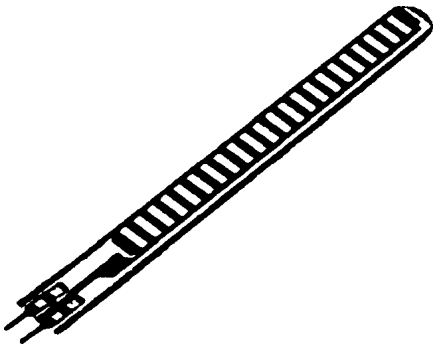
Η τιμή της αντίστασης της μελάνης (εικόνα 23) κυμαίνεται μεταξύ 10ΚΩ και 50ΚΩ σε θέση ηρεμίας (0 μοίρες) ανάλογα με το μήκος του αισθητήρα και αυξάνεται κατά ένα παράγοντα 8 με 10 σε πλήρη κάμψη (100 – 180 μοίρες).



Εικόνα 23: Αισθητήρια Αγώγιμης Μελάνης

3.2.2 Αισθητήρια Κάμψης Ωμικού/Αγώγιμου Άνθρακα

Κάτι ανάλογο συμβαίνει και σε αυτό τον τύπο αισθητήρων [9] με τη διαφορά ότι τώρα το υλικό που χρησιμοποιείται εδώ είναι σωματίδια αγώγιμου άνθρακα τα οποία εκτυπώνονται επάνω σε μία λωρίδα εύκαμπτου πλαστικού (εικόνα 24).



Εικόνα 24: Αισθητήρια Ωμικού Άνθρακα

3.3 Συνδεσμολογία Αισθητήρων

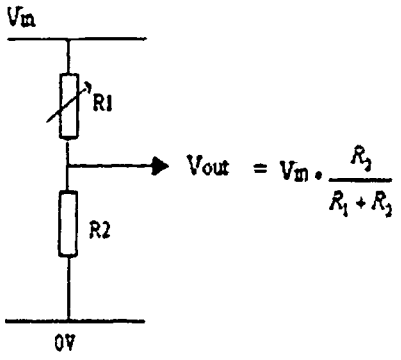
Οι αισθητήρες κάμψης, όπως αναφέρθηκε και πριν, συμπεριφέρονται σαν κοινές μεταβλητές αντιστάσεις. Για να παραχθεί όμως το αναλογικό σήμα, το οποίο θα περάσει ως είσοδος στον μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC) του μικροελεγκτή, θα πρέπει να υλοποιηθεί ένα απλό κύκλωμα διαιρέτη τάσης για τον κάθε αισθητήρα. Ο διαιρέτης τάσης τροφοδοτείται με 5V και το σήμα εξόδου του κυμαίνεται από 0V έως 2.5V περίπου. Στη συνέχεια, αυτά τα σήματα θα περνούν μέσα από διατάξεις τελεστικών ενισχυτών συνδεδεμένων ως ακολουθητές τάσης και από εκεί τα σήματα θα οδηγούνται στον ADC του μικροελεγκτή. Ο ADC θα διαβάζει τιμές στις εισόδους του από 0V έως 2.5V περίπου.

3.3.1 Λήψη αναλογικών σημάτων μέσω Διαιρέτη Τάσης

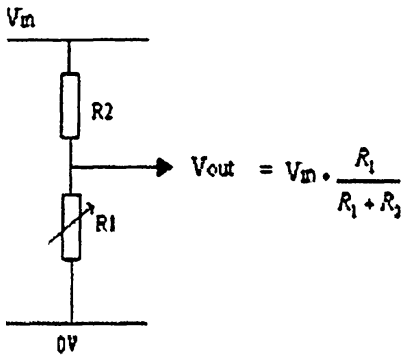
Ο αισθητήρας κάμψης από μόνος τους σαν στοιχείο δεν μπορεί να παράγει αναλογικό σήμα τάσης. Για να γίνει αυτό θα πρέπει ο αισθητήρας να τοποθετηθεί σε μία διάταξη τροφοδοσίας και διαιρέτη τάσης.

Υπάρχουν δύο τρόποι για την σύνδεση του αισθητήρα κάμψης πάνω στον διαιρέτη τάσης. Ο πρώτος τρόπος είναι ο αισθητήρας να τοποθετηθεί στο πάνω μισό του διαιρέτη όπως φαίνεται στο σχήμα 3α. Η τάση τροφοδοσίας του κυκλώματος είναι V_{in} = 5V και η αντίσταση R_1 του παραπάνω κυκλώματος αντιπροσωπεύει τον αισθητήρα κάμψης.

Ο δεύτερος τρόπος είναι να αντιστρέψουμε τη συνδεσμολογία. Δηλαδή ο αισθητήρας να τοποθετηθεί στο κάτω μισό όπως φαίνεται στο Σχήμα 3β.



Σχήμα 3α: 1η Τοπολογία Διαιρέτη τάσης



Σχήμα 3β: 2η Τοπολογία Διαιρέτη τάσης

Τα αισθητήρια αγώγιμης μελάνης μίας ίντσας σε θέση ηρεμίας (επίπεδα) έχουν αντίσταση $\mathbf{R_{flat}}$ η οποία κυμαίνεται μεταξύ 2ΚΩ και 3ΚΩ ενώ τα αισθητήρια ωμικού άνθρακα δύο ιντσών έχουν αντίσταση $\mathbf{R_{flat}}$ η οποία κυμαίνεται μεταξύ 25ΚΩ και 30ΚΩ. Σε κάμψη γωνίας 90°, τα αισθητήρια αγώγιμης μελάνης μίας ίντσας έχουν αντίσταση $\mathbf{R_{bend}}$ η οποία κυμαίνεται μεταξύ 40ΚΩ και 50ΚΩ ενώ τα αισθητήρια ωμικού άνθρακα δύο ιντσών σε κάμψη γωνίας 180° (διπλωμένα) έχουν αντίσταση $\mathbf{R_{bend}}$ η οποία κυμαίνεται μεταξύ 100ΚΩ και 120ΚΩ. Τα αισθητήρια ωμικού άνθρακα θα χρησιμοποιηθούν σε κάμψη 180°. Δηλαδή η θέση ηρεμίας τους θα είναι σε γωνία 180°. Αυτό γίνεται καλύτερα κατανοητό στη συνέχεια όπου αναλύεται η κατασκευή της διάταξης των αισθητήρων (Γάντι). Για το σχήμα 3α, θέτοντας την R_2 ίση με την αντίσταση του αισθητήρα σε επίπεδη θέση (R_{flat}) :

- Για επίπεδη θέση, το σήμα εξόδου παίρνει τη μέγιστη τιμή του $V_{out} = 2.5V$ ($R_1 = R_{flat}$).
- Για κεκλιμένη θέση, το σήμα εξόδου παίρνει την ελάχιστη τιμή του $V_{out} = 0V$ ($R_1 = R_{bend}$).

Για το σχήμα 3β, αλλάζοντας την θέση των R1 και R2 και θέτοντας ξανά την R_2 ίση με την αντίσταση του αισθητήρα σε επίπεδη θέση (R_{flat}) :

- Για επίπεδη θέση , το σήμα εξόδου παίρνει την ελάχιστη τιμή του $V_{out} = 2.5V$ ($R_1 = R_{flat}$).
- Για κεκλιμένη θέση ,το σήμα εξόδου παίρνει την μέγιστη τιμή του $V_{out} = 5V$ ($R_1 = R_{bend}$).

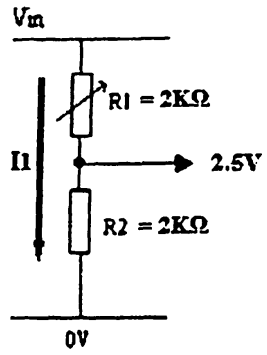
Επιλέχθηκε ο πρώτος τρόπος (Σχήμα 3α). Δηλαδή το αναλογικό σήμα θα μειώνεται καθώς ο αισθητήρας κάμπτεται. Ο λόγοι για τους οποίους επιλέχθηκε αυτός ο τρόπος ήταν δύο.

1) Για απλότητα, καθώς το αισθητήριο που είναι τοποθετημένο στο δάχτυλο κάμπτεται προς τα κάτω, αντίστοιχα και το παραγόμενο αναλογικό σήμα τάσης πέφτει προς τα 0V. Έτσι υπάρχει μία αναλογική σχέση των δύο αυτών μεγεθών που βοηθάει στην κατανόηση των αποτελεσμάτων.
2) Το κάθε παραγόμενο αναλογικό σήμα, αφού μετατραπεί σε ψηφιακό από τον ADC του μικροελεγκτή, θα χρησιμοποιηθεί σε συνάρτηση η οποία θα καθορίζει την κίνηση των κινητήρων των αρθρώσεων των δαχτύλων της ρομποτικής χείρας. Σε αυτή τη συνάρτηση είναι επιθυμητό, το ψηφιακό πλέον σήμα να μειώνεται με την κάμψη των αισθητήρων. Ο λόγος γι'αυτό θα γίνει κατανοητός στο κεφάλαιο 6 όπου αναλύεται η μέθοδος παραγωγής των παλμών οδήγησης των κινητήρων, καθώς οι κινητήρες αυτοί για να κινήσουν τους άξονες τους, χρειάζονται παλμούς μεταβαλλόμενου εύρους.

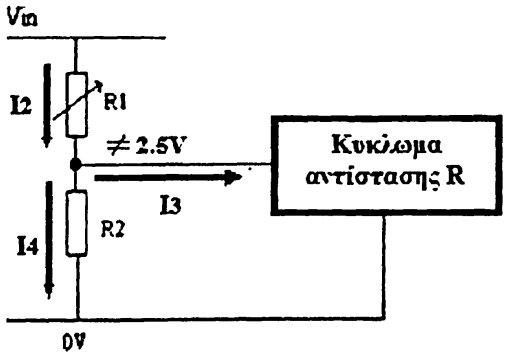
3.3.2 Ακολουθητής Τάσης - Απομονωτής

Τα αναλογικά σήματα των αισθητήρων, πριν μεταφερθούν στις εισόδους του ADC του μικροελεγκτή για επεξεργασία, θα πρέπει να περάσουν πρώτα από κυκλώματα ακολουθητών τάσης. Εάν συνδέσουμε τα σήματα αυτά απευθείας στις εισόδους του ADC, τα σήματα αλλοιώνονται λόγω ύπαρξης φορτίου στις εισόδους

(κανάλια) του ADC. Αυτό εξηγείται στη συνέχεια με τη βοήθεια των σχημάτων 25α και 25β.



Εικόνα 25α : Σωστό Σήμα Εξόδου



Εικόνα 25β: Λανθασμένο Σήμα Εξόδου

Στην εικόνα 25α, φαίνεται ο διαιρέτης τάσης που χρησιμοποιήθηκε πριν για την παραγωγή του αναλογικού σήματος με V_{in} = 5V και R_1 και R_2 έστω 2KΩ χωρίς να έχει συνδεθεί κάπου αλλού. Το κύκλωμα διαρρέεται από ρεύμα I_1 και παράγει ένα αναλογικό σήμα 2.5V . Μέχρι εδώ όλα λειτουργούν έτσι όπως πρέπει.

Στην εικόνα 25β όμως, αν συνδέουμε ένα κύκλωμα αντίστασης R στο σημείο όπου παράγεται το αναλογικό σήμα. Το σήμα πλέον δεν θα έχει την τιμή 2.5V αλλά κάτι άλλο. Αυτό συμβαίνει γιατί, καθώς το κύκλωμα αντίστασης R συνδέεται στο κύκλωμα, το διαπερνάει υποχρεωτικά ένα ρεύμα I_3. Αυτό σημαίνει ότι το ρεύμα $I_2 = I_3+I_4$, δηλαδή $I_2 > I_4$, και έτσι η πτώση τάσης επάνω στην αντίσταση R_1 είναι μεγαλύτερη από την πτώση τάσης πάνω στην αντίσταση R_2 και αυτό έχει σαν αποτέλεσμα το αναλογικό σήμα εξόδου να είναι μικρότερο από 2.5V. Έτσι το σήμα εξόδου του διαιρέτη τάσης έχει αλλοιωθεί.

Το κάθε κανάλι του ADC έχει αντίσταση εισόδου 40KΩ. Έτσι, για την εικόνα 3β, αν τοποθετήσουμε στη θέση του κυκλώματος αντίστασης R, την αντίσταση 40KΩ των καναλιών του ADC (όπου είναι και το ζητούμενο), και με $R_1 = R_2 = 2KΩ$, το αναλογικό σήμα το οποίο περνάει στον ADC είναι 2.44V και όχι 2.5V όπως θα έπρεπε. Αυτό προκύπτει ως εξής:

$R_2 // R \rightarrow R_{parallel} = \frac{2K\Omega \times 40K\Omega}{2K\Omega + 40K\Omega} = 1.904\ K\Omega$

$R_{total} = R_1 + R_{parallel} =$ 3.904KΩ (Συνολική αντίσταση διαιρέτη τάσης)

$V_{cc}/R_{total} = I_{total} = 5V/3.904K\Omega = 1.28mA$ (Συνολικό ρεύμα κυκλώματος)

$V_{parallel} = I_{total} \times R_{parallel} = 1.28mA \times 1.904K\Omega \cong 2.44V$ (Πτώση τάσης πάνω στην $R_{parallel}$)

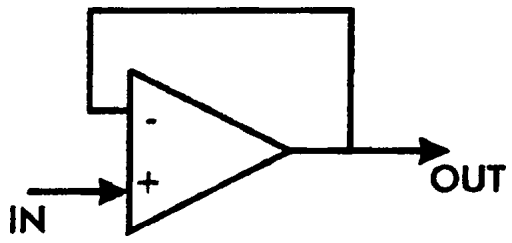
$V_{parallel} = V_{out} \cong 2.44V$ (< 2.5V).

Το ρεύμα που καταναλώνεται πάνω στο φορτίο (αντίσταση R) είναι:

$$I_{Load} = \frac{V_{parallel}}{R} = \frac{2.44V}{40K\Omega} = 61\mu A$$

Έτσι παρατηρείται πως ο διαιρέτης τάσης τροφοδοτεί το κύκλωμα αντίστασης R με ρεύμα $I_{Load} = 61\mu A$, αλλοιώνοντας έτσι το σήμα εξόδου V_{out}.

Σε αυτό το σημείο είναι που χρειαζόμαστε τον απομονωτή. Όπως το λέει και η λέξη, το κύκλωμα αυτό απομονώνει το κύκλωμα εισόδου από το κύκλωμα εξόδου του. Το ρεύμα που διαρρέει το κάθε κανάλι του ADC παρέχεται από την τροφοδοσία του απομονωτή και όχι από το κύκλωμα εισόδου του που είναι ο διαιρέτης τάσης. Έτσι το αναλογικό σήμα που μας ενδιαφέρει μένει ανεπηρέαστο. Οι απομονωτές είναι κυκλώματα τελεστικών ενισχυτών (Τ.Ε.) με μεγάλη αντίσταση εισόδου (θεωρητικά άπειρη), έτσι δεν καταναλώνουν ρεύμα από το κύκλωμα εισόδου (διαιρέτες τάσης), και πολύ μικρή αντίσταση εξόδου (θεωρητικά μηδενική), τροφοδοτώντας έτσι με όσο ρεύμα χρειάζεται τα κυκλώματα στις εξόδους τους (κανάλια ADC). Επίσης, οι απομονωτές εμφανίζουν στην έξοδό τους σήμα ίδιο με αυτό που δέχονται στην είσοδό τους (εικόνα 26α). Δηλαδή το κέρδος τους είναι μονάδα.



Εικόνα 26α: Τελ. Ενισχυτής (Τ.Ε.) συνδεδεμένος ώς Απομονωτής

Αυτό αποδεικνύεται ως εξής:

Γνωρίζουμε από την θεωρία των τελεστικών ενισχυτών (Τ.Ε.) πως

$$V_{out} = A \times (V_+ - V_-),$$

εφόσον η είσοδος (V_-) ενώνεται με την έξοδο (V_{out}) του Τ.Ε. έχουμε

$$V_- = V_{out}$$

Αναδιαμορφώνοντας τον πρώτο τύπο παίρνουμε

$V_{out}/A = (V_+ - V_{out})$, επειδή όμως $A = \infty$

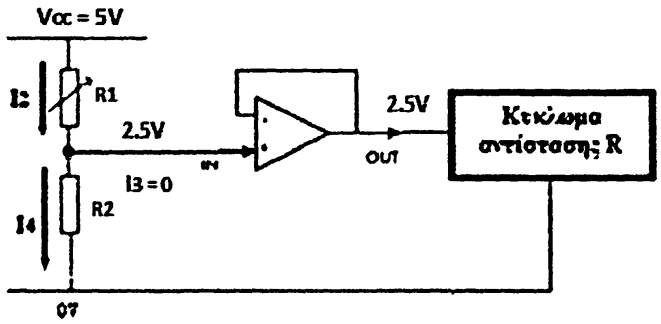
$0 = (V_+ - V_{out})$, και άρα

$V_{out} = V_+$, όμως

$V_+ = V_{in}$, οπότε

$V_{out} = V_{in}$ και $A = \frac{V_{out}}{V_{in}} = 1$ *(Μοναδιαίο κέρδος)*

Τοποθετώντας τον απομονωτή ανάμεσα σε διαιρέτη τάσης και το κανάλι του ADC έχουμε το κύκλωμα της εικόνας 26β.



Εικόνα 26β: Κύκλωμα λήψης αναλογικού σήματος

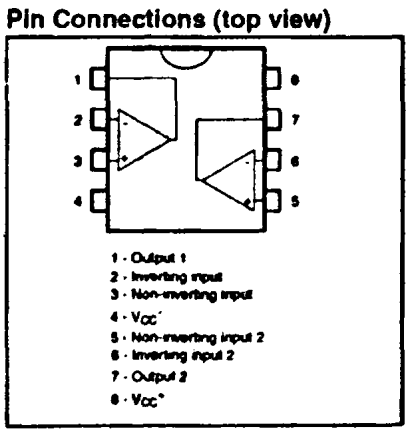
Όπως αναφέρθηκε πριν, ο Τ.Ε. έχει άπειρη αντίσταση εισόδου που σημαίνει ότι το $I_3 = 0$ (εικόνα 26β) και έτσι το σήμα εξόδου του διαιρέτη τάσης είναι 2.5V εφόσον δεν περνάει ρεύμα προς την είσοδο του Τ.Ε. Αντίθετα η αντίσταση εξόδου του Τ.Ε. είναι πολύ μικρή και έτσι το κύκλωμα αντίστασης R (κανάλι ADC) τροφοδοτείται από την τροφοδοσία του Τ.Ε. αφήνοντας το σήμα του διαιρέτη τάσης αναλλοίωτο. Επίσης, εφόσον το κέρδος είναι 1 στον απομονωτή όπως αποδείχθηκε πιο πάνω, το σήμα εξόδου είναι ίδιο με το σήμα εισόδου.

3.4 Σχεδίαση και Κατασκευή της πλακέτας διάταξης αισθητήρων.

Οι αισθητήρες, οι οποίοι και θα τοποθετηθούν επάνω στη διάταξη αισθητήρων (γάντι), αποτελούνται από δύο ακροδέκτες (εικόνα 23 και 24). Αυτοί οι ακροδέκτες οδηγούνται σε μία πλακέτα η οποία περιέχει τα κυκλώματα όλων των τελεστικών ενισχυτών αλλά και όλων τους διαιρετών τάσης της δίαταξης αισθητήρων.

3.4.1 Ανάπτυξη πλακέτας

Για την υλοποίηση της πλακέτας χρησιμοποιήθηκαν είκοσι τελεστικοί ενισχυτές LM358AD [10]. Το κάθε ολοκληρωμένο περιέχει δύο ανεξάρτητους τελεστικούς ενισχυτές υψηλού κέρδους οι οποίοι είναι ειδικά σχεδιασμένοι για να τροφοδοτούνται από μονοπολική πηγή τάσης (Vcc^+ – Gnd) και λειτουργούν σε ένα μεγάλο εύρος τάσεων. Το σημαντικότερο χαρακτηριστικό αυτών των τελεστικών ενισχυτών είναι ότι στη γραμμική λειτουργία, η οποία και θα χρησιμοποιηθεί, το εύρος τάσεων στις εισόδους του μπορεί να περιέχει και την τιμή 0V και η τάση εξόδου του τελεστικού ενισχυτή μπορεί να φτάνει και αυτή μέχρι τα 0V έχοντας βάλει μονοπολική τάση τροφοδοσίας (V_{cc+} – Gnd). Αυτό το χαρακτηριστικό είναι πολύ σημαντικό γιατί στους διαιρέτες τάσης που θα χρησιμοποιηθούν τα αναλογικά σήματα εξόδων τους πρέπει να κυμαίνονται μεταξύ 2.5V και 0V. Στις εικόνες 27α και 27β φαίνεται το απλοποιημένο σχηματικό με τις λειτουργίες των ακροδεκτών του LM358AD και το μέγεθος του ολοκληρωμένου.

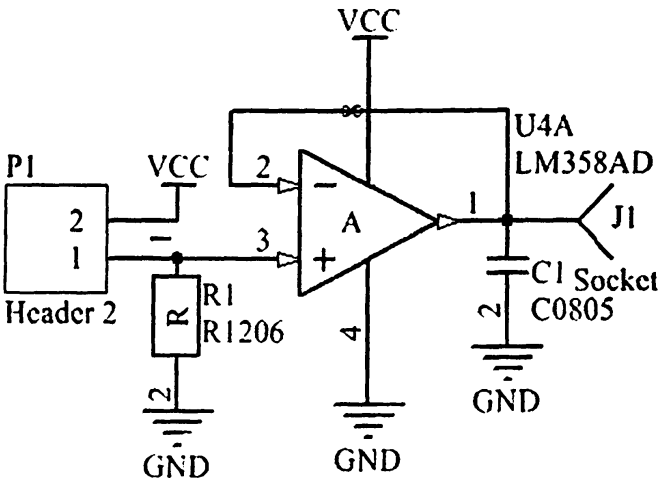


Εικόνα27α: Ακροδέκτες Τ.Ε.

Εικόνα 27β: Μέγεθος I.C.

Συνολικά τοποθετήθηκαν στην πλακέτα δέκα τέτοια ολοκληρωμένα κυκλώματα, δηλαδή 20 τελεστικοί ενισχυτές. Οι 3 από τους 20 τοποθετήθηκαν για μελλοντική επέκταση – αύξηση των αισθητήρων.

Στην εικόνα 28 παρουσιάζεται το σχηματικό διάγραμμα με την συνδεσμολογία του διαιρέτη τάσης και ενός τελεστικού ενισχυτή. Στον διπλό ακροδέκτη P συνδέονται οι ακροδέκτες των αισθητηρίων κάμψης. Ο τελεστικός ενισχυτής είναι συνδεδεμένος ως απομονωτής (όπως περιγράφηκε πιο πριν). Στην αναστρέφουσα είσοδο του τοποθετείται το αναλογικό σήμα του διαιρέτη τάσης και στην έξοδο του είναι συνδεδεμένος ένας ακροδέκτης (J1). Από τον ακροδέκτη αυτό το αναλλοίωτο αναλογικό σήμα μεταφέρεται στη συνέχεια στην είσοδο ενός από τα κανάλια του ADC, του μικροελεγκτή. Συνολικά η πλακέτα της διάταξης αισθητήρων (Γάντι) που σχεδιάσθηκε και κατασκευάσθηκε περιέχει 20 τέτοια κυκλώματα ένα για κάθε τελεστικό ενισχυτή.



Εικόνα 28: Συνδεσμολογία Τ.Ε. της πλακέτας παραγωγής σημάτων

Το κύκλωμα είναι παρόμοιο για όλους τους αισθητήρες με τη σημαντική διαφορά ότι σε κάθε τέτοιο κύκλωμα η αντίσταση R του διαιρέτη τάσης που πηγαίνει στη γείωση θα είναι διαφορετική κάθε φορά καθώς οι αισθητήρες δεν έχουν ακριβώς την ίδια αντίσταση ούτε είναι όλοι τους τοποθετημένοι με τον ίδιο τρόπο επάνω στο γάντι το οποίο θα χρησιμοποιεί ο χρήστης για τον έλεγχο της ρομποτικής χείρας. Για να γίνει αυτό καλύτερα κατανοητό, στον επόμενο πίνακα (πίνακας 1) δίνονται οι τιμές των αντιστάσεων των αισθητήρων σε θέση ηρεμίας πριν και μετά την τοποθέτηση τους πάνω στην διάταξη του γαντιού.

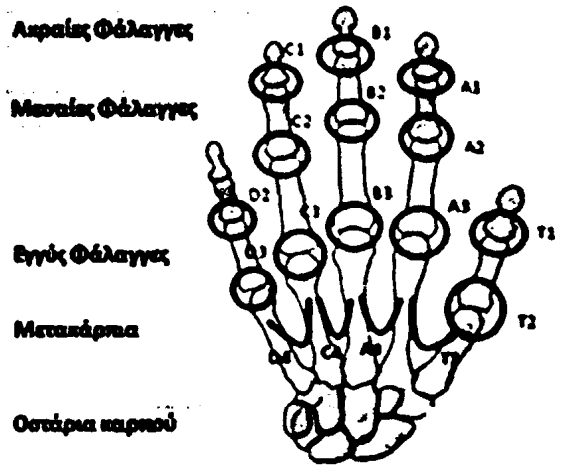
	A1	A2	A3	A4	B1	B2	B3	C1	C2	C3	C4
Πριν ΚΩ	2.1	2.15	2.2	30	2.3	2.2	2.3	1.7	2	2.2	25
Μετά ΚΩ	2	3.3	3.5	100	2.3	2.5	3.3	1.9	2.5	3.2	90

Πίνακας 1α

	D1 D2	D3	D4	T1	T2	T3
Πριν ΚΩ	1.9	2.2	34	1.6	1.8	28
Μετά ΚΩ	2.3	3	101	1.25	2.7	120

Πίνακας 1β

Στην εικόνα 29 παρουσιάζονται τα σημεία του χεριού του χρήστη πάνω στα οποία θα τοποθετηθούν τα αισθητήρια κάμψης. Επίσης δίνονται και οι ονομασίες των αισθητήρων κάμψης του κάθε δαχτύλου.

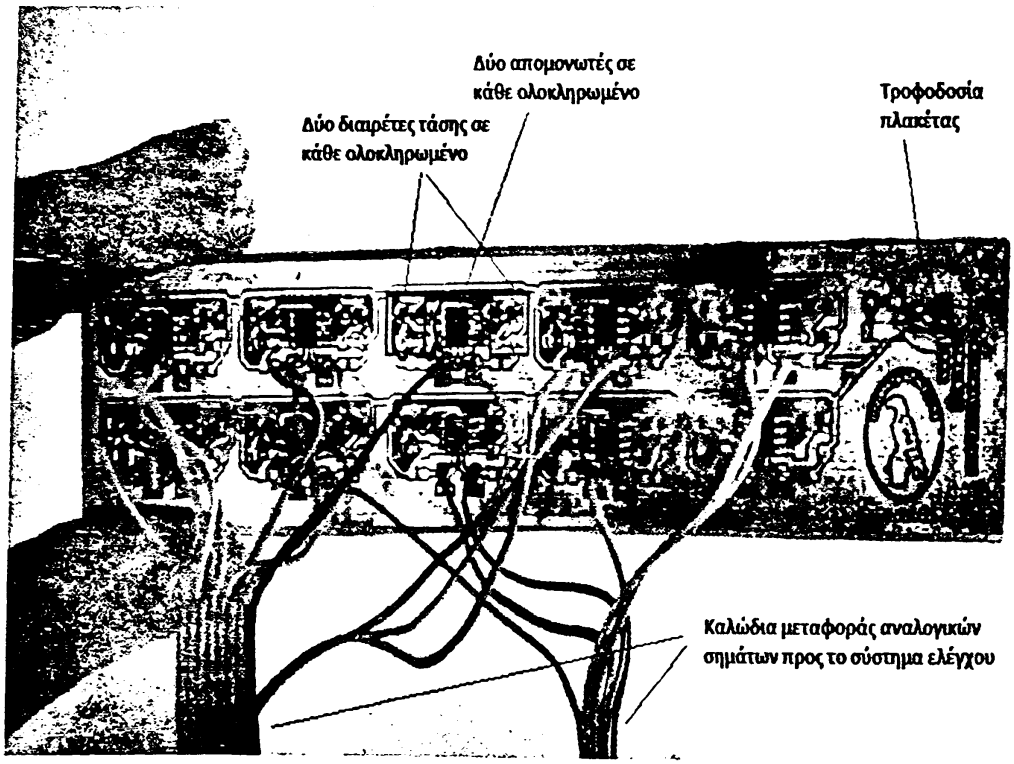


Εικόνα 29: Θέση αισθητήρων πάνω στο ανθρώπινο χέρι

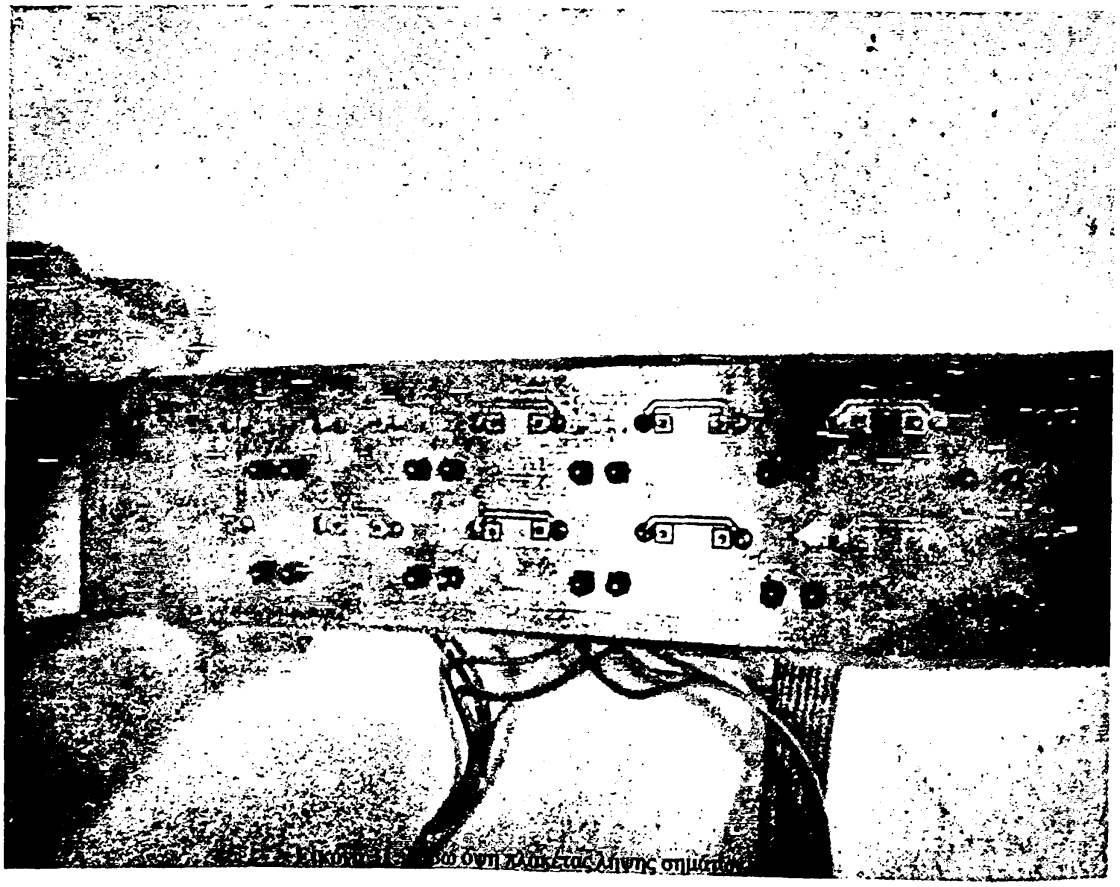
Οι κύκλοι αντιστοιχούν στα αισθητήρια αγώγιμης μελάνης μίας ίντσας και οι κυρτές γραμμές αντιστοιχούν στα αισθητήρια ωμικού άνθρακα δύο ιντσών οι οποίοι θα τοποθετηθούν στο γάντι δευτέρου επιπέδου και το οποίο θα αναλυθεί παρακάτω.

- Τα A1,A2,A3,A4 αντιστοιχούν στα αισθητήρια του δείκτη
- Τα B1,B2,B3 αντιστοιχούν στα αισθητήρια του μέσου δαχτύλου
- Τα C1,C2,C3,C4 αντιστοιχούν παράμεσου δαχτύλου
- Τα D2,D3,D4 αντιστοιχούν στα αισθητήρια του μικρού δαχτύλου
- Τα T1,T2,T3 αντιστοιχούν στα αισθητήρια του αντίχειρα
- Τα A4, C4, D4, T3 ελέγχουν την πλάγια κίνηση όλων των δαχτύλων

Στις εικονες 30 και 31 φαινεται η πλακετα της διαταξης αισθητηρων μετά την κατασκευή της. Στο παράρτημα Π3.2 δίνεται το layout (σχέδιο της πλακέτας) το οποίο χρησιμοποιήθηκε για την αποτύπωση των κυκλωμάτων πάνω σε αυτή.



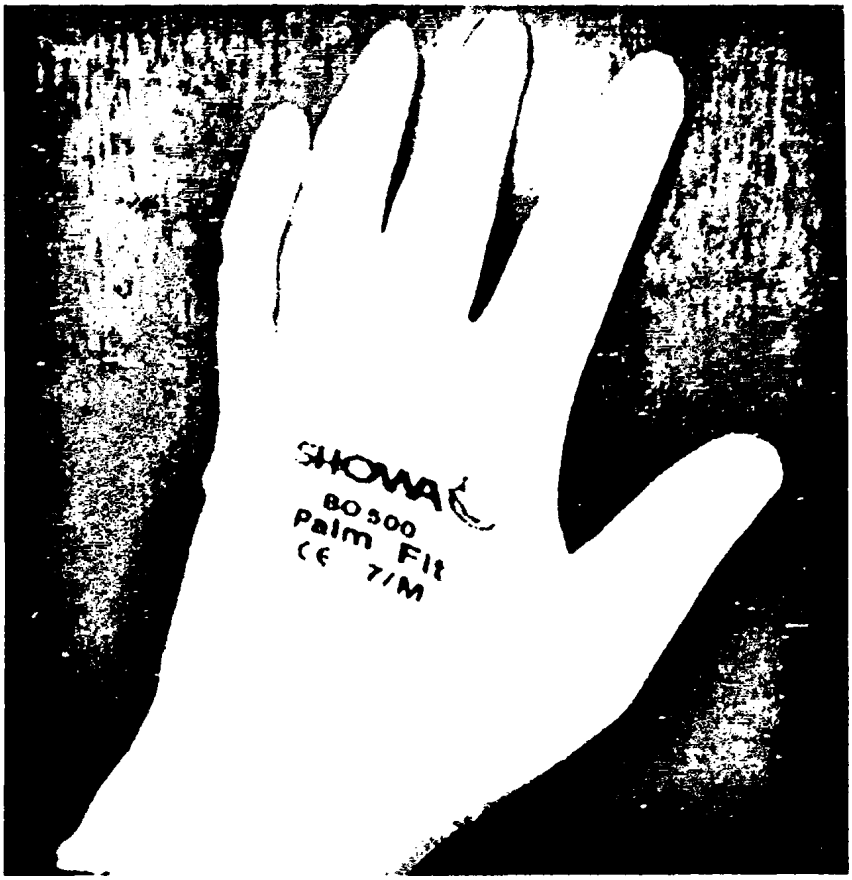
Εικόνα 30: Πλακέτα λήψης σημάτων



Εικόνα 31: Εμπρόσθια όψη πλακέτας λήψης σημάτων

3.5 Διάταξη Αισθητήρων

Οι δεκαοκτώ αισθητήρες (αγώγιμης μελάνης και άνθρακα) που αναλύθηκαν παραπάνω, καθώς δεν ήταν δυνατόν να κολληθούν πάνω στο ανθρώπινο χέρι τοποθετήθηκαν επάνω σε ένα κοινό γάντι εργασίας όπως αυτό που εικονίζεται παρακάτω (εικόνα 32).



Εικόνα 32: Γάντι του εμπόδου

Τοποθετήθηκε ένας αισθητήρας κάμψης αγώγιμης μελάνης σε κάθε μία από τις αρθρώσεις των πέντε δαχτύλων του χεριού. Αυτό θα δίνει την δυνατότητα ανεξάρτητων κινήσεων όχι μόνο των δαχτύλων της ρομποτικής χείρας μεταξύ τους αλλά και των αρθρώσεων του κάθε δαχτύλου ξεχωριστά.

Στην εικόνα 31α φαίνεται η πίσω όψη της διάταξης των αισθητήρων πρώτου επιπέδου μετά την ολοκλήρωση της και στην εικόνα 33β φαίνεται η μπροστινή όψη αυτής. Το αισθητήριο στην εικόνα 33β ελέγχει την άρθρωση μεταξύ μετακαρπίου οστού του αντίχειρα και της πρώτης φάλαγγας του αντίχειρα (στην εικόνα 27 αντιστοιχεί στον κύκλο Τ2).

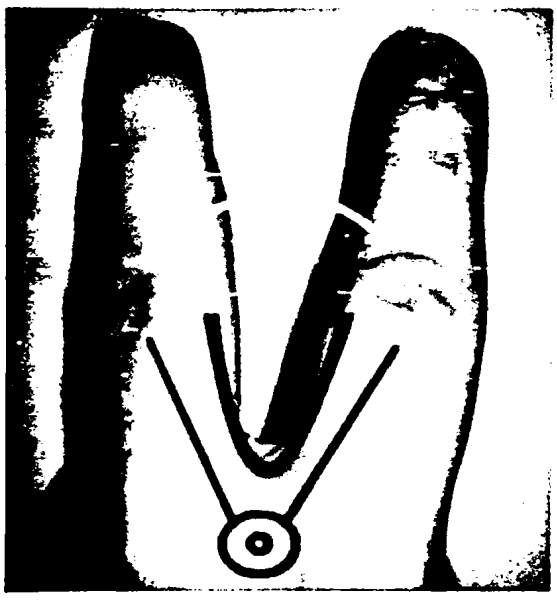
Εικόνα 33α: Τοποθέτηση αισθητήρων στο γαντι

Εικόνα 33β: Αισθητήρας μετάκάρπιου του αντίχειρα

Καθώς οι αρθρώσεις των δαχτύλων του χρήστη θα κάμπτονται τα αντίστοιχα αισθητήρια θα κάμπτονται και αυτά αναλόγως. Στην εικόνα 34 αυτό γίνεται καλύτερα κατανοητό.

Εικόνα 34: Κάμψη Αισθητήρων αγώγιμης μελάνης

Σε αυτό το σημείο θα αναλυθεί η διάταξη αισθητήρων 2^{ου} επιπέδου η οποία αφορά τους αισθητήρες άνθρακα δύο ιντσών . Οι αισθητήρες αυτοί θα καταγράφουν το σήμα που θα προκύπτει από το άνοιγμα και το κλείσιμο μεταξύ δύο διαδοχικών δαχτύλων του χρήστη ο οποίος θα φοράει και το γάντι. Στην εικόνα 35 αυτό γίνεται καλύτερα αντιληπτό.

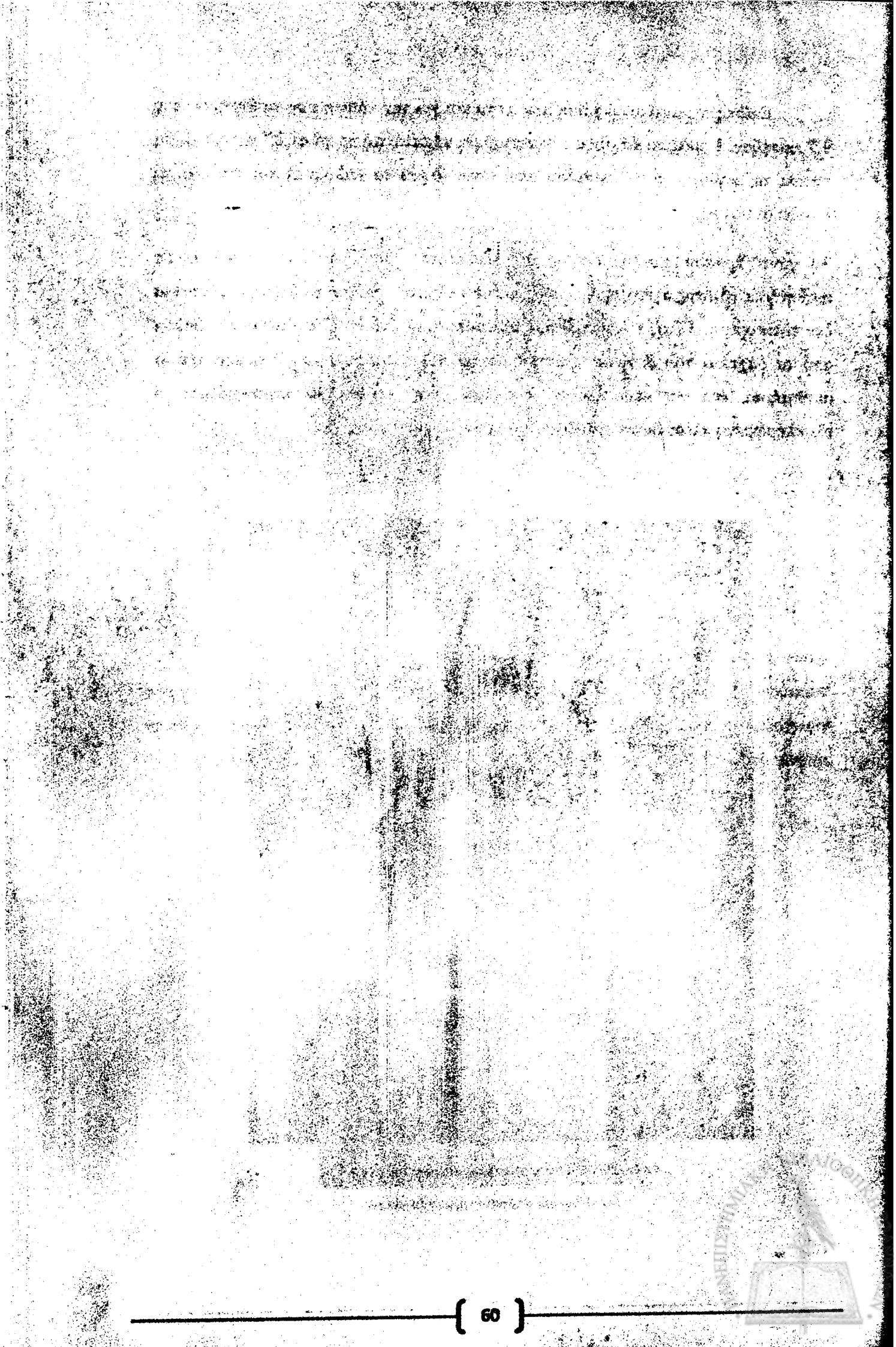


Εικόνα 35: Κάμψη αισθητήρων Ωμικού Άνθρακα

Επίσης το γάντι αυτό χρησιμοποιείται και για την κάμψη των αισθητήρων του $1^{ου}$ επιπέδου. Καθώς τα δάχτυλα του χεριού θα κάμπτονται το γάντι $2^{ου}$ επιπέδου θα πατάει τα αισθητήρια $1^{ου}$ επιπέδου από πάνω (αφού τα καλύπτει) και έτσι θα τα λυγίζει αντίστοιχα.

Το γάντι πρώτου επιπέδου εισάγεται μέσα στο γάντι δευτέρου επιπέδου. Τα αισθητήρια κάμψης αγώγιμης μελάνης πλέον καλύπτονται από το δεύτερο γάντι και δεν είναι ορατά. Τέλος τοποθετούνται τα αισθητήρια άνθρακα δύο ιντσών ανάμεσα από τα δάχτυλα του δεύτερου γαντιού. Συνολικά τοποθετήθηκαν τέσσερα τέτοια αισθητήρια, ένα για κάθε ζεύγος δαχτύλων. Στην εικόνα 36 παρουσιάζεται η ολοκληρωμένη πλέον διάταξη των αισθητήρων.

Εικόνα 36: Γάντι 1ου επιπέδου μέσα στο γάντι 2ου επιπέδου



ΚΕΦΑΛΑΙΟ 4^{ο}

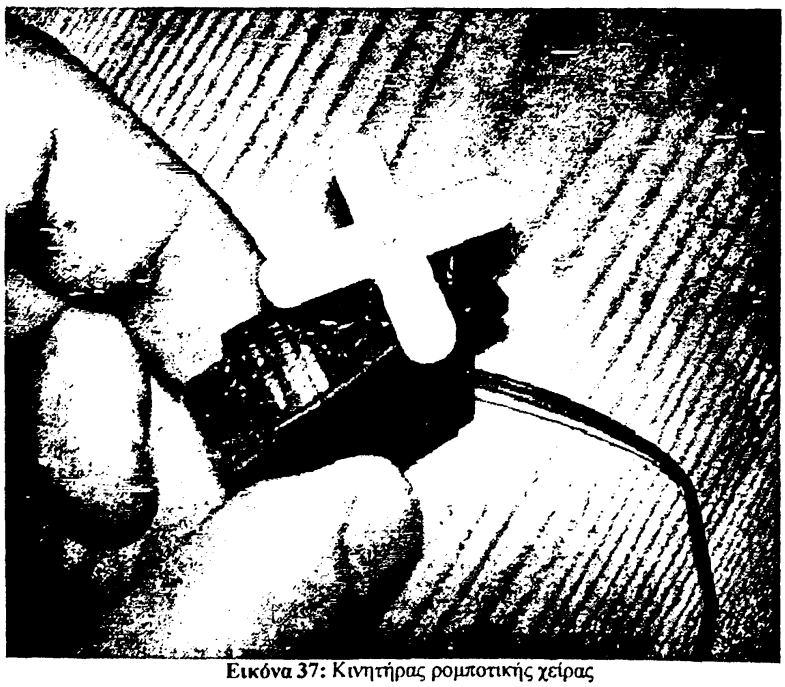
Κινητήρες της Ρομποτικής Χείρας

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται αναφορά στους κινητήρες που χρησιμοποιήθηκαν για τις κινήσεις των δαχτύλων της ρομποτικής χείρας.

Ο λόγος που επιλέχθηκαν αυτοί οι συγκεκριμένοι κινητήρες είναι γιατί:

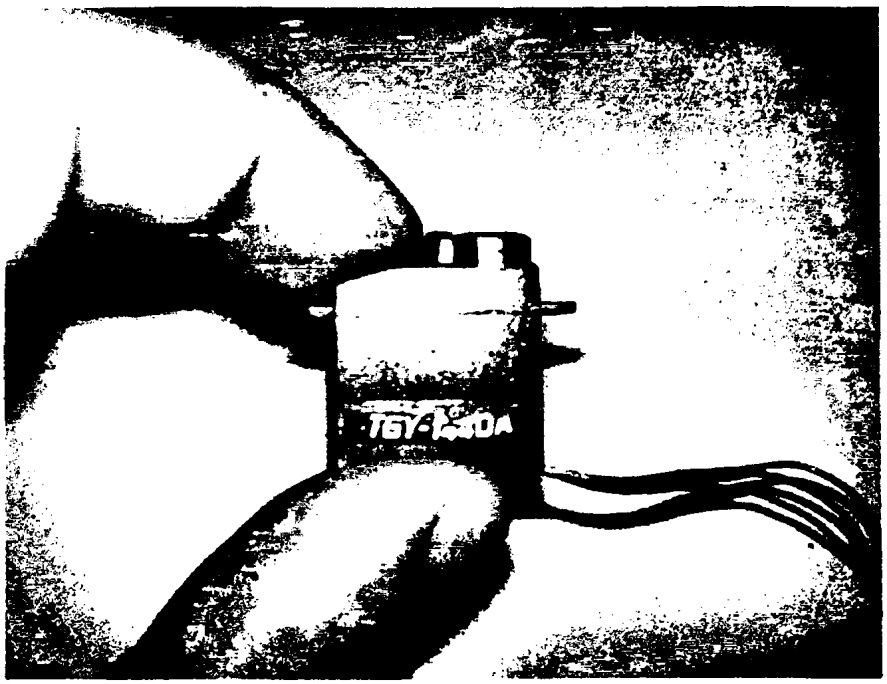
- μπορούν να κάνουν κινήσεις ακριβείας χιλιοστού
- Ελέγχονται μέσω μικροελεγκτή με χρήση σημάτων PWM
- έχουν το κύκλωμα ανάδρασης τοποθετημένο επάνω στον κινητήρα
- έχουν σχετικά καλή ροπή για το μέγεθος τους (2.2kg*cm)
- είναι σχετικά φθηνοί (6-8 euro ο κάθε κινητήρας)
- όλα αυτά είναι τοποθετημένα μέσα σε ένα μικρό κουτί που αποτελεί και το περίβλημα του σέρβο-κινητήρα.
- μαζί με κάθε σέρβο-κινητήρα παρέχονται και τα αντίστοιχα άκρα πρόσδεσης τα οποία τοποθετούνται επάνω στον άξονα (εικόνα 37)



Εικόνα 37: Κινητήρας ρομποτικής χείρας

4.2 Σέρβο-κινητήρας (Servo Motor)

Ο σέρβο-κινητήρας είναι ένας απλός DC κινητήρας που μέσω μίας διάταξης ελέγχου είναι σε θέση να εκτελέσει κινήσεις ακριβείας. Στην εικόνα 38 φαίνεται ένας από τους σέρβο-κινητήρες που χρησιμοποιήθηκαν για την κατασκευή της ρομποτικής χείρας.

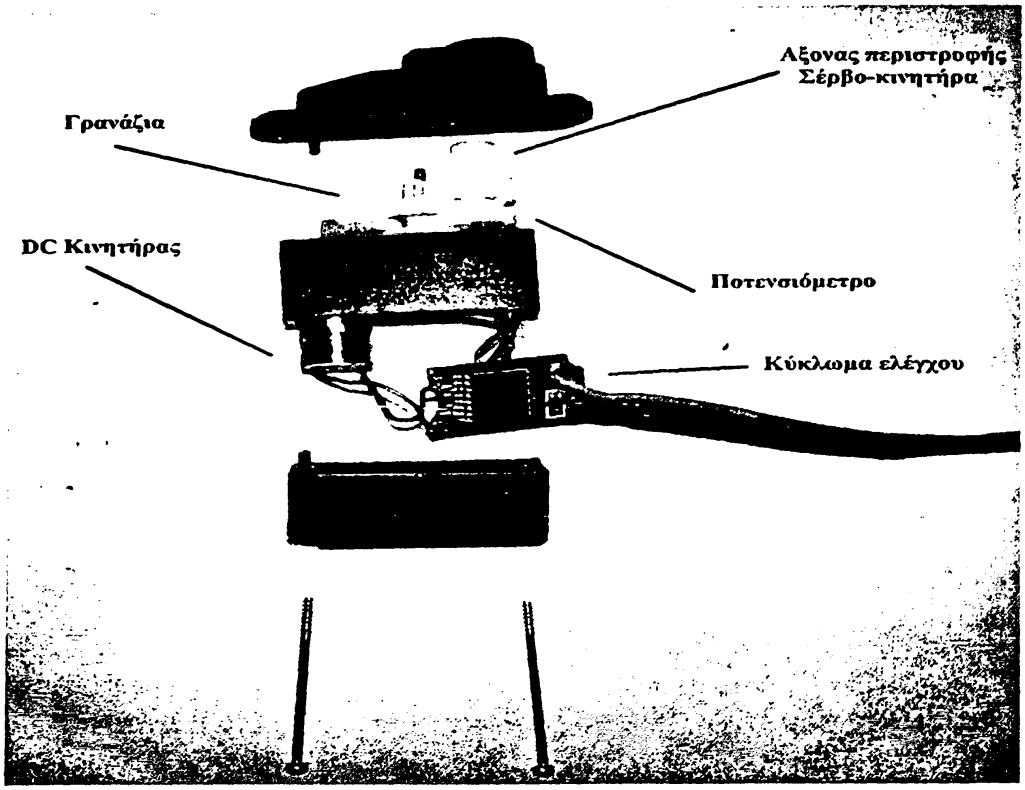


Εικόνα 38: 2ο είδος κινητήρων ρομποτικής χείρας

4.2.1 Λειτουργία Σέρβο-Κινητήρα

Εφαρμόζοντας μία πηγή τάσης στα άκρα ενός DC κινητήρα ο άξονας του (ρότορας) περιστρέφεται συνεχώς καθώς δεν υπάρχει κάποιο σύστημα ελέγχου να τον σταματάει. Αν σε αυτό τον κινητήρα προστεθεί και ένας μηχανισμός ο οποίος είναι σε θέση να παρατηρεί και να ελέγχει την κίνηση του περιστρεφόμενου ρότορα, τότε έχουμε ένα σέρβο-κινητήρα [11].

Στην εικόνα 39 φαίνονται τα μέρη από τα οποία αποτελείται ο σέρβο-κινητήρας.



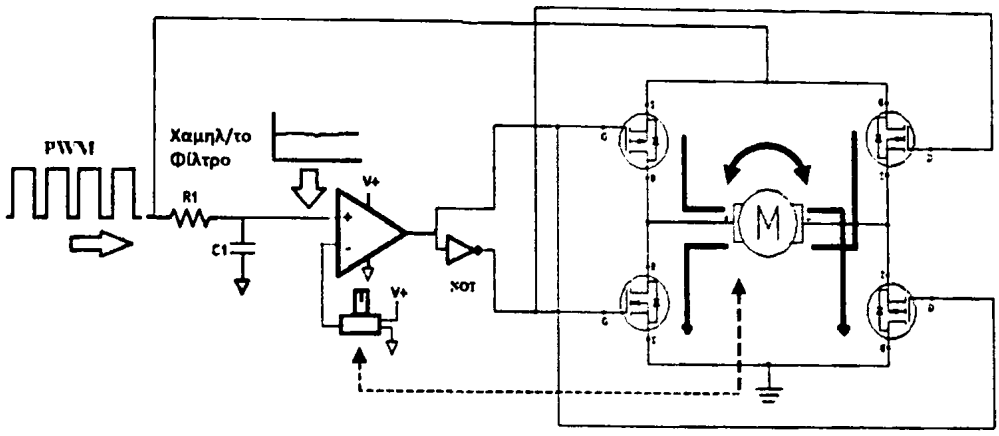
Εικόνα 39: Εσωτερικό του Servo

Ο σέρβο-κινητήρας είναι μία μικρή συσκευή η οποία αποτελείται από ένα DC κινητήρα, ένα μηχανισμό οδοντωτών τροχών (γρανάζια) και ένα ηλεκτρονικό κύκλωμα ελέγχου.

Ο κινητήρας είναι ένας απλός DC κινητήρας όπως αυτοί που χρησιμοποιούνται σε κασετόφωνα, dvd players κλπ. Αυτός ο κινητήρας συνδέεται μηχανικά μέσω του περιστρεφόμενου άξονά του σε μία διάταξη από οδοντωτούς τροχούς (κοινώς γρανάζια) τα οποία λειτουργούν ως μειωτήρες, δηλαδή υποβιβάζουν την ταχύτητα περιστροφής του κινητήρα και ταυτόχρονα αυξάνουν τη ροπή του.

Το κύκλωμα ελέγχου στην πιο απλή του μορφή αποτελείται από ένα τελεστικό ενισχυτή, ένα χαμηλοπερατό φίλτρο, ένα ποτενσιόμετρο και μια H-Bridge όπως φαίνεται στην εικόνα 40.

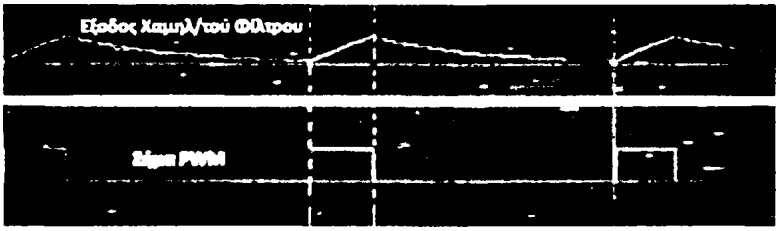
Σκοπός του παρακάτω κυκλώματος είναι να αναδείξει τη βασική λειτουργία των σέρβο-κινητήρων καθώς ο κάθε κατασκευαστής έχει και διαφορετικό κύκλωμα ελέγχου στον πυρήνα του.



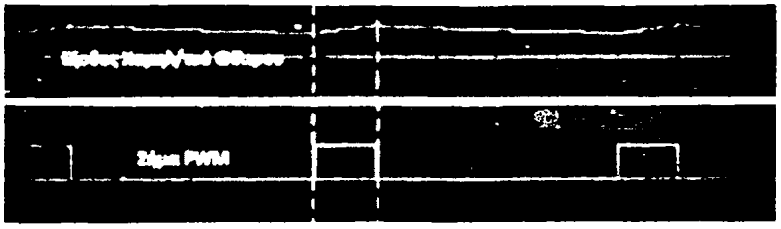
Εικόνα 40: Κύκλωμα Ελέγχου servo

Ο τελεστικός ενισχυτής είναι ο «εγκέφαλος» του συστήματος. Κάθε χρονική στιγμή συγκρίνει το σήμα ανάδρασης του ως προς το σήμα ελέγχου (η σήμα αναφοράς).

Το σήμα ελέγχου είναι ένα σήμα αναλογικής τάσης το οποίο προκύπτει μετά από την επεξεργασία του σήματος PWM μέσω ενός χαμηλοπερατού φίλτρου όπως φαίνεται στην εικόνα 40. Όταν το σήμα PWM γίνεται λογικό "1", ο πυκνωτής φορτίζεται ενώ, όταν το σήμα PWM γίνεται λογικο"0" ο πυκνωτής αποφορτίζεται μέσω της αντίστασης R_1. Έτσι στην έξοδο του χαμηλ/τού φίλτρου παράγεται ένα σήμα αναλογικής τάσης. Αυτό είναι και το σήμα ελέγχου στην μη αναστρέφουσα είσοδο του συγκριτή. Στην εικόνα 41 αυτό γίνεται καλύτερα κατανοητό.



Εικόνα 41α Μετατροπή PWM σε αναλογική τάση



Εικόνα 41β Εξομάλυνση αναλογικής τάσης

Όπως γίνεται φανερό το αναλογικό σήμα που παράγεται από το χαμηλοπερατό φίλτρο στην εικόνα 41α έχει μία αρκετά μεγάλη κυμάτωση. Αυτό το φαινόμενο μπορεί να εξαλειφθεί (εικόνα 41β) τοποθετώντας μεγαλύτερες τιμές στα στοιχεία του φίλτρου. Στην εικόνα 41α η τιμή της αντίστασης είναι 10KΩ και του πυκνωτή 3uF ενώ στην εικόνα 41β οι τιμές είναι 30KΩ και 4.7uF αντίστοιχα.

Το σήμα ανάδρασης του συγκριτή, είναι μία αναλογική τιμή τάσης η οποία προκύπτει από ένα ποτενσιόμετρο το οποίο είναι τοποθετημένο κάτω από τον άξονα περιστροφής του σέρβο-κινητήρα και που εικονίζεται με διακεκομμένη γραμμή στην εικόνα 40. Καθώς περιστρέφεται ο άξονας, περιστρέφεται ανάλογα και το ποτενσιόμετρο, αλλάζοντας έτσι την τιμή του σήματος ανάδρασης.

Εάν το σήμα ανάδρασης είναι μικρότερο από το σήμα ελέγχου τότε η έξοδος του συγκριτή παράγει θετική τάση (V_+) και ο άξονας του DC κινητήρα μέσω της H-Bridge στρέφεται προς τη μία κατεύθυνση μέχρι τα δύο σήματα να εξισωθούν. Μόλις οι τιμές τάσεων των εισόδων εξισωθούν, ο συγκριτής παράγει 0V στην έξοδο του γυρνώντας τον άξονα του DC κινητήρα αντίθετα. Αυτό έχει σαν συνέπεια να περιστραφεί αντίθετα και το ποτενσιόμετρο, δίνοντας έτσι ξανά μικρότερη τιμή τάσης στην αναστρέφουσα είσοδο του συγκριτή. Έτσι επαναλαμβάνεται η ίδια διαδικασία κρατώντας τον άξονα του σέρβο- κινητήρα σε μία θέση.

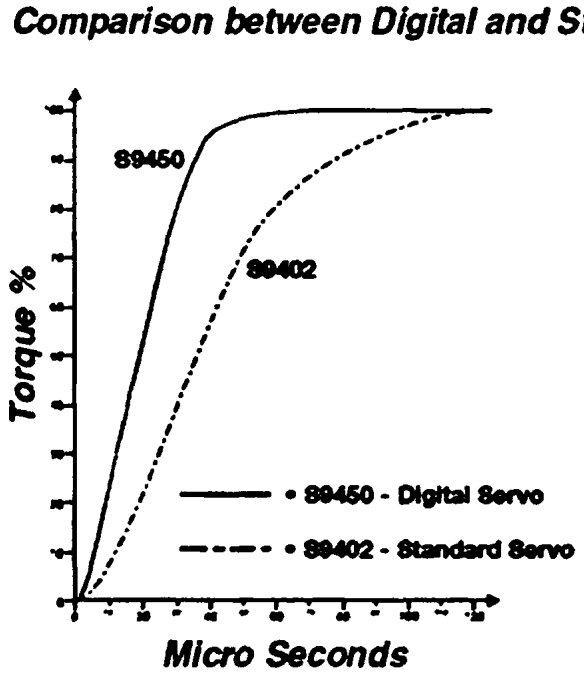
Εάν το σήμα ανάδρασης είναι μεγαλύτερο από το σήμα ελέγχου τότε η έξοδος του συγκριτή παράγει τάση ίση με 0V. Αυτό έχει ως αποτέλεσμα την περιστροφή του κινητήρα και άρα του ποτενσιόμετρου προς την αντίθετη κατεύθυνση, όπου η διαδικασία επαναλαμβάνεται όπως και πριν. Η θέση του άξονα του κινητήρα ελέγχεται από την μη αναστρέφουσα είσοδο του συγκριτή. Έτσι γίνεται ο έλεγχος και με αυτό τον τρόπο ο σέρβο-κινητήρας μπορεί να κάνει κινήσεις ακριβείας καθώς μία μικρή διαφορά των εισόδων του συγκριτή οδηγεί σε μικρή περιστροφή του άξονα και του ποτενσιόμετρου.

4.3 Τύποι σέρβο- κινητήρων που χρησιμοποιήθηκαν για την κατασκευή της ρομποτικής χείρας.

Για την υλοποίηση της ρομποτικής χείρας χρησιμοποιήθηκαν οι ψηφιακοί, Mini και Micro Servos [12][13]. Αυτά τα δύο είδη δεν έχουν καμία απολύτως διαφορά ως προς τη λειτουργία τους ούτε διαφέρουν στον τρόπο με τον οποίο ελέγχονται από τον χρήστη. Και στα δύο είδη εφαρμόζεται το ίδιο σήμα ελέγχου PWM συχνότητας 50Hz. Η διαφορά τους είναι στο μέγεθος τους καθώς οι Micro σέρβο-κινητήρες είναι μικρότεροι από τους Mini σέρβο-κινητήρες. Οι λόγοι για τους οποίους επιλέχθηκαν οι ψηφιακοί σέρβο-κινητήρες έναντι των αντίστοιχων αναλογικών περιγράφονται στην επόμενη σελίδα.

Ο κινητήρας του αναλογικού servo δέχεται ένα PWM σήμα ελέγχου το οποίο ανανεώνεται 50 φορές το δευτερόλεπτο. Από την άλλη πλευρά ο κινητήρας του ψηφιακού servo δέχεται το ίδιο σήμα PWM αλλά μέσω επεξεργασίας από τον μικροελεγκτή του συστήματος ελέγχου ανανεώνεται μέχρι και 400 φορές το δευτερόλεπτο (400Hz). Ανανεώνοντας τη θέση του κινητήρα οκτώ φορές πιο γρήγορα από τον αναλογικό, ο ψηφιακός servo μπορεί να δώσει τη μέγιστη ροπή από την αρχή της κίνησης έχοντας έτσι μικρότερο Dead-Band (Το Dead-Band είναι ο χρόνος ανάμεσα στη στιγμή που ο άξονας του κινητήρα θα αρχίσει να γυρνάει μέχρι τη στιγμή που θα αποκτήσει την μέγιστη ροπή του). Αυτό μπορεί να τεκμηριωθεί πρακτικά τοποθετώντας αρχικά σε δύο κινητήρες (αναλογικό και ψηφιακό) το ίδιο σήμα PWM. Αλλάζοντας την τιμή του σήματος αυτού, παρατηρείται πως ο ψηφιακός σέρβο-κινητήρας αντιδρά πιο γρήγορα από τον αναλογικό σε απότομη αλλαγή του σήματος PWM.

Στην εικόνα 42 φαίνεται ενδεικτικά το γράφημα της λειτουργίας δύο σέρβο-κινητήρων ίδιας ροπής αλλά διαφορετικού τύπου (ο ένας είναι αναλογικός S9402, και ο άλλος ψηφιακός S9450 σύμφωνα με τα στοιχεία που δίνει ο κατασκευαστής).



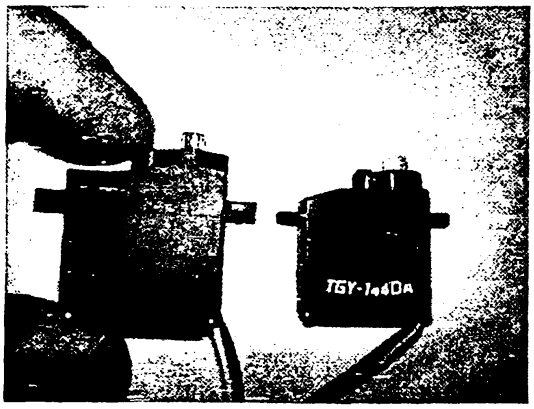
Εικόνα 42: Απόδοση Ψηφιακών και Αναλογικών servo

Όπως γίνεται φανερό η χρήση ψηφιακών σέρβο-κινητήρων για τον έλεγχο των φαλάγγων των δαχτύλων της ρομποτικής χείρας ήταν μονόδρομος καθώς ο κάθε ένας από αυτούς τους κινητήρες θα έπρεπε να είναι σε θέση να αντιδρά όσο το δυνατόν πιο

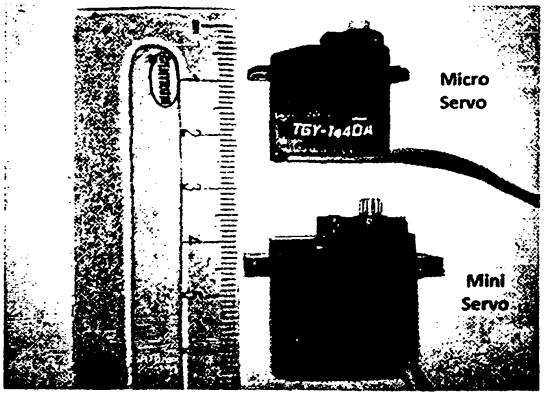
γρήγορα στις αλλαγές του σήματος PWM και ακόμη, να είναι σε θέση να σφίξει και να κρατήσει αντικείμενα όταν χρειαστεί.

Προϋπόθεση για την κατασκευή της ρομποτικής χείρας ήταν οι κινητήρες να είναι όσο το δυνατό μικρότεροι έτσι ώστε η τελική κατασκευή να προσομοιώνει όσο το δυνατόν περισσότερο τη μορφή και τις διαστάσεις του ανθρώπινου χεριού. Επιπλέον οι κινητήρες θα έπρεπε να έχουν όσο το δυνατόν μεγαλύτερη ροπή. Αυτά τα δύο μεγέθη, όγκος και ροπή, κινούνταν σχεδόν ανάλογα καθώς οι κινητήρες με μικρό όγκο είχαν και μικρή ροπή και το αντίθετο.

Μετά από έρευνα ως προς τα χαρακτηριστικά των σέρβο-κινητήρων από διάφορους κατασκευαστές χρησιμοποιήθηκαν αυτοί της εταιρίας Turnigy οι οποίοι βρέθηκαν να είναι και οι πιο κατάλληλοι για την κατασκευή της ρομποτικής χείρας αφού διέθεταν σχετικά καλή ροπή για το μέγεθος τους. Στις εικόνες 43α και 43β φαίνονται οι δύο τύποι σέρβο-κινητήρων που χρησιμοποιήθηκαν.



Εικόνα 43α: Σύγκριση μεγέθους

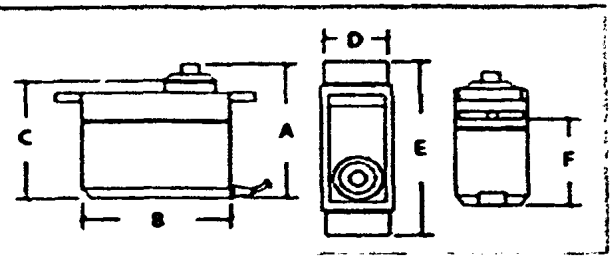


Εικόνα 43β

Τα servos που επιλέχθηκαν για τις αρθρώσεις των ακραίων φαλάγγων είναι τα TGY-1440A και ανήκουν στην κατηγορία των Micro Servos λόγω των πολύ μικρών

διαστάσεων τους. Στον πίνακα 2 παρουσιάζονται οι διαστάσεις τους, η ροπή η γωνιακή ταχύτητα και το βάρος τους.

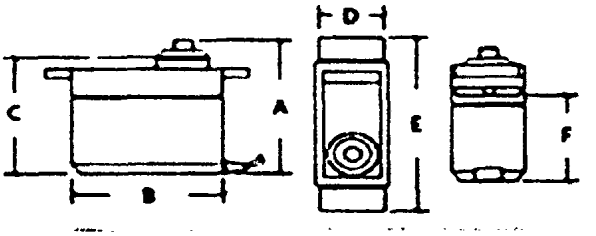
Weight (g)	4.4
Torque (kg*cm)	0.8 - 1
Speed (Sec/60deg)	0.1
A(mm)	25
B(mm)	21
C(mm)	23
D(mm)	9
E(mm)	28
F(mm)	15



Πίνακας 2: Χαρακτηριστικά Micro Servo TGY-1440A

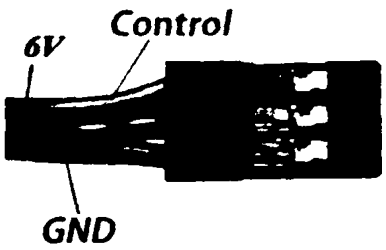
Τα servos που επιλέχθηκαν για τις μεσαίες και τις εγγύς φάλαγγες είναι τα TSS-10MG και ανήκουν στην κατηγορία των Mini Servos λόγω των σχετικά μικρών διαστάσεων τους. Στον πίνακα 3 παρουσιάζονται οι διαστάσεις τους, η ροπή η γωνιακή ταχύτητα και το βάρος τους.

Weight (g)	10
Torque (kg*cm)	2.2 - 2.5
Speed (Sec/60deg)	0.12
A(mm)	28
B(mm)	23
C(mm)	25
D(mm)	12
E(mm)	32
F(mm)	16



Πίνακας 3: Χαρακτηριστικά Mini Servo TSS-10MG

Το σημαντικότερο στοιχείο των δύο κινητήρων είναι η ροπή. Ο Micro servo μπορεί να παράγει μέχρι 1 kg*cm και ο Mini servo μπορεί να παράγει μέχρι 2.6 kg*cm σε τάση λειτουργίας 6V. Η τάση λειτουργίας μπορεί να κυμανθεί από 4.5V μέχρι 6V. Επειδή όμως η τάση έχει άμεσο αντίκτυπο στη ροπή των κινητήρων επιλέχθηκε η μεγαλύτερη τάση λειτουργίας των 6V για μέγιστη ροπή. Από τους κινητήρες όπως φαίνεται και στις προηγούμενες εικόνες προεξέχουν τρία καλώδια (εικόνα 44). Τα δύο είναι για την τροφοδοσία του σέρβο-κινητήρα και το τρίτο για το σήμα ελέγχου. Από αυτό το καλώδιο θα περνάει το σήμα PWM (που αναφέρθηκε στην εικόνα 40) στον κινητήρα για την οδήγηση του.



Εικόνα 44: Ακροδέκτες ελέγχου των servo

ΚΕΦΑΛΑΙΟ 5^{o}

Τροφοδοσία Συστήματος

5.1. Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια αναλυτική περιγραφή της τροφοδοσίας του συστήματος. Χρησιμοποιήθηκε ένα κλασσικό τροφοδοτικό επιτραπέζιου Η/Υ το οποίο τροποποιήθηκε κατάλληλα για τις ανάγκες του συστήματος. Επίσης, περιγράφεται αναλυτικά ο τρόπος επιλογής των switching regulators οι οποίοι υποβιβάζουν την τάση στις επιθυμητές τιμές για την λειτουργία του συστήματος.

5.2 Τροφοδοτικό Υ/Η

Για την υλοποίηση της τροφοδοσίας του συστήματος αρχικά πρέπει να ληφθεί υπόψη α) το συνολικό ρεύμα με το οποίο θα τροφοδοτείται το σύστημα και β) η τάση λειτουργίας του.

Ο λόγος είναι πως ένας ψηφιακός Σέρβο-Κινητήρας (Σ-Κ) χρειάζεται 200mA για να λειτουργήσει σωστά δίνοντας τη μέγιστη ροπή στον **ελεύθερο** άξονα του, ενώ ένας αναλογικός Σ-Κ χρειάζεται 40mA για τον ίδιο σκοπό. Το ρεύμα το οποίο τραβάει ένας ψηφιακός Σ-Κ σε κατάσταση " **stall** " ανέρχεται στα 800mA και το ρεύμα που τραβάει ένας αναλογικός Σ-Κ στην ίδια " **stall** " ανέρχεται στα 500mA.

Για την ασφαλέστερη λειτουργία του συστήματος θέτουμε ως μέγιστες τιμές ρευμάτων για τους ψηφιακούς Σ-Κ : I_{max}= 1 Ampere και για τους αναλογικούς Σ-Κ: I_{max} = 600mA.

Το κάθε δάχτυλο του χεριού αποτελείται από τέσσερεις Σ-Κ, τρείς ψηφιακούς και ένα αναλογικό. Παίρνοντας το I_{max} για κάθε Σ-Κ υπολογίζουμε το μέγιστο ρεύμα για ένα δάχτυλο :I_{finger}= 800mA + 800mA +800mA + 600mA → $\mathbf{I_{finger}= 2.8A \approx 3A}$.

Έτσι λοιπόν το συνολικό ρεύμα που απαιτεί το σύστημα μόνο για τη λειτουργία των Σ-Κ είναι $\mathbf{I_{ΟΛ}= 15\ Amp}$.

Γι'αυτό το λόγο χρησιμοποιήθηκε ένα τροφοδοτικό σταθερού υπολογιστή, ισχύος 550 watt το οποίο μπορεί να δώσει μέχρι **18A σε τάση 12V** και **25A σε τάση 5V** και το οποίο ικανοποιεί πλήρως τις απαιτήσεις των Σ-Κ του συστήματος σε ρεύμα.

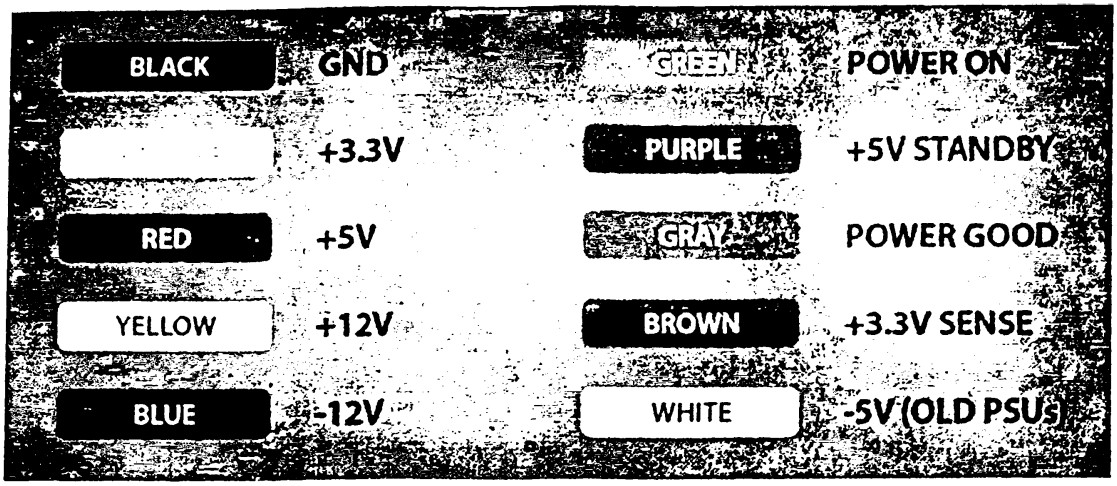
Η τάση στην οποία μπορεί να λειτουργήσει ένας Σ-Κ κυμαίνεται αυστηρά μεταξύ 4.8V και 6V. Αυξάνοντας την τάση σε ένα Σ-Κ μεγαλώνει και η ροπή στον άξονα του καθώς και η γωνιακή του ταχύτητα.

Ενδεικτικά αναφέρεται πως για τον ψηφιακό Σ-Κ **Turnigy TSS-10MG** με τάση λειτουργίας 4.8V η ροπή του είναι T = 2.2 Kg*cm και η γωνιακή του ταχύτητα είναι ω= 8.73rad/sec . Ενώ με τάση λειτουργίας 6V η ροπή του (Torque) ανέρχεται σε T= 2.6Kg*cm και η γωνιακή του ταχύτητα σε ω= 10.47rad/sec. Έτσι, εφόσον αυτό που μας ενδιαφέρει κυρίως είναι η ροπή των Σ-Κ ώστε οι αρθρώσεις των δαχτύλων του ρομποτικού χεριού να κινούνται πιο εύκολα και «τραβώντας» λιγότερο ρεύμα , χρησιμοποιούμε την μεγαλύτερη τάση που μπορούμε δηλαδή τα 6 volt.

Όπως αναφέρθηκε πριν μία από τις τάσεις εξόδου του τροφοδοτικού είναι τα 12V. Χρησιμοποιώντας διατάξεις όπως τα switching regulators μπορούμε να μειώσουμε την τάση εισόδου (12V) στα 6V η οποία είναι και η επιθυμητή τάση για να λειτουργήσουν οι Σ-Κ.

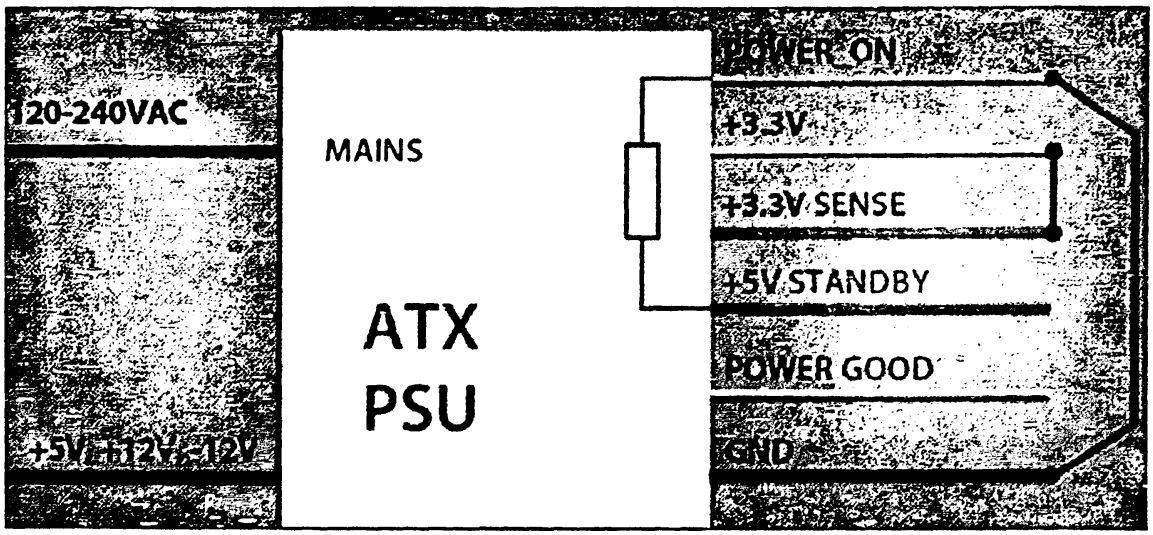
Έτσι λοιπόν έχουμε όλα τα απαραίτητα στοιχεία που χρειαζόμαστε για να προχωρήσουμε στην κατασκευή της τροφοδοσίας του συστήματος.

Αρχικά έπρεπε να γίνουν κάποιες τροποποιήσεις στο τροφοδοτικό του Υ/Η ώστε να είναι σε θέση να λειτουργήσει. Στον πίνακα 5 παρουσιάζονται όλοι οι χρωματισμοί των καλωδίων του τροφοδοτικού με την αντίστοιχη λειτουργία τους.



Πίνακας 5: Καλώδια που περιέχονται σε ένα τροφοδοτικό PC

Η τάση που μας ενδιαφέρει είναι τα 12V. Τα σήματα ελέγχου είναι τέσσερα : **Πράσινο, Μωβ, Γκρι και Καφέ**. Πρώτο και σημαντικότερο βήμα είναι να συνδέσουμε το Πράσινο καλώδιο (Power On) στο Μαύρο Καλώδιο (Ground) απευθείας η μέσω ενός διακόπτη δύο θέσεων, διαφορετικά το τροφοδοτικό δεν θα λειτουργήσει. Ο λόγος γι'αυτό είναι ότι το Πράσινο καλώδιο είναι συνδεδεμένο μέσω μίας αντίστασης Pull-Up (20kΩ-50kΩ) στο Μωβ καλώδιο όπως φαίνεται και στο παρακάτω σχήμα:



Εικόνα 45: Τροποποίηση σημάτων

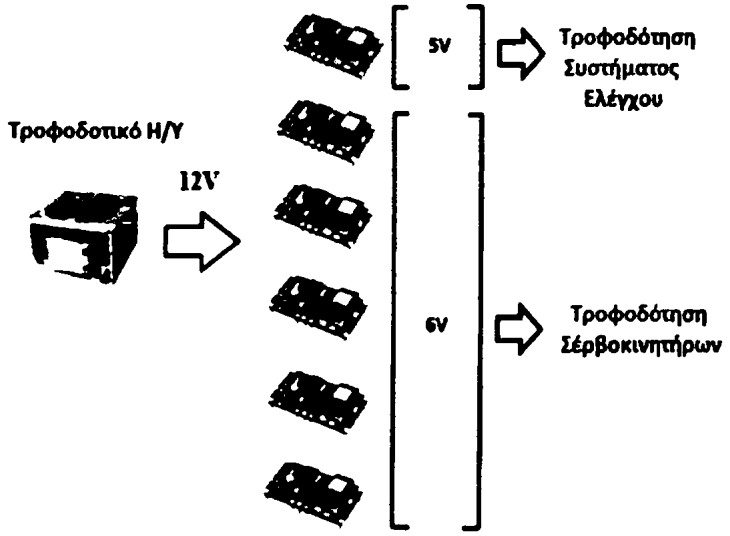
Στη συνέχεια συνδέουμε το Γκρι καλώδιο (Power Good) σε ένα LED σε σειρά με μία αντίσταση 330Ω και στο Ground. Το Γκρι καλώδιο έχει τιμή λογικό "0" (0V) όσο οι τάσεις εξόδου του τροφοδοτικού δεν έχουν πάρει τη μέγιστη τιμή τους και παίρνει λογικό "1" (5V) μόλις αυτό συμβεί. Τέλος το Καφέ καλώδιο είναι σήμα ανάδρασης και ελέγχει την πτώση τάσης πάνω στα πορτοκαλί καλώδια 3.3V. Πρέπει να είναι συνδεδεμένο μαζί με τα πορτοκαλί καλώδια.

Στην εικόνα 46 φαίνεται το τροφοδοτικό μετά τις τροποποιήσεις που έγιναν.

Εικόνα 46: Εσωτερικό Τροφοδοτικού

5.3 Επιλογή Regulators

Όπως αναφέρθηκε πριν, από τα 12V του τροφοδοτικού του Υ/Η χρειαζόμαστε μόνο τα 6V για τη μέγιστη απόδοση των Σ-Κ. Γι'αυτό το λόγο χρησιμοποιούμε Regulators για τον υποβιβασμό της τάσης.



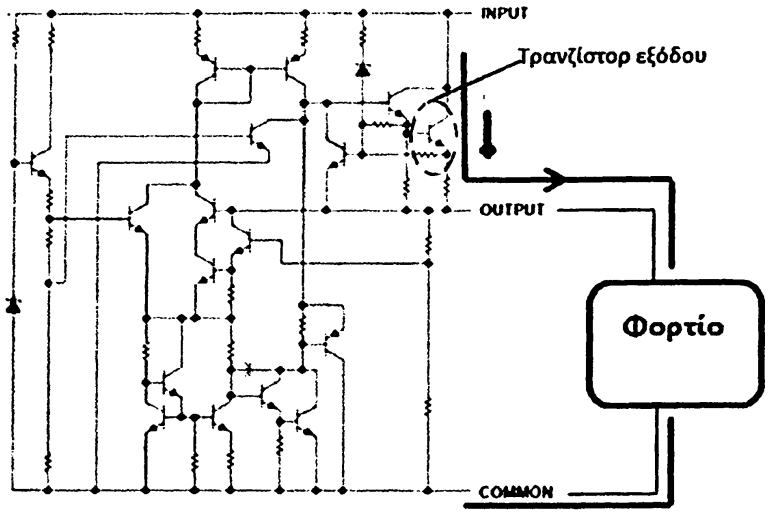
Εικόνα 47: Πλάνο τροφοδοσίας ρομποτικής χείρας

Γενικότερα υπάρχουν δύο είδη Regulators, οι γραμμικοί (linear) και οι (διακοπτόμενοι) switching.

Οι γραμμικοί έχουν απόδοση 20 – 50% ενώ οι διακοπτόμενοι έχουν 80 – 90%. Ο λόγος γι'αυτό βρίσκεται στη λειτουργία των δύο regulators.

5.3.1 Εξήγηση Λειτουργίας Linear και Switching regulators

Στους linear, το τρανζίστορ το οποίο διαμορφώνει την τάση εξόδου λειτουργεί στη γραμμική περιοχή με αποτέλεσμα να υπάρχει μια διαφορά δυναμικού μεταξύ συλλέκτη και εκπομπού και ταυτόχρονα το ρεύμα που θα τροφοδοτεί το φορτίο περνάει από το τρανζίστορ με αποτέλεσμα η παραγόμενη ισχύς να καταναλώνεται στο τρανζίστορ με τη μορφή θερμότητας (εικόνα 48).



Εικόνα 48: Ροή ρεύματος και θερμότητα

Αντίθετα με τους linear, στους switching regulators το τρανζίστορ βρίσκεται διαδοχικά στην περιοχή Κόρου και Αποκοπής λειτουργώντας ως διακόπτης με χρήση PWM (Pulse Width Modulation). Όταν το τρανζίστορ βρίσκεται στην Αποκοπή υπάρχει μεγάλη διαφορά δυναμικού στα άκρα του (ίση με την επιθυμητή τάση εξόδου) αλλά το διαπερνάει ένα πάρα πολύ μικρό ρεύμα της τάξης των μA. Έτσι η ισχύς που καταναλώνεται από το τρανζίστορ είναι πολύ μικρή. Όταν το τρανζίστορ βρίσκεται στον Κόρο το διαπερνάει μεγάλο ρεύμα αλλά η διαφορά δυναμικού στα άκρα του τρανζίστορ είναι πολύ μικρή και έτσι η καταναλισκόμενη ισχύς του τρανζίστορ ως θερμότητα είναι πάλι μικρή.

Συνήθως οι linear regulators χρησιμοποιούνται σε εφαρμογές που καταναλώνουν ισχύ < 1 Watt ενώ οι switching regulators χρησιμοποιούνται σε εφαρμογές που καταναλώνουν ισχύ > 1 Watt.

Όπως αναφέρθηκε πιο πριν το κάθε δάχτυλο του ρομποτικού χεριού θα τραβάει ρεύμα $I_{finger} \approx 3A$.

- Αν χρησιμοποιήσουμε linear regulators για τον υποβιβασμό της απώλεια σε μορφή θερμότητας πάνω στον Linear Regulator υπολογίζεται ως εξής:

$$\boxed{P_{wasted} = (Vin - Vout) * I_{finger}}$$

Δηλαδή, η τάση εισόδου από το τροφοδοτικό του υπολογιστή μείον την τάση εξόδου από τον linear regulator και όλο αυτό επί το ρεύμα που διαρρέει το κύκλωμα, δηλαδή τους τέσσερεις κινητήρες του κάθε ρομποτικού δαχτύλου.

Αντικαθιστώντας με τα στοιχεία που έχουμε για το κάθε δάχτυλο του ρομποτικού χεριού έχουμε:

$$P_{wasted} = (12V - 6V) * 3A \rightarrow P_{wasted} = 18 \text{ Watt}$$

Αυτό σημαίνει πως το 1/2 της ισχύος θα γινόταν θερμότητα πάνω στον linear regulator (η διαφορετικά η αποδοτικότητα του είναι 50%). Ακόμη, θα χρειαζόμασταν ένα πολύ ογκώδη απαγωγό θερμότητας (Heatsink) έτσι ώστε να προφυλάξουμε τον regulator από υπερθέρμανση και πιθανή καταστροφή του.

Συνολικά για τα 5 δάχτυλα του χεριού θα είχαμε απώλεια ίση με **90 Watt.**

- Αν χρησιμοποιήσουμε switching regulators με αποδοτικότητα 90% για τον υποβιβασμό της τάσης, η απώλεια ισχύος σε μορφή θερμότητας πάνω στον switching regulator μπορεί να βρεθεί ως εξής :

$$\left.\begin{array}{l} (P_{out}/P_{in}) *= Efficiency \\ P_{out} = V_{out} * I_{finger} \\ Efficiency = 90\% \end{array}\right\} \left.\begin{array}{l} P_{in} = \frac{18W}{0.9} \rightarrow P_{in} = 20W \\ \\ P_{out} = 6V * 3A = 18W \end{array}\right\} \left.\begin{array}{l} \\ P_{wasted} = P_{out} - P_{in} \\ P_{wasted} = 2 \text{ Watt} \end{array}\right\}$$

Συνολικά για τα 5 δάχτυλα του χεριού θα είχαμε απώλεια ίση με **10 Watt.**

Συγκρίνοντας τα δύο αποτελέσματα γίνεται φανερό πως η απώλεια ισχύος υπό μορφή θερμότητας στην πρώτη περίπτωση είναι πολύ μεγαλύτερη από την δεύτερη περίπτωση.

$$\Delta P = 90\text{Watt} - 10\text{Watt} = \mathbf{80\ Watt}$$

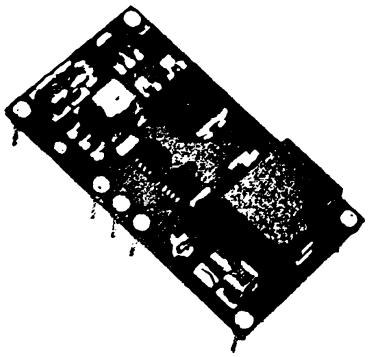
Αποδεικνύεται λοιπόν πως η χρήση switching regulators είναι και η καλύτερη επιλογή για την τροφοδότηση των Σ-Κ του ρομποτικού χεριού καθώς η κατανάλωση ισχύος είναι πολύ μικρότερη. Έτσι λοιπόν από το τροφοδοτικό του υπολογιστή συνδέουμε τα switching regulators τα οποία θα δίνουν την τάση που επιθυμούμε

Στην επόμενη παράγραφο γίνεται αναφορά στην επιλογή των Switching Regulators και διευκρινίζονται κάποια από τα σημαντικότερα χαρακτηριστικά τους.

5.3.2 Επιλογή Switching Regulators

Για το σύστημα επιλέχθηκαν οι switching regulators **PTN78020W [14]** από την εταιρία Texas Instruments. Τα κυριότερα χαρακτηριστικά τους είναι:

- Ρεύμα εξόδου I_{out} έως 6A
- Αποδοτικότητα έως 96%
- Ρυθμιζόμενη V_{out} (2.5 V - 12.6 V)
- Εύρος V_{in} **(7 V - 36 V)**
- Λειτουργία Ανάδρασης

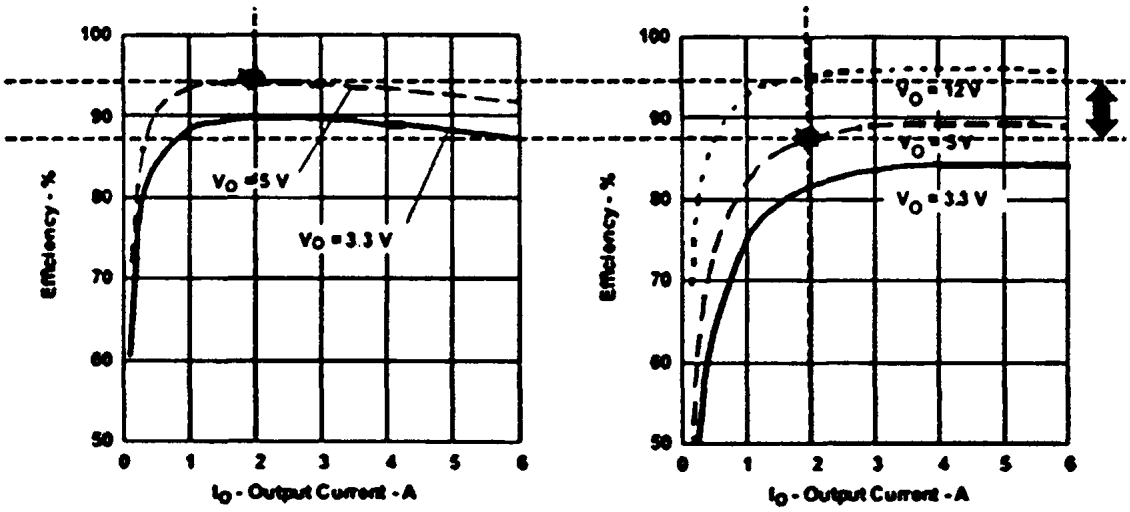


Εικόνα 49: Switching Regulatoc PTN78020W

Όπως αναφέρθηκε και προηγουμένως θα χρησιμοποιηθούν 5 switching regulators ένας για κάθε δάχτυλο. Θα μπορούσε να χρησιμοποιηθεί ένας ανα 2 δάχτυλα, αυτό όμως θα οδηγούσε τους regulators να λειτουργούν στο όριο των δυνατοτήτων τους (δηλ. 6A) κάτι το οποίο δεν είναι επιθυμητό.

Το PTN78020W σύμφωνα με το datasheet του κατασκευαστή έχει αποδοτικότητα έως και 96%. Το μέγεθος αυτό εξαρτάται από τρείς παράγοντες: την τάση εισόδου (V_{in}), την τάση εξόδου (V_{out}) και το ρεύμα εξόδου (I_{out}).

Από τα παρακάτω γραφήματα του datasheet (εικόνα 50) βλέπουμε πως για τάση εισόδου 7V (εικόνα 48α) και 15V (εικόνα 48β) , για τάση εξόδου 5V (και για τα δύο γραφήματα) και για ρεύμα εξόδου 1.5A (και για τα δύο γραφήματα) η αποδοτικότητα κυμαίνεται μεταξύ 95% και 87% αντίστοιχα.



Εικόνα 50α Εύρεση αποδοτικότητας των Regulators Εικόνα 50β

Εφόσον η τάση εισόδου (V_{in}) και η τάση εξόδου (V_{out}) στην οποία θα λειτουργούν τα PTN78020W είναι 12V και 6V αντίστοιχα και το ρεύμα με το οποίο θα τροφοδοτούν τους Σ-Κ θα κυμαίνεται μεταξύ $I_{finger\text{-}min}$= 1A και $I_{finger\text{-}max}$= 3A (ανάλογα με τις κινήσεις των δαχτύλων) παίρνοντας ως μέση τιμή τα 2A και παρατηρώντας τα παραπάνω διαγράμματα μπορούμε να πούμε πως η αποδοτικότητα των regulators θα κυμαίνεται μεταξύ 89% - 95%. Όπως θα φανεί στη συνέχεια είναι φανερό πως με χρήση linear regulators η καταναλισκόμενη ισχύς του συστήματος σε θερμότητα θα ήταν πολύ μεγαλύτερη καθώς η αποδοτικότητα θα ήταν περίπου 50%.

Μπορούμε λοιπόν να βρούμε την μετατρεπόμενη ισχύ σε θερμότητα πάνω στο PTN78020W βάσει της κυμαινόμενης αποδοτικότητας που θεωρήσαμε παραπάνω.

$$P_{out}/P_{in} = Efficiency_{min} = 0.89 \text{ ή } 89\%$$
$$P_{out}/P_{in} = Efficiency_{max} = 0.95 \text{ ή } 95\%$$

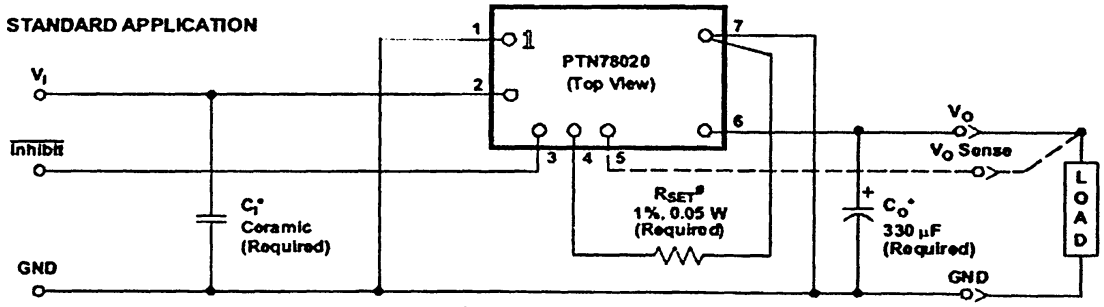
$$P_{out\_min} = Vout * I_{finger\_min} = 6V * 1A = 6W \rightarrow$$
$$P_{out\_max} = V_{out} * I_{finger\_max} = 6V * 3A = 18W \rightarrow$$

$$\rightarrow P_{in\_min} = 6W/0.89 = 6.74W$$
$$\rightarrow P_{in\_max} = 18W/0.95 = 18.94W$$

$$\boxed{P_{wasted\_min} = 740mW \\ P_{wasted\_max} = 940mW}$$

5.4 Σχεδίαση και Κατασκευή πλακετών Τροφοδοσίας

Η συνδεσμολογία που απαιτείται για τα PTN78020W απεικονίζεται στο επόμενο σχηματικό διάγραμμα.



Εικόνα 51: Κύκλωμα σύνδεσης του switching regulator

Για να παραχθεί στην έξοδο (pin 6) η επιθυμητή τάση (6V) θα πρέπει να τοποθετηθεί μεταξύ του pin 4 και της γείωσης μία αντίσταση ακριβείας 1% R_{SET} η οποία υπολογίζεται από τον παρακάτω τύπο:

$$R_{SET} = 54.9K\Omega * \frac{1.25V}{V_{out}-V_{min}} - R_p \text{ , όπου } V_{min} = 2.5V \text{ και } R_p = 6.49k\Omega$$

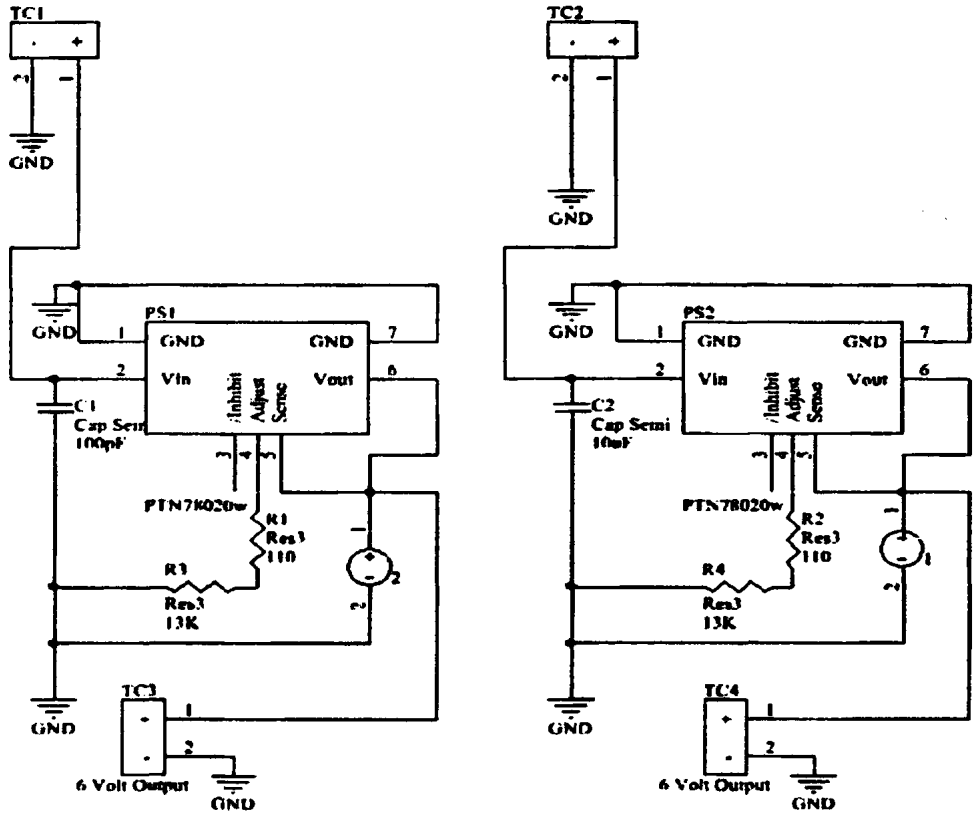
Έτσι λοιπόν,

για $V_{out} = 6V \rightarrow R_{SET} = 13.1\ k\Omega$

για $V_{out} = 5V \rightarrow R_{SET} = 20.9\ k\Omega$

Το datasheet μας δίνει την ελάχιστες τιμές που μπορούμε να έχουμε στους πυκνωτές C_{IN} =2.2μF (κεραμικός) και C_{OUT} =330μF (ηλεκτρολυτικός). Στο εργαστήριο υπήρχαν διαθέσιμοι ≥ 10μF (κεραμικοί) και 330μF (ηλεκτρολυτικοί). Επιλέχθηκαν C_{IN} =10μF και C_{OUT} = 330μF λόγω διαθεσιμότητας.

Στην εικόνα 52 παρουσιάζεται το σχηματικό διάγραμμα του κυκλώματος βάσει του οποίου θα σχεδιαστούν οι πλακέτες πάνω στις οποίες θα τοποθετηθούν οι switching regulators.



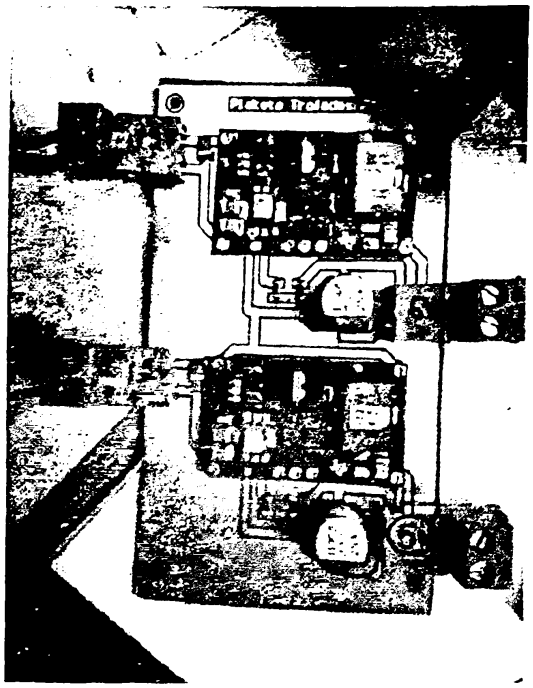
Εικόνα 52: Σχηματικό πλακέτας τροφοδοσίας

Το παραπάνω σχηματικό κύκλωμα αποτελείται από δύο πανομοιότυπες συνδεσμολογίες των δύο από τους έξι switching regulators που θα χρησιμοποιηθούν για την τροφοδοσία του συστήματος. Έτσι συνολικά κατασκευάστηκαν τρείς τέτοιες πλακέτες από τις οποίες, οι πέντε θα έχουν τάση 6V στην έξοδο τους ενώ η τελευταία θα έχει τάση 5V στην έξοδο της.

Τα 6V των πέντε πλακετών θα πηγαίνουν στην τροφοδότηση των σέρβο-κινητήρων ενώ τα 5V θα πηγαίνουν στη τροφοδότηση των μικροελεγκτών, των τελεστικών ενισχυτών και των αισθητήρων αγώγιμης μελάνης.

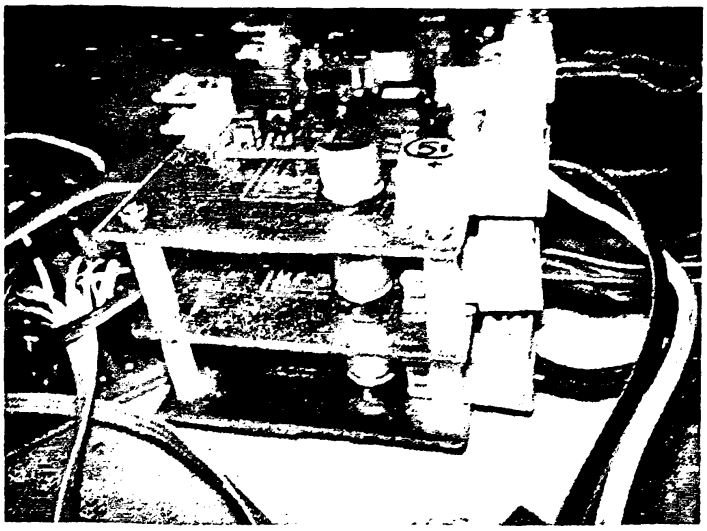
Ο λόγος για τον οποίο υπάρχουν δύο switching regulators ανά πλακέτα και όχι όλα σε μία πλακέτα είναι γιατί σε περίπτωση που θα εμφανιζόταν κάποια βλάβη σε κάποιον regulator κατά τη διάρκεια των διαφόρων δοκιμών του συστήματος να είναι πιο εύκολη η αντικατάσταση του από ένα νέο regulator χωρίς να επηρεαστεί το υπόλοιπο κομμάτι όπως για παράδειγμα στην περίπτωση ενός βραχυκυκλώματος.

Στην εικόνα 53 απεικονίζεται η πλακέτα, πάνω στην οποία τοποθετούνται οι δύο regulators, μετά την κατασκευή της. Στο παράρτημα Π3.1 δίνεται το Layout των πλακετών τροφοδοσίας.



Εικόνα 53: Κάτοψη πλακέτων τροφοδοσίας

Τέλος παρουσιάζεται η πλακέτα μετά την τοποθέτησή της μαζί με τις άλλες δυο .



Εικόνα 54: Τελική μορφή πλακετών τροφοδοσίας

ΚΕΦΑΛΑΙΟ 6^{ο}

Το σύστημα Ελέγχου

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο αναλύεται ο τρόπος με τον οποίο οι μικροελεγκτές του συστήματος παράγουν σήματα παλμών μεταβλητού εύρους PWM μέσω των χρονιστών τους (16-bit), τέτοια ώστε να οδηγήσουν τους κινητήρες της ρομποτικής χείρας. Στη συνέχεια αναλύεται η διαδικασία μετατροπής των αναλογικών σημάτων, τα οποία λαμβάνονται από την διάταξη αισθητήρων, σε ψηφιακή μορφή μέσω του μετατροπέα (ADC) των μικροελεγκτών. Επίσης αναλύεται ο τρόπος με τον οποίο τα ψηφιακά αυτά σήματα, αφού τροποποιηθούν μέσω κατάλληλων συναρτήσεων, ελέγχουν τα σήματα PWM μέσω των χρονιστών.

6.2 Οι Timers του ATmega2560

Οι σέρβο-κινητήρες που είναι υπεύθυνοι για την μετάδοση της κίνησης στα δάχτυλα της ρομποτικής χείρας, ελέγχονται από σήματα παλμών μεταβλητού εύρους (PWM) [6,8,11] τα οποία παράγονται από τους χρονιστές του μικροελεγκτή. Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο υπάρχουν δύο τύποι χρονιστών στον μικροελεγκτή, οι 8-bit και οι 16-bit.

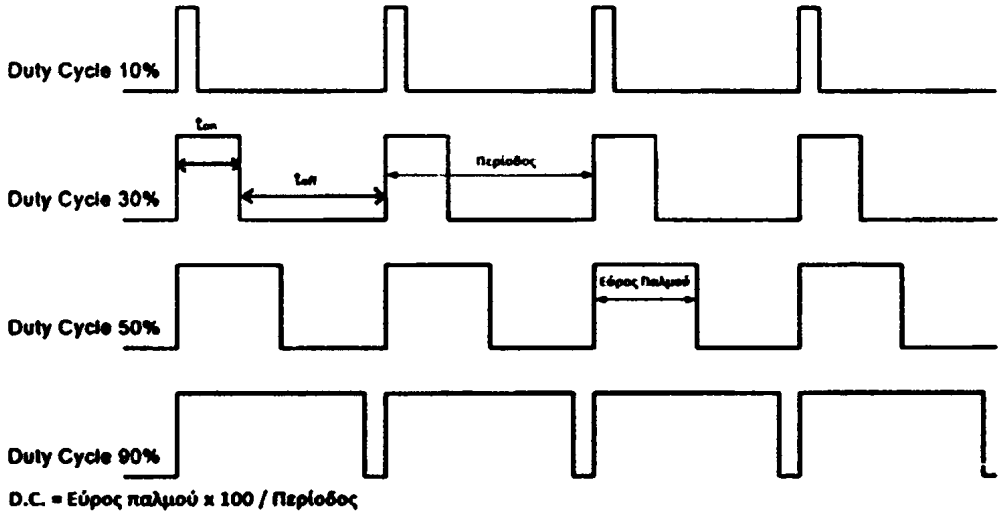
6.3 Pulse Width Modulation (PWM)

Με τη λειτουργία PWM είμαστε σε θέση να ελέγξουμε την θέση των σέρβο-κινητήρων με απόλυτη ακρίβεια αλλάζοντας το εύρος του παλμού κρατώντας σταθερή τη συχνότητα του. Το χρονικό διάστημα κατά το οποίο ο παλμός είναι σε κατάσταση HIGH (λογικό ¨1¨) ονομάζεται t_{on}. Το χρονικό διάστημα κατά το οποίο ο παλμός είναι σε κατάσταση LOW (λογικό ¨0¨) ονομάζεται t_{off}. Το άθροισμα αυτών των δύο χρονικών διαστημάτων αποτελεί και την περίοδο του σήματος PWM. Δηλαδή, $T = t_{on} + t_{off}$. Οι χρόνοι αυτοί μετρώνται σε sec, msec, μsec κ.λ.π. ανάλογα με την περίοδο του σήματος που θέλουμε να κατασκευάσουμε η να επεξεργαστούμε. Επίσης, ένας όρος που χρησιμοποιείται αρκετά είναι το Duty Cycle (D.C.). Ως D.C. ορίζεται ο λόγος του χρόνου που το σήμα είναι σε κατάσταση HIGH

προς την περίοδο (T = $t_{on} + t_{off}$) αυτού του σήματος και μετριέται σε ποσοστό επί τοις εκατό. Το D.C. περιορίζεται από την περίοδο του σήματος, καθώς δεν μπορεί να γίνει μεγαλύτερο από αυτή, όπως φαίνεται και από τον επόμενο τύπο:

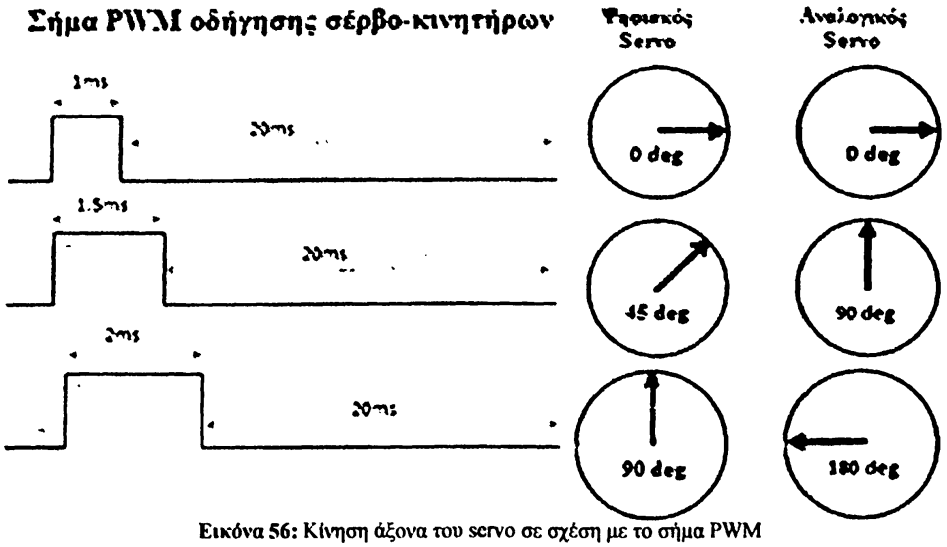
$$D.C.\% = \frac{t_{on}}{t_{on} + t_{off}} \times 100 = \frac{t_{on}}{T} \times 100$$

Στην εικόνα 55 παρουσιάζονται τέσσερις παλμοσειρές ίδιας περιόδου με διαφορετικά D.C. κάθε φορά. Αν για παράδειγμα η περίοδος του σήματος είναι T = 20msec τότε το D.C. = 10% αντιστοιχεί σε t_{on} = 2 msec, το D.C. = 30% αντιστοιχεί σε t_{on}=6 msec και το D.C. = 90% αντιστοιχεί σε t_{on} = 18 msec



Εικόνα 55: Σήματα PWM

Οι σέρβο-κινητήρες αντιλαμβάνονται PWM σήματα όπως αυτά της εικόνας 55 μόνο που έχουν κάποιους περιορισμούς. Η περίοδος του PWM σήματος πρέπει να βρίσκεται μεταξύ 22.2 msec και 18.18 msec (45Hz και 55Hz αντίστοιχα) . Επιπλέον, το εύρος του παλμού πρέπει να είναι ανάμεσα στα χρονικά περιθώρια 1msec – 2msec. Το Duty Cycle για t_{on} = 1msec εφαρμόζοντας τον παραπάνω τύπο ισούται με 5% και το Duty Cycle για t_{on} = 2msec ισούται με 10%. Για D.C. 5% ο άξονας του σέρβο-κινητήρα μετακινείται στις 0 μοίρες, ενώ για D.C. 10% ο άξονας του σέρβο-κινητήρα μετακινείται στις 90 ή 180 μοίρες ανάλογα με το αν ο σέρβο-κινητήρας είναι ψηφιακός η αναλογικός αντίστοιχα.



Εικόνα 56: Κίνηση άξονα του servo σε σχέση με το σήμα PWM

Μετά από δοκιμές που έγιναν στους σέρβο-κινητήρες της ρομποτικής χείρας βρέθηκε ότι, το εύρος των παλμών που «αντιλαμβάνονται» κυμαίνεται μεταξύ των χρονικών διαστημάτων t_{on}= 900 μsec και t_{on}= 2.1 msec.

Πέρα από αυτά τα όρια οι σέρβο-κινητήρες δεν αντιδρούν καθώς είναι ρυθμισμένοι να σταματούν μέσω κάποιων φρένων τα οποία είναι τοποθετημένα επάνω στους οδοντωτούς τροχούς του συστήματος μεταφοράς της κίνησης του άξονα τους. Συνίσταται τα σήματα PWM να μην ξεπερνούν αυτά τα όρια ειδάλλως υπάρχει μεγάλη πιθανότητα να καεί ο κινητήρας που βρίσκεται μέσα στη διάταξη των σέρβο-κινητήρων.

6.4 Επιλογή χρονιστών (Timers) στον ATmega2560

Ο ATmega2560 διαθέτει 4 timers των 16 bit (Timer1,3,4,5) και δύο timers των 8 bit (Timer 0,2).Αυτό σημαίνει πως οι 16-bit timers μπορούν να μετρήσουν από το 0 έως το 65535 (65536 τιμές) ενώ οι 8-bit timers μπορούν να μετρήσουν από το 0 έως το 255 (256 τιμές).

Σε αυτό το σημείο θα πρέπει να αναφερθεί πως χρησιμοποιείται κρύσταλλος 16 MHz στους μικροελεγκτές του συστήματος έτσι ώστε οι εντολές να εκτελούνται στο μικρότερο δυνατό χρόνο. Δηλαδή 62,5 nsec – 125 nsec ανά εντολή καθώς, μία εντολή στους AVR απαιτεί 1-2 κύκλους μηχανής για την ολοκλήρωση της. Έτσι τα σήματα PWM θα πρέπει να παραχθούν από τους μικροελεγκτές με βάση τα 16MHz του ρολογιού. Σε πολλές εφαρμογές είναι αναγκαία η χρήση πολύ χαμηλότερων

συχνοτήτων από αυτές που μπορούν να παράγουν οι χρονιστές(8-bit ή 16-bit) του ATmega2560 (μετρώντας απλά από το μηδέν μέχρι τη μέγιστη τιμή τους). Γι'αυτό το λόγο, οι χρονιστές διαθέτουν ένα μηχανισμό για τη διαίρεση της συχνότητας του ρολογιού του μικροελεγκτή (MHz) στην είσοδο τους έτσι ώστε να παράγουν σήματα πολύ χαμηλών συχνοτήτων (Hz) στις εξόδους τους. Αυτός ο μηχανισμός λέγεται Prescaler και δίνει την δυνατότητα διαίρεσης της συχνότητας του ρολογιού με τιμές όπως, ÷8, ÷64, ÷256, ÷1024 (Παράρτημα Π4, Εικ.114).

Εφόσον οι σέρβο-κινητήρες είναι σε θέση να αντιληφθούν σήματα με συχνότητα 50Hz θα πρέπει να γίνει μελέτη για να διαπιστωθεί ποιοι timers (8 ή 16 bit) των μικροελεγκτών μπορούν να χρησιμοποιηθούν για την παραγωγή αυτής της συχνότητας.

6.4.1 Timers 8-bit

Αν χρησιμοποιηθούν οι timers των 8-bit η μικρότερη συχνότητα πού μπορεί να παραχθεί είναι f = 61Hz.

Αυτό αποδεικνύεται ως εξής:

1. Απαιτείται T_{clk} = 20msec → f_{clk} = 50Hz με χρήση κρυστάλλου f=16MHz
2. Χρησιμοποιώντας τον μεγαλύτερο Prescaler = 1024 που μπορούμε, η συχνότητα του timer πέφτει σε f_{timer} = 15,625 kHz → T_{timer} = 64μsec.
3. Πολλαπλασιάζοντας αυτό το χρόνο με 256 (όσα και τα βήματα του χρονιστή) παράγεται μία συχνότητα $\mathbf{f_{generated} = 61Hz.}$

Αυτό σημαίνει πως, στην καλύτερη περίπτωση, η χαμηλότερη συχνότητα που μπορούμε να πάρουμε με τους 8-bit timers είναι 61Hz. Φυσικά αυτό υπερβαίνει κατά 11Hz την επιθυμητή συχνότητα για την λειτουργία των σέρβο-κινητήρων. Συνεπώς, οι 8-bit timers απορρίπτονται για την υλοποίηση του συστήματος.

6.4.2 Timers 16-bit

Αν χρησιμοποιηθούν οι timers των 16-bit η μικρότερη συχνότητα πού μπορεί να παραχθεί είναι F = 4,2Hz.

Αυτό αποδεικνύεται ως εξής:

1. Απαιτείται T_{clk} = 20msec → f_{clk} = 50Hz με χρήση κρυστάλλου f=16MHz
2. Χρησιμοποιώντας τον μεγαλύτερο Prescaler = 1024 που μπορούμε, να πάρουμε η συχνότητα του χρονιστή ελαττώνεται σε f_{timer} = 15,625 kHz → T_{timer} = 64usec.
3. Πολλαπλασιάζοντας αυτό το χρόνο με 65536 (όσα και τα βήματα του χρονιστή) παράγεται μία συχνότητα $\mathbf{F_{generated}}$ **= 4,194 Hz.**

Εφόσον λοιπόν η ελάχιστη συχνότητα που μπορεί να παραχθεί με τους 16-bit timers είναι 4,2Hz, αυτό σημαίνει πως και μία μεγαλύτερη συχνότητα όπως τα 50Hz θα είναι δυνατό να παραχθεί με κρύσταλλο 16MHz. Έτσι επιλέγουμε τους 16-bit timers για την παραγωγή των σημάτων PWM.

6.5 Υλοποίηση σημάτων PWM με timers 16-bit

Οι timers 16-bit δίνουν τη δυνατότητα παραγωγής PWM τριών τύπων.

- Fast PWM mode
- Phase correct mode
- Phase and Frequency correct mode

Και οι τρείς τύποι παραγωγής PWM μπορούν να υλοποιήσουν τα σήματα που απαιτούνται για την οδήγηση των σέρβο-κινητήρων. Επιλέχθηκε ο πρώτος τύπος ως ο πλέον κατάλληλος αφού, δεν δημιουργείται κάποιο πρόβλημα με τη φάση και τη συχνότητα των σημάτων.

Η λειτουργία Phase correct χρησιμοποιείται σε περιπτώσεις όπου η φάση του παραγόμενου σήματος μεταβάλλεται με την αλλαγή του D.C. του παλμού και η λειτουργία Frequency correct χρησιμοποιείται όταν η συχνότητα του 16-bit timer είναι επιθυμητό να αλλάζει συνεχώς από το πρόγραμμα χωρίς αυτό να διαταράσσει την μέτρηση του timer και να δημιουργεί κενά στην αλλαγή μεταξύ δύο διαφορετικών συχνοτήτων.

6.5.1 Προγραμματισμός χρονιστών 16-bit με Fast PWM

Στη λειτουργία Fast PWM [6,8,16], ο χρονιστής μετράει κανονικά από το 0 μέχρι μία μέγιστη τιμή (TOP) που έχει επιλέξει ο προγραμματιστής και η οποία είναι

μικρότερη ή ίση με την τιμή 65535. Μέσω αυτής της τιμής, όπως θα φανεί αργότερα, ρυθμίζεται η συχνότητα του σήματος PWM.

Στον πίνακα 6 παρουσιάζονται οι δεκαέξι διαφορετικοί τρόποι παραγωγής παλμών με την χρήση των 16-bit χρονιστών οι οποίοι επιλέγονται μέσω των bits WGMn0/1/2/3 τα οποία βρίσκονται στους καταχωρητές επιλογής mode, TCCRnA και TCCRnB. Ο τρόπος που είναι όμως ο ιδανικότερος για την υλοποίηση των Fast PWM σημάτων τα οποία χρειαζόμαστε για την οδήγηση των κινητήρων είναι κυκλωμένος με κόκκινο χρώμα. Ο λόγος εξηγείται αμέσως μετά.

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	-	-	-
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Πίνακας 6: Επιλογή PWM mode

Όπως αναφέρθηκε και στην αρχή αυτού του κεφαλαίου ο μικροελεγκτής ATmega2560 αποτελείται από 4 χρονιστές 16-bit, τους Timer1/3/4/5. Αυτοί οι χρονιστές έχουν ο καθένας τους:

- Ένα καταχωρητή μέτρησης τον TCNTn (Timer-Counter) ο οποίος έχει αρχική τιμή το μηδέν και αυξάνεται σε κάθε χτύπο του ρολογιού (εννοείται αφού ενεργοποιηθεί ο χρονιστής) μέχρι να φτάσει τη μέγιστη τιμή του που είναι η 65535 και στη συνέχεια μηδενίζεται επαναλαμβάνοντας συνεχώς αυτή τη διαδικασία.
- Τρείς καταχωρητές σύγκρισης , τους OCRnA, OCRnB και OCRnC (Output Compare Register) στους οποίους αντιστοιχεί και μία έξοδος – ακροδέκτης, οι OCnA/B/C όπως φαίνεται και στην εικόνα 57.

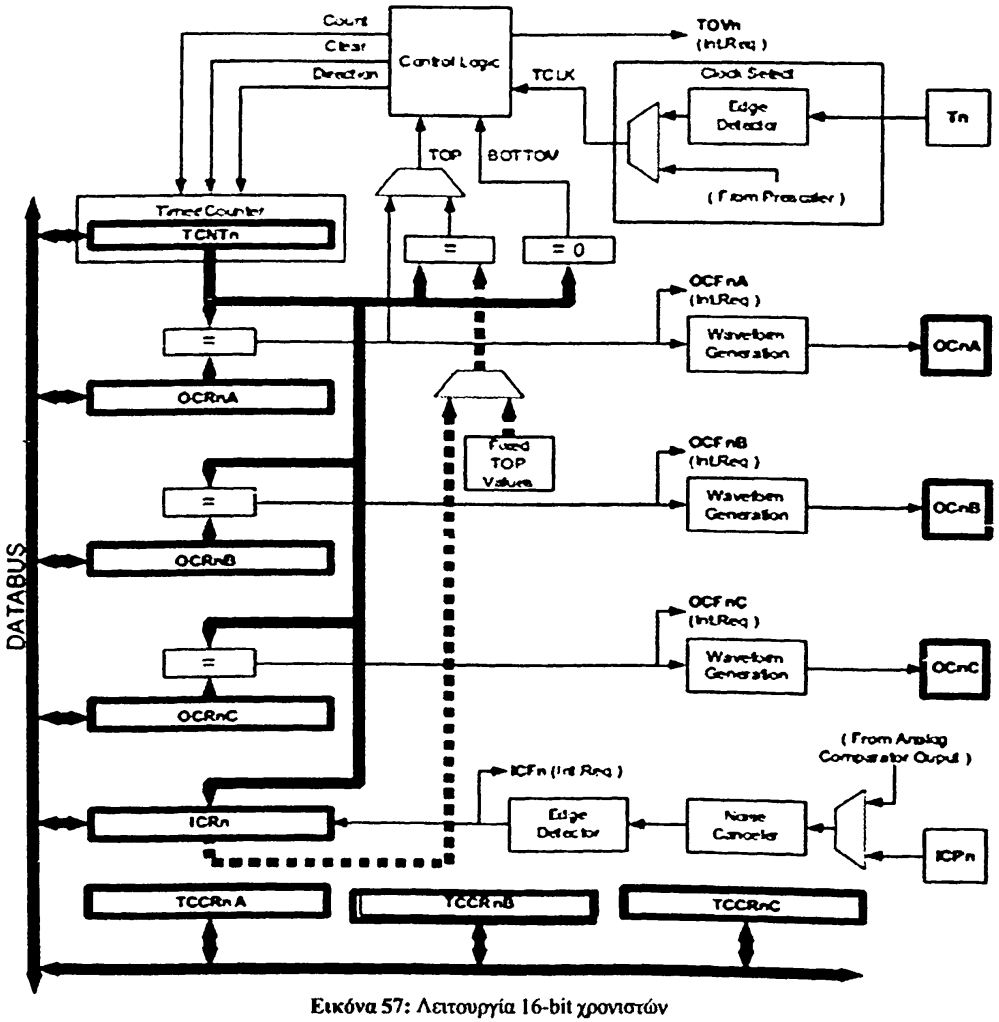
Εδώ να σημειωθεί πως όπου n = 1,3,4,5 (ο αριθμός του χρονιστή).

Ο ρόλος αυτών των καταχωρητών (OCRn) είναι να συγκρίνουν την τιμή που περιέχεται σε αυτούς (η οποία εισάγεται μέσω του προγράμματος), με την τιμή του καταχωρητή μέτρησης του χρονιστή τους (TCNTn) σε κάθε κύκλο του ρολογιού του χρονιστή. Μόλις οι δύο τιμές, καταχωρητή μέτρησης του χρονιστή TCNT και καταχωρητή OCRnA/B/C, εξισωθούν τότε οι αντίστοιχες έξοδοι αλλάζουν κατάσταση και από λογικό "1" γίνονται λογικό "0" ενώ ο χρονιστής συνεχίζει να αυξάνει μέχρι να φτάσει την τιμή TOP που έχει οριστεί από το πρόγραμμα.

Μόλις φτάσει την μέγιστη τιμή TOP, ο καταχωρητής μέτρησης του χρονιστή TCNTn μηδενίζεται και επαναλαμβάνει την ίδια διαδικασία.

Η μέγιστη τιμή TOP τοποθετείται μέσω προγράμματος στον καταχωρητή ICRn (Input Capture Register). Από την τιμή TOP ρυθμίζουμε και την συχνότητα των παραγόμενων παλμών.

Τέλος, οι καταχωρητές OCRnA/B/C, TCNTn και ICRn είναι 16-bit και μπορούν να γραφούν και να διαβαστούν οποιαδήποτε στιγμή μέσω προγράμματος.



Εικόνα 57: Λειτουργία 16-bit χρονιστών

Για να τοποθετήσουμε τον χρονιστή στη λειτουργία που μας ενδιαφέρει θα πρέπει να ρυθμίσουμε ανάλογα τους καταχωρητές λειτουργίας TCCRnA/B του κάθε χρονιστή.

Αυτοί οι τρείς καταχωρητές έχουν μήκος 8 bit ο καθένας. Το κάθε bit στοχεύει και σε μία ρύθμιση του χρονιστή (n). Έτσι αλλάζοντας τις τιμές των bits των καταχωρητών αυτών ο χρονιστής αναλαμβάνει την εκτέλεση και μίας διαφορετικής λειτουργίας.

Όπως αναφέρθηκε στην προηγούμενη σελίδα ο καταχωρητής ICRn είναι υπεύθυνος για την παραγωγή της επιθυμητής συχνότητας (50Hz) τοποθετώντας σε αυτόν μέσω προγράμματος μία τιμή η οποία θα αντιπροσωπεύει και την τιμή TOP για τον χρονιστή στον οποίο και αντιστοιχεί.

Αυτή η τιμή υπολογίζεται από τους εξής τύπους

$$f_{generated\ wave} = \frac{F_{timer\ clk}}{TOP+1} \qquad f_{timer\ clk} = \frac{F_{oscillator}}{N}$$

$$\Rightarrow \quad f_{generated\ wave} = \frac{F_{timer\ clk}}{(TOP+1)*N}$$

Όπου N = Η τιμή που έχει ο Prescaler

Το ζητούμενο είναι η τιμή TOP .

Έτσι για $F_{generated\ wave}$ = 50Hz , $F_{timer\ clk}$ = 16MHz και N = 8 η τιμή TOP προκύπτει **TOP = ICRn = 39999.** Άρα στον καταχωρητή ICRn τοποθετούμε την τιμή 39999 (0x9C3F σε hex).

Στη συνέχεια θα πρέπει να τοποθετηθούν στους καταχωρητές TCCRnA/B εκείνες οι τιμές που θα θέσουν τους χρονιστές σε λειτουργία Fast PWM mode 14, με τιμή Prescaler = 8 και ενεργοποίηση και των τριών εξόδων OCnA/B/C ως μη αντιστραμμένες (δηλαδή οι έξοδοι είναι σε κατάσταση "High" μέχρι να εξισωθούν οι καταχωρητές OCRnA/B/C και TCNTn, και μετά αλλάζουν σε κατάσταση "0"). Έτσι,

- Για ***Fast PWM*** mode 14, πρέπει στον TCCRnA τα bit WGMn0 και WGMn1 να πάρουν τιμές "0" και "1" αντίστοιχα και στον TCCRnB τα bit WGMn2 και WGMn3 να πάρουν τιμές "1" και "1".

- Για ***ενεργοποίηση και των τριών εξόδων*** OCnA/B/C ως μη αντιστραμμένες, πρέπει στον καταχωρητή TCCRnA τα bit COMnA1, COMnA0, COMnB1, COMnB0, COMnC1, COMnC0 να πάρουν τις τιμές "1" και "0","1" και "0", "1" και "0" αντίστοιχα
- Για ***Prescaler = 8*** , πρέπει στον TCCRnB τα bit CSn1 και CSn0 να πάρουν τις τιμές "1" και "0" αντίστοιχα.

Έτσι → TCCRnA = 0b10101010 (0xAA σε hex) και

TCCRnB = 0b00011010 (0x1A σε hex)

Τέλος, θα πρέπει να υπολογιστούν οι τιμές που θα τοποθετηθούν στους καταχωρητές OCRnA/B/C έτσι ώστε να επιτευχθούν τα επιθυμητά D.C. των παλμών PWM τα οποία θα είναι σε θέση να οδηγήσουν τους σέρβο-κινητήρες.

Η τιμή του D.C. για Fast PWM δίνεται από τον εξής τύπο:

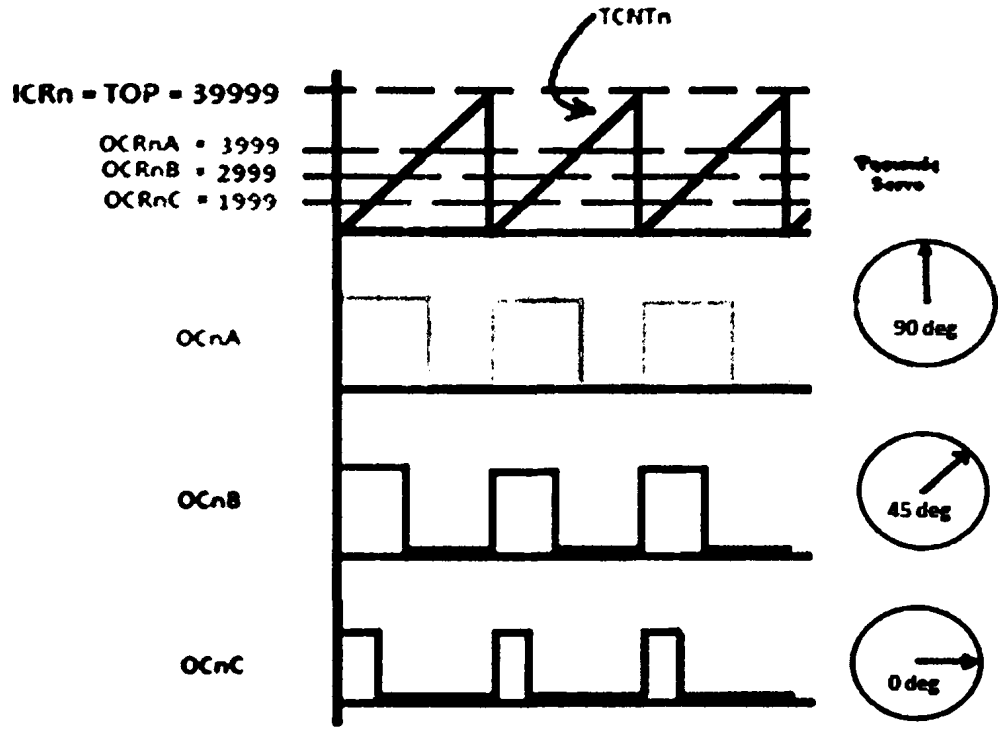
$$Duty\ Cycle = \frac{OCRnX}{TOP + 1} \times 100$$

Παράλληλα, εφόσον έχει βρεθεί πως η συχνότητα του παραγόμενου σήματος είναι $F_{generated\ wave}$ = 50Hz ($T_{generated\ wave}$ = 20msec) και το επιθυμητό πλάτος παλμού για την οδήγηση των σέρβο-κινητήρων κυμαίνεται από 1 msec - 2 msec μπορεί εύκολα να βρεθεί (με απλή μέθοδο των τριών) πως το Duty Cycle για 1 msec έχει τιμή D.C. = 5% και για 2 msec έχει τιμή D.C. = 10%. Τοποθετώντας αυτά τα ποσοστά στον προηγούμενο τύπο βρίσκουμε τις τιμές των OCRnA/B/C για 1 msec και 2 msec να είναι 1999 (0x07CF σε hex) και 3999 (0x0F9F σε hex) αντίστοιχα.

Συγκεντρώνοντας τα αποτελέσματα που βρήκαμε έχουμε:

- **TOP = ICRn = 39999,** για $F_{generated\ wave}$ = 50Hz
- TCCRnA = 0b**10101010** (0xAA σε hex)
- TCCRnB = 0b**00011010** (0x1A σε hex)
- Για 5% D.C (1 msec) **OCRnA/B/C = 1999**
- Για 10% D.C (2 msec) **OCRnA/B/C = 3999**

Έτσι, τα σήματα PWM τα οποία παράγονται μετά από αυτή τη διαδικασία παρουσιάζονται στην εικόνα 58. Ο τριγωνικός παλμός στην εικόνα είναι η τιμή του καταχωρητή TCNTn του χρονιστή (n) 16-bit καθώς αυξάνει στο χρόνο. Μόλις ο καταχωρητής TCNTn φτάσει την μέγιστη τιμή TOP, στον επόμενο κύκλο μηχανής μηδενίζεται και αρχίζει πάλι από την αρχή. Στο δεξί μέρος της εικόνας φαίνονται οι περιστροφές των αξόνων των σέρβο-κινητήρων ανάλογα με το σήμα PWM.

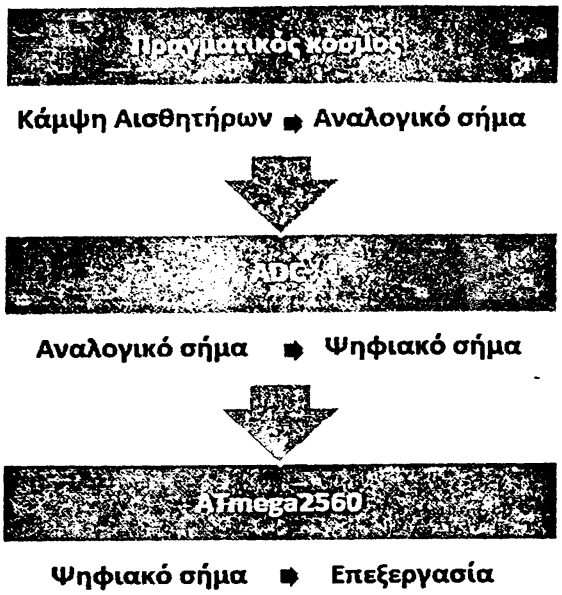


Εικόνα 58: Τιμές καταχορητών για κίνηση των servo

6.6 Λήψη αναλογικών σημάτων από την διάταξη αισθητήρων στα κανάλια του ADC

Όπως αναφέρθηκε και στον 4° κεφάλαιο, θα χρησιμοποιηθούν συνολικά 17 αισθητήρια κάμψης, 13 αγώγιμης μελάνης και 4 ωμικού άνθρακα. Κάθε ένα από αυτά τα αισθητήρια θα ελέγχει την κάμψη μιας άρθρωσης της ανθρώπινης χείρας και αντίστοιχα θα παράγει ένα αναλογικό σήμα τάσης μέσω της πλακέτας παραγωγής σημάτων που αναλύθηκε επίσης στο 4° κεφάλαιο.

Αυτά τα αναλογικά σήματα στη συνέχεια μεταφέρονται στις εισόδους των ADC των δύο μικροελεγκτών όπου και επεξεργάζονται μέσω της διαδικασίας που αναλύεται παρακάτω. Στην εικόνα 59 αυτό δίνεται ένα εποπτικό διάγραμμα της πλήρους διαδικασίας.



Εικόνα 59: Λογική επεξεργασίας σημάτων απο τον ADC

6.6.1 Ο ADC του ATmega2560

Σε αυτή την ενότητα παρουσιάζεται η διαδικασία μετατροπής των αναλογικών σημάτων σε ψηφιακή μορφή από τον A/D του μικροελεγκτή ATmega2560 [6,8,17]. Αυτός ο ADC έχει τα παρακάτω χαρακτηριστικά,

- Έχει ανάλυση 10-bits που σημαίνει ότι διαθέτει μικρότερο μέγεθος ανά βήμα (step size) από έναν ADC 8-bit. Μέγεθος ανά βήμα είναι η μικρότερη μεταβολή που μπορεί να ανιχνευθεί από τον ADC.
- Είναι συνδεδεμένος με ένα πολυπλέκτη (MUX) 16 καναλιών ο οποίος επιτρέπει την μετατροπή μέχρι και 16 αναλογικών τάσεων, που βρίσκονται στα κανάλια PORTF και PORTK του μικροελεγκτή, σε ψηφιακές τιμές.
  Διαθέτει ξεχωριστό ακροδέκτη τροφοδοσίας ,τον AV_{CC}, ο οποίος δεν πρέπει να έχει διακύμανση τάσης μεγαλύτερη από ± 0.3V από την V_{CC} του μικροελεγκτή ώστε να κάνει ακριβείς μετρήσεις των σημάτων.
- Παρέχει τρείς τάσης αναφοράς 1.1V, 2.56V, ή η AV_{CC} από τις οποίες μπορεί να επιλεχθεί η μία μέσω software.

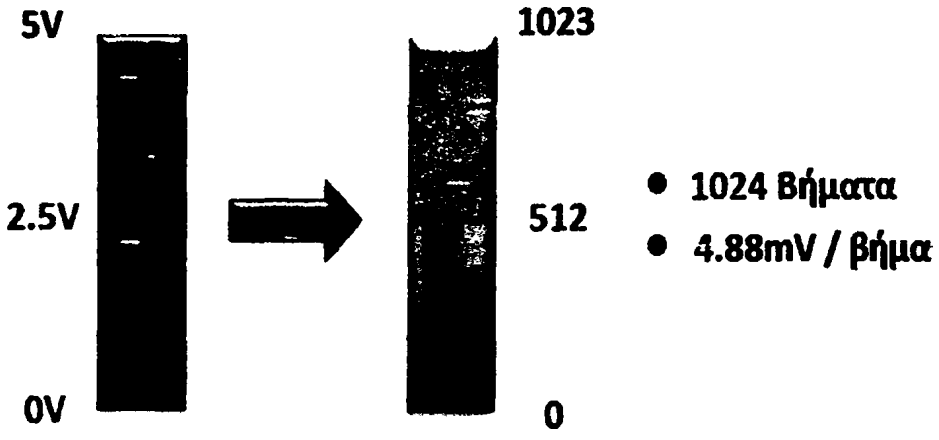
6.6.2 Τάση Αναφοράς των ADC του συστήματος

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο τα κανάλια του ADC έχουν ανάλυση 10–bit. Αυτό σημαίνει ότι αν εμείς θέλουμε να ανιχνεύσουμε ένα αναλογικό σήμα που μεταβάλλεται από 0V μέχρι 5V και να το μετατρέψουμε σε ψηφιακό, πρέπει πρώτα να δώσουμε στον ADC μια τάση αναφοράς (V_{REF}), όπου ο ADC θα συγκρίνει το αναλογικό σήμα προς μέτρηση σύμφωνα με αυτή την τάση. Η τάση αναφοράς παίρνει την μέγιστη τιμή του αναλογικού σήματος προς μέτρηση. Στη συνέχεια ο ADC παίρνει την τάση αναφοράς (V_{REF}) και τη διαιρεί με την ανάλυση του η οποία είναι 2^{10} =1024 ίσα μέρη. Δηλαδή κάθε μέρος (ή βήμα) του ADC θα αντιστοιχεί σε (5V)/1024 = **4.88mV**.

Ο γενικός τύπος είναι:

$$Μεγεθος\ Βήματος = \frac{Τάση\ αναφοράς\ (V_{REF})}{Ανάλυση\ μετατροπής\ (ADC\ Resolution)}$$

Έτσι η πλήρης αντιστοιχία του αναλογικού με το ψηφιακό σήμα φαίνεται καλύτερα στην εικόνα 60 για τάση αναφοράς, $V_{REF} = 5V$.



Εικόνα 60: Ισοδυναμία Αναλογικού και ψηφιακού σήματος

Τα αναλογικά σήματα που διαβάζονται από τους αισθητήρες κάμψης κυμαίνονται μεταξύ 2.5V (περίπου) και 0V. Ως Τάση Αναφοράς στους μικροελεγκτές χρησιμοποιήθηκε η τιμή 5V. Ένα εύλογο ερώτημα εδώ θα ήταν γιατί

χρησιμοποιήθηκαν τα 5V και όχι τα 2.5V ως τάση αναφοράς, εφόσον το σήμα δεν υπερβαίνει τα 2.5V. Η απάντηση είναι πως επειδή τα αισθητήρια που βρίσκονται τοποθετημένα ανάμεσα από τα γάντια $1^{ου}$ και $2^{ου}$ επιπέδου (όπως αναλύθηκε στο $4^{ο}$ κεφάλαιο) λόγω τριβών και πιέσεων από τις επιφάνειες τους κατά την τοποθέτηση τους στο χέρι του χρήστη, μπορεί να προκαλέσουν τάση μεγαλύτερη από 2.5V. Υπενθυμίζεται εδώ πως η αντίσταση των αισθητηρίων κατά την θέση ηρεμίας τους κυμαίνεται μεταξύ 1.9ΚΩ και 3.5ΚΩ για τα αισθητήρια αγώγιμης μελάνης, και 90ΚΩ με 120ΚΩ για τα αισθητήρια ωμικού άνθρακα. (Πίνακας 1, Κεφάλαιο $4^{ο}$).

Έτσι λοιπόν καθώς ο χρήστης τοποθετεί το γάντι, μικρές αλλαγές είναι σίγουρο πως θα επέλθουν στην αντίσταση των αισθητήρων (κάτι που διαπιστώθηκε από τις πολλές δοκιμές πάνω στην ρομποτική χείρα) με αποτέλεσμα τα παραγόμενα αναλογικά σήματα από τους διαιρέτες τάσης να έχουν τιμές όπως 2.8V, 3V ακόμη και 3.5V. Αυτές οι τιμές τάσεων όμως δεν θα είναι ανιχνεύσιμες από τον ADC και θα χάνονται καθώς θα έχει τοποθετηθεί τάση αναφοράς 2.5V. Αυτό με τη σειρά του θα έχει σαν αποτέλεσμα την αλλοίωση της μετατροπής και η μετατροπή αυτή θα βγάλει λανθασμένη κίνηση στον αντίστοιχο σέρβο-κινητήρα.

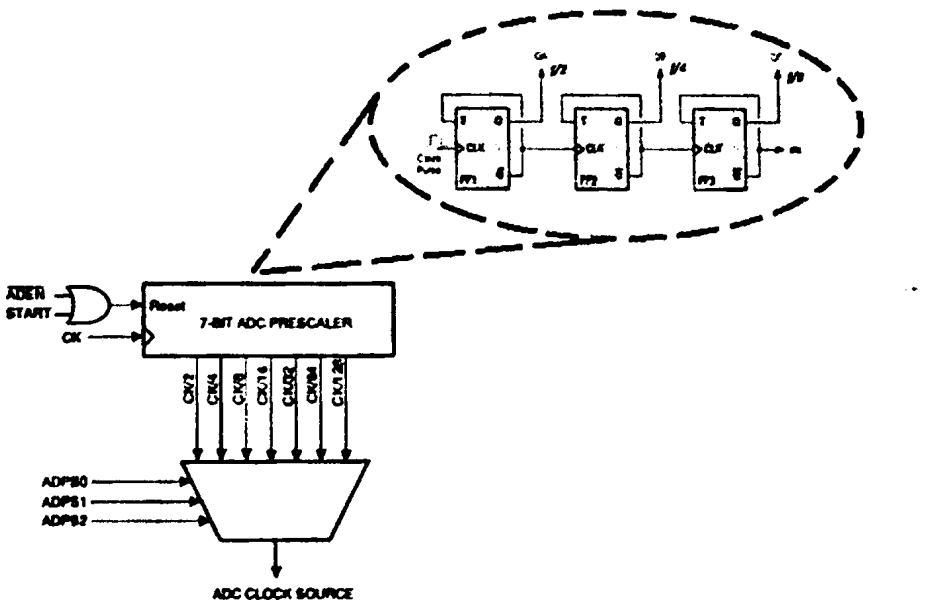
Γι'αυτό λοιπόν το λόγο χρησιμοποιήθηκε για ασφάλεια τάση αναφοράς $V_{REF} = 5V$ έτσι ώστε να ανιχνεύονται όλες οι τιμές τάσης και πλέον μέσω των συναρτήσεων που αναλύονται στη συνέχεια είναι δυνατή η πλήρης επεξεργασία των σημάτων και κατ'επέκταση ο πλήρης έλεγχος των σέρβο-κινητήρων.

6.6.3 Συχνότητα Λειτουργίας του ADC

Ο ADC των μικροελεγκτών για να κάνει ακριβείς και σωστές μετρήσεις χρειάζεται κάποιο χρόνο. Ο χρόνος αυτός εξαρτάται από την μέθοδο την οποία χρησιμοποιεί ο ADC για να κάνει τις μετατροπές και η οποία λέγεται Successive Approximation (αναλυτική παρουσίαση της μεθόδου γίνεται στο παράρτημα Π5.1.1).

Σύμφωνα με το Datasheet του μικροελεγκτή ο ADC κάνει ακριβείς μετατροπές σε συχνότητες που κυμαίνονται μεταξύ 50KHz και 200KHz. Ο ADC όμως δεν έχει δικό του ρολόι (κρύσταλλο) για την παραγωγή αυτών των συχνοτήτων και χρησιμοποιεί το ρολόι του μικροελεγκτή. Η συχνότητα όμως στην οποία λειτουργούν οι μικροελεγκτές είναι της τάξης των MHz. Έτσι η συχνότητα αυτή θα

πρέπει να διαιρεθεί με κάποιο τρόπο. Ο ADC έχει στο εσωτερικό του μία διάταξη διαίρεσης συχνότητας τον ADC Prescaler ο οποίος αποτελείται από σειρές διαδοχικά συνδεδεμένων D-FlipFlops και ένα πολυπλέκτη (MUX) όπως φαίνεται στην εικόνα 61.



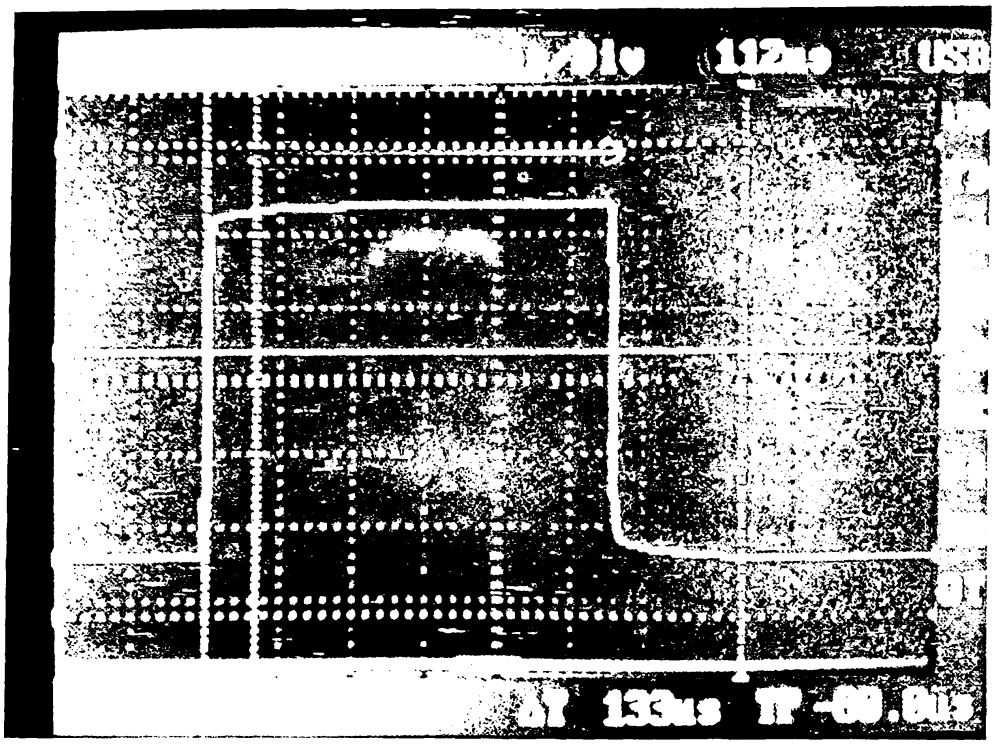
Εικόνα 61: Λειτουργία Prescaler

Ο ADC Prescaler έχει διάφορες επιλογές διαίρεσης της αρχικής συχνότητας F_{clk}, ξεκινώντας από $(\div 2)$ έως $(\div 128)$ όπως φαίνεται και στην εικόνα 59. Η συχνότητα που έχουμε επιλέξει για τους μικροελεγκτές είναι $F_{clk} = 16MHz$. Χρησιμοποιώντας την επιλογή $(F_{clk} \div 128)$ παίρνουμε συχνότητα F_{ADC}= 125 KHz. Η συχνότητα αυτή ικανοποιεί την απαίτηση του ADC για συχνότητα $50\,KHz < F_{ADC} < 200KHz$.

6.6.4 Διάρκεια της μετατροπής του ADC του συστήματος

Κάθε μετατροπή του ADC διαρκεί 14 κύκλους ρολογιού. Έτσι από την f_{ADC} βρίσκουμε την $T_{ADC} = 1/f_{ADC} = 8\,\mu sec$. Πολλαπλασιάζοντας την T_{ADC} με τους 13 κύκλους ρολογιού παίρνουμε $t_{conversion} = T_{ADC} \times 14 = 112\,\mu sec$. Έτσι η κάθε μετατροπή χρειάζεται $112\,\mu sec$ για να ολοκληρωθεί.

Αυτό ισχύει θεωρητικά . Για να δούμε αν όντως αυτό ισχύει και στην πράξη, γράφτηκε ένα μικρό πρόγραμμα το οποίο κάνει συνεχώς μετατροπές από ένα κανάλι του ADC. Κάθε φορά που η μετατροπή τελειώνει αλλάζει την κατάσταση ενός ακροδέκτη του μικροελεγκτή από λογικό "0" σε λογικό "1". Το αποτέλεσμα φαίνεται στον παλμογράφο εικόνα 62.



Εικόνα 62: Χρονος Μετατροπής ADC

Ο παλμός ισοδυναμεί με τον χρόνο μίας μετατροπής και όπως φαίνεται και στην εικόνα έχει διάρκεια 112 *μsec*. Όσο δηλαδή υπολογίστηκε και θεωρητικά.

Στον πρώτο μικροελεγκτή ο οποίος θα ελέγχει τους δώδεκα από τους 18 σέρβο-κινητήρες θα χρησιμοποιηθούν αντίστοιχα δώδεκα κανάλια του ADC (ADC0 – ADC11). Άρα ο χρόνος ολοκλήρωσης των 12 μετατροπών είναι:

$$t_{total\_1} = t_{conversion} \times 12 = 1.344\ msec$$

Όπου $t_{conversion}$ είναι ο χρόνος μίας μετατροπής.

Στον δεύτερο μικροελεγκτή ο οποίος θα αναλάβει τον έλεγχο των υπόλοιπων 5 από τους 18 κινητήρες θα χρησιμοποιηθούν αντίστοιχα 6 κανάλια του ADC (ADC0 – ADC5). Άρα ο χρόνος ολοκλήρωσης των 6 μετατροπών είναι:

$$t_{total\_2} = t_{conversion} \times 6 = 0.56\ msec\ (\text{ή}\ 560\ \mu sec)$$

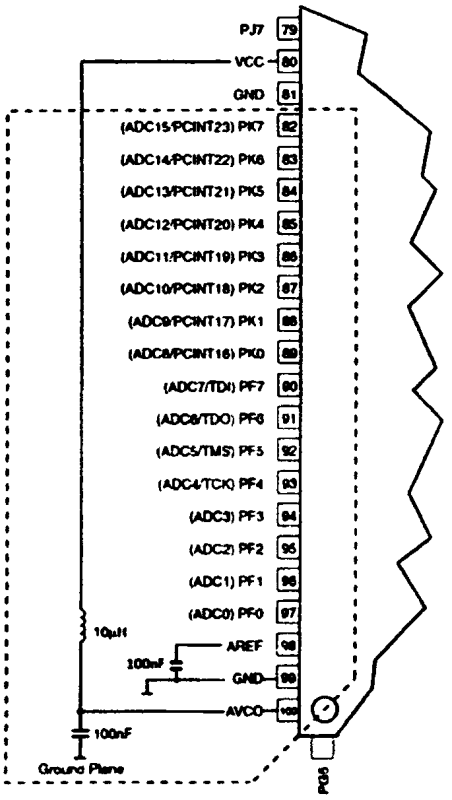
Αυτοί οι χρόνοι είναι πάρα πολύ μικροί, σε σχέση με την πραγματική απόκριση του συστήματος, για να δημιουργήσει κάποια καθυστέρηση στη γενικότερη λειτουργία του. Δηλαδή, αυτή η καθυστέρηση δεν επηρεάζει την απόκριση του συστήματος ως προς την κίνηση των σέρβο-κινητήρων.

6.7 Προγραμματισμός των ADC των μικροελεγκτών του συστήματος

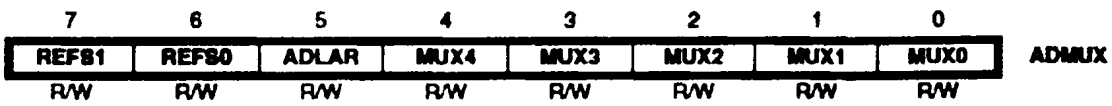
Για τον προγραμματισμό του ADC χρησιμοποιούμε έξι καταχωρητές 8-bit, τους ADMUX, ADCSRA, ADCSRB και SFIOR για τον έλεγχο της λειτουργίας του ADC και τους ADCH και ADCL για την αποθήκευση του αποτελέσματος της μετατροπής. Με τους καταχωρητές αυτούς μπορούμε να ελέγξουμε πλήρως τη λειτουργία του ADC.

Πριν κάνουμε οποιαδήποτε άλλη ενέργεια θα πρέπει να αρχικοποιήσουμε τον ADC επιλέγοντας την κατάλληλη τάση αναφοράς ,δηλαδή εδώ θέτουμε $V_{REF} = 5V$.

Όσον αφορά το hardware, ο ακροδέκτης AVCC που τροφοδοτεί τον ADC συνδέεται στα 5V μέσω ενός χαμηλοπερατού φίλτρου το οποίο δίνεται από τον κατασκευαστή για την απαλοιφή του θορύβου στην τροφοδοσία του ADC. Ο ακροδέκτης AREF στον οποίο ο χρήστης μπορεί να βάλει δική του τάση αναφοράς συνδέεται μέσω ενός πυκνωτή με τη γείωση. Ο πυκνωτής λειτουργεί ως φίλτρο για την αποκοπή θορύβου, όπως φαίνεται στην εικόνα 61. Όσων αφορά το software για να ορίσουμε στον ADC την $V_{REF} = 5V$ θα πρέπει να ρυθμίσουμε τα bits επιλογής της V_{REF}, REFS1 και REFS0 (Reference Selection bits1:0) που βρίσκονται στον καταχωρητή ADMUX όπως φαίνεται στην εικόνα 64.



Εικόνα 63: Συνδεση τροφοδοσίας ADC



Εικόνα 64: Καταχωρητής ADMUX

Στον πίνακα 7 δίνονται όλοι οι δυνατοί συνδυασμοί για την επιλογή της V_{REF} μέσω των bits REFS1:0

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Πίνακας 7: επιλογή τάσης αναφοράς

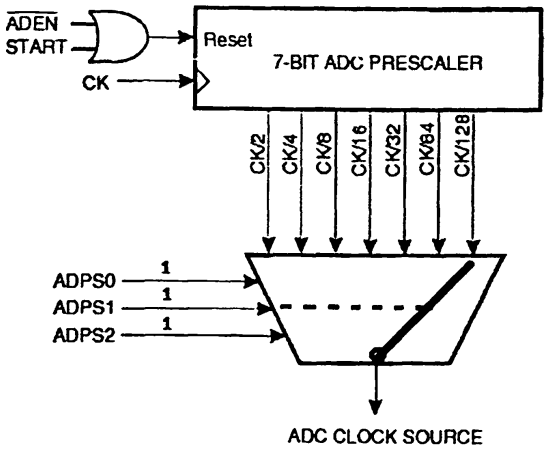
Εφόσον θέλουμε την $V_{REF} = 5V$ και η τροφοδοσία του ADC είναι AVCC = 5V επιλέγουμε $V_{REF} = AVCC$ κάνοντας τα τα bits REFS1 = "0" και REFS0 = "1".

Στη συνέχεια, μπορούμε να ορίσουμε τον διαιρέτη συχνότητας του ADC μέσω του ADC Prescaler, χρησιμοποιώντας τον καταχωρητή ADCSRA ρυθμίζοντας τα bits ADPS2 ADPS1 και ADPS0 (εικόνα 65).

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Εικόνα 65: Bits Ελέγχου Prescaler

Όπως αναφέρθηκε και στην ενότητα της συχνότητας δειγματοληψίας (6.5.3 - 2η Παράγραφος) για να παραχθεί για την λειτουργία του ADC συχνότητα $f_{ADC} = 125KHz$ από το ρολόι του μικροελεγκτή με $f_{clk} = 16MHz$ θα πρέπει να διαιρέσουμε την f_{clk} με την τιμή 128. Η επιλογή αυτή δίνεται από τον πολυπλέκτη του Prescaler αν μέσω του καταχωρητή λειτουργίας ADCSRA βάλουμε στο ADPS2 = "1", στο ADPS1 = "1" και στο ADPS0 = "1" όπως φαίνεται και στην εικόνα 66.

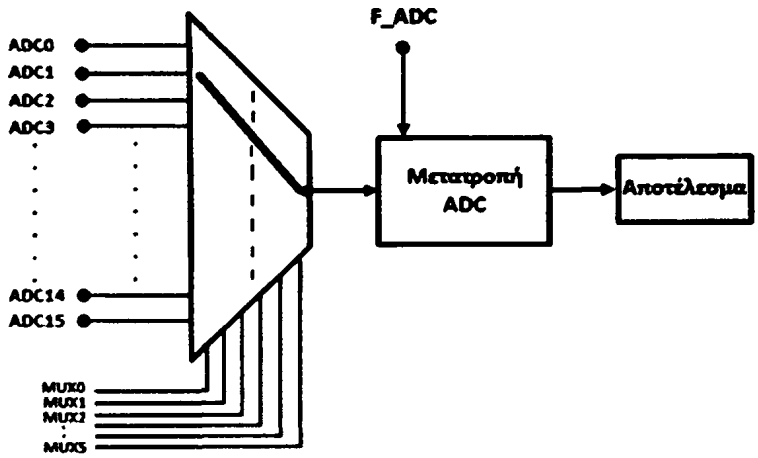


Εικόνα 66: Επιλογή Prescaler 128

Στη συνέχεια, γίνεται επιλογή του καναλιού στο οποίο θα γίνει η μετατροπή του αναλογικού σήματος σε ψηφιακό. Αυτό γίνεται μέσω των καταχωρητών ADCSRB και ADMUX ρυθμίζοντας τα bits MUX5 και MUX4 ,MUX3, MUX2, MUX1, MUX0 αντίστοιχα. Αυτά τα bits όπως γίνεται αντιληπτό είναι τα bits επιλογής του πολυπλέκτη καναλιών του ADC όπως φαίνεται και στις εικόνες 67 και 68.

7	6	5	4	3	2	1	0	
-	ACME	-	-	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
R	R/W	R	R	R/W	R/W	R/W	R/W	
7	6	5	4	3	2	1	0	
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Εικόνα 67: Bits επιλογής καναλιού για μετατροπή σήματος



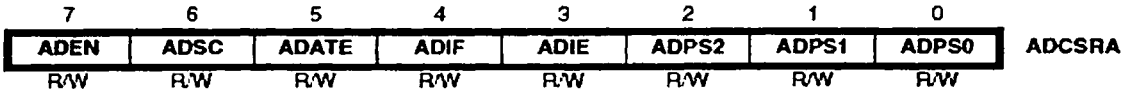
Εικόνα 68: Απλοποιημένο Διαγραμμα επιλογής καναλιών ADC

Έτσι, κάνοντας το bit MUX1 = "1" και όλα τα άλλα "0" επιλέγεται το δεύτερο κανάλι (ADC1) για μετατροπή κ.ο.κ.

Στον πρώτο μικροελεγκτή ο οποίος θα ελέγχει τους δώδεκα από τους 19 σέρβο-κινητήρες θα χρησιμοποιηθούν αντίστοιχα τα πρώτα δώδεκα κανάλια του ADC (ADC0 – ADC11) και στον δεύτερο μικροελεγκτή ο οποίος θα αναλάβει τον έλεγχο των υπόλοιπων 7 από τους 19 κινητήρες θα χρησιμοποιηθούν αντίστοιχα τα πρώτα 6 κανάλια του ADC (ADC0 – ADC5).

Τέλος, εφόσον έχουν πραγματοποιηθεί όλες οι προηγούμενες απαραίτητες ρυθμίσεις για την αρχικοποίηση του ADC, είναι δυνατό πλέον να ενεργοποιηθεί και να ξεκινήσει την μετατροπή.

Για την ενεργοποίηση του ADC ενεργοποιείται από τον χρήστη το bit ADEN (AD Enable) στον καταχωρητή ADCSRA (εικόνα 69), δηλαδή ADEN = "1". Αφού γίνει αυτό, για να ξεκινήσει η μετατροπή πρέπει να ενεργοποιήσουμε το bit ADSC (AD Start Conversion) που βρίσκεται στον ίδιο καταχωρητή. Δηλαδή ADSC = "1".



Εικόνα 69: Bits Ενεργοποίησης και ξεκινήματος ADC

Με το τέλος της μετατροπής ενεργοποιείται αυτόματα από το hardware το bit ADIF (AD Interrupt Flag). Δηλαδή γίνεται ADIF = "1"υποδεικνύοντας στο σύστημα ότι η μετατροπή ολοκληρώθηκε και ότι το αποτέλεσμα της μετατροπής είναι αποθηκευμένο στους 8-bit καταχωρητές ADCH και ADCL.

Να σημειωθεί το σήμα από το κάθε αισθητήριο κάμψης πηγαίνει σε ένα κανάλι του ADC και αφού γίνει η μετατροπή του καθενός σε ψηφιακό με την σειρά των βημάτων που αναφέρονται παραπάνω, το αποτέλεσμα αποθηκεύεται στους καταχωρητές ADCH και ADCL.

6.8 Συναρτήσεις μετατροπής των ψηφιακών σημάτων των αισθητήρων σε σήματα PWM

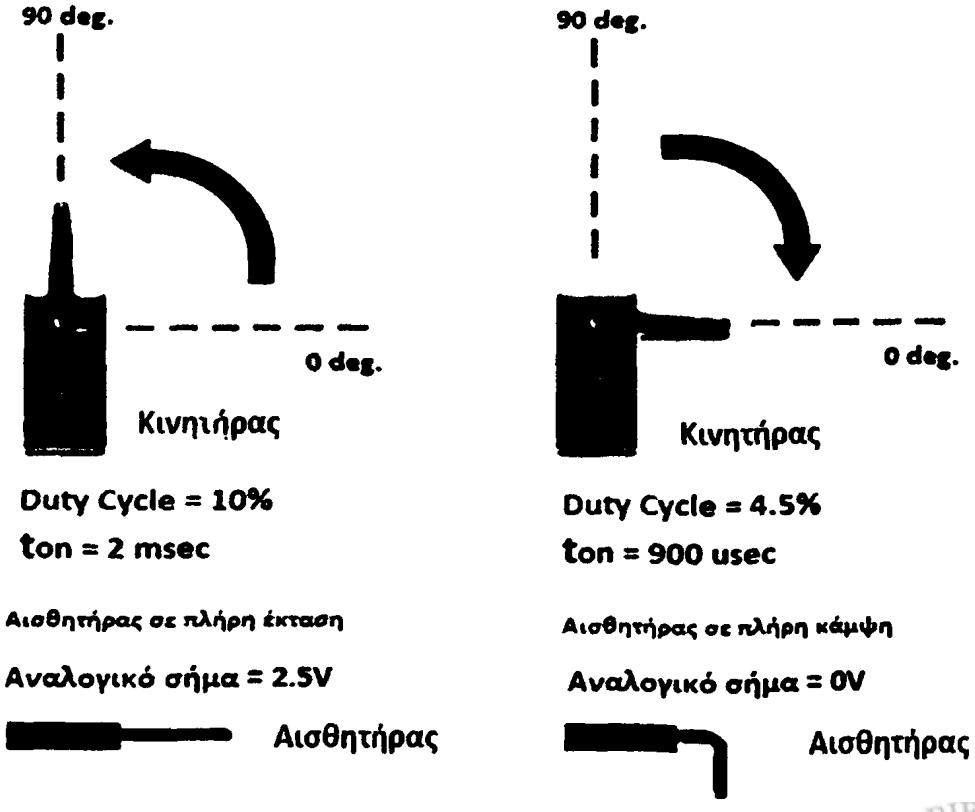
Στη συνέχεια, γίνεται επεξεργασία των ψηφιακών πλέον σημάτων των αισθητήρων κάμψης για την οδήγηση - έλεγχο των σέρβο-κινητήρων μέσω των καταχωρητών OCRnA/B/C των 16-bit χρονιστών Timer1/3/4/5.

Όπως αναφέρθηκε και στο 6^{ο} κεφάλαιο ο κάθε 16-bit χρονιστής περιέχει τρείς καταχωρητές σύγκρισης OCRnA/B/C οι οποίοι συγκρίνουν το περιεχόμενο τους, το οποίο έχει τοποθετηθεί και ανανεώνεται συνεχώς μέσω software, με αυτό του καταχωρητή TCNTn του χρονιστή καθώς αυτός αυξάνεται (n = 1,3,4,5). Μόλις οι τιμές αυτών των καταχωρητών εξισωθούν, οι έξοδοι/ακροδέκτες των χρονιστών του μικροελεγκτή αλλάζουν την κατάσταση τους από λογικό "1" σε λογικό "0" μέχρι ο χρονιστής να φτάσει στη μέγιστη τιμή του (TOP που ορίζεται από τον καταχωρητή ICRn) και να ξαναρχίσει από την αρχή (0 ή 0x0000). Μέσω αυτής της διαδικασίας που γίνεται συνεχόμενα παράγονται και τα σήματα PWM που οδηγούν τους σέρβο-κινητήρες.

Άρα, όπως γίνεται αντιληπτό, εάν εμείς αλλάζουμε τις τιμές των καταχωρητών OCRnA/B/C μέσω προγράμματος, κάθε φορά που οι χρονιστές κάνουν επανεκκίνηση της μέτρησης τους, τα σήματα PWM θα μεταβάλλονται και αυτά ανάλογα και τελικά οι σέρβο-κινητήρες θα εκτελούν κίνηση.

Οι τιμές που θα αποθηκεύονται στους καταχωρητές OCRnA/B/C είναι το αποτέλεσμα της επεξεργασίας των ψηφιακών σημάτων των αισθητήρων με μία σταθερά απροσήμαστου ακέραιου (unsigned integer). Αυτή η επεξεργασία γίνεται μέσω κάποιων απλών συναρτήσεων οι οποίες εξηγούνται στη συνέχεια.

Οι σέρβο-κινητήρες κινούνται από 0^o έως 90^o για t_{on} του PWM σήματος από 900μsec έως 2msec αντίστοιχα. Έτσι αποφασίστηκε οι αρθρώσεις των δαχτύλων της ρομποτικής χείρας να είναι σε πλήρη έκταση στις 90^o και σε πλήρη κάμψη στις 0^o. Δηλαδή οι άξονες των κινητήρων θα ευθυγραμμίζονται στις 90^o για $t_{on} = 2$msec ενώ, για $t_{on} = 900$μsec θα ευθυγραμμίζονται στις 0^o. Αυτό γίνεται καλύτερα κατανοητό στην εικόνα 70.



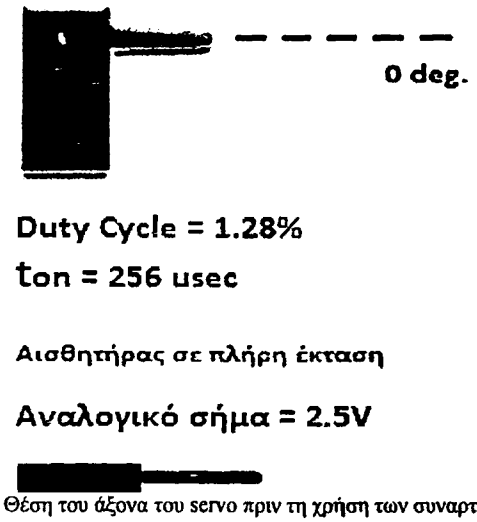
Εικόνα 70 : Συμπεριφορά κινητήρων με κάμψη αισθητήρων

Για να βρεθούν οι συναρτήσεις που περιγράφθηκαν προηγουμένως θα πρέπει πρώτα να βρεθούν τα τελικά σημεία αυτών σε σχέση με τις τελικές θέσεις των αξόνων των κινητήρων.

Γνωρίζοντας πλέον πως για να παραχθεί σήμα PWM με εύρος παλμού t_{on}=2msec, θα πρέπει να τοποθετηθεί στους καταχωρητές OCRnA/B/C η τιμή 3999 και για εύρος παλμού t_{on}=900μsec θα πρέπει να τοποθετηθεί στους καταχωρητές OCRnA/B/C η δεκαδική τιμή 1799 (όπως αποδείχθηκε στην ενότητα 6.4.1).

Παράλληλα όπως εξηγήθηκε και στο 3° κεφάλαιο, το επιθυμητό αναλογικό σήμα των αισθητήρων, όταν αυτοί βρίσκονται στη θέση ηρεμίας τους, είναι 2.5V ή αφού μετατραπούν σε ψηφιακό σήμα, 512 (έχοντας ως τάση αναφοράς τα 5V).

Έτσι, αν βάλουμε την τιμή αυτή στους καταχωρητές θα περιμέναμε οι άξονες των κινητήρων να ευθυγραμμίζονται στις 90° όπως φαίνεται και στην εικόνα 68. Αυτό όμως δεν μπορεί να συμβεί έτσι απλά εφόσον η ψηφιακή τιμή που πρέπει να αποθηκευθεί στους OCRnA/B/C για ευθυγράμμιση των αξόνων είναι 3999. Αυτό που πραγματικά συμβαίνει βάζοντας την τιμή 512 στους OCRnA/B/C φαίνεται στην εικόνα 71.



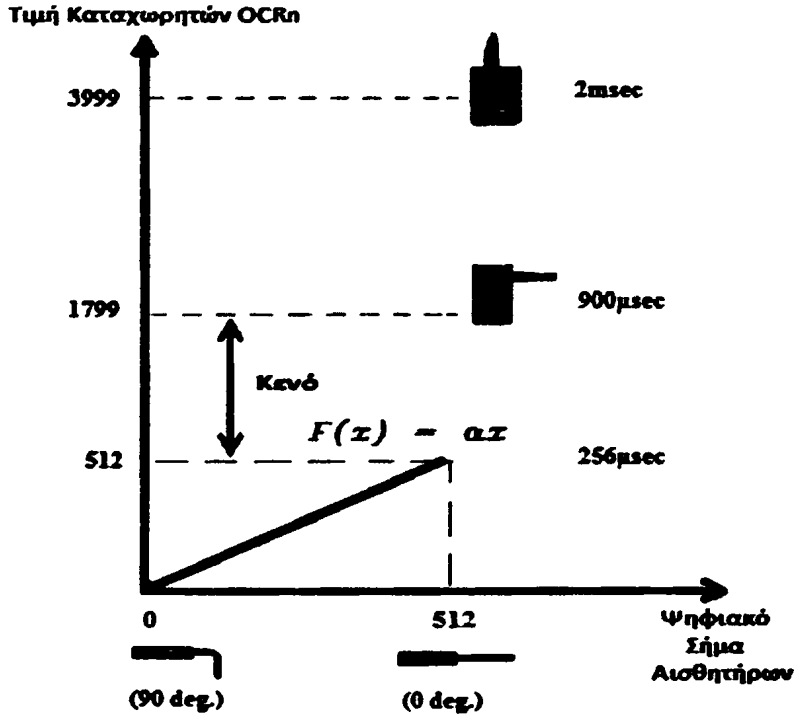
Εικόνα 71: Θέση του άξονα του servo πριν τη χρήση των συναρτήσεων

Αυτό συμβαίνει, γιατί βάζοντας την τιμή 512 στους καταχωρητές OCRnA/B/C δημιουργείται ένα σήμα PWM με πλάτος παλμού t_{on} = 256μsec το οποίο είναι πολύ μικρότερο από τα 900μsec, αυτό δηλαδή που μπορεί να διαχειριστεί ο κινητήρας ως κατώτατη τιμή. Έτσι ο άξονας του σέρβο-κινητήρα κολλάει στις 0°.

Αντίστοιχα εάν τοποθετήσουμε τον αισθητήρα σε πλήρη κάμψη η τιμή που αποθηκεύεται στους OCRnA/B/C είναι 0, το σήμα PWM έχει πλάτος παλμού t_{on} = 0

και ουσιαστικά το σήμα PWM εξαλείφεται . Και σε αυτή την περίπτωση ο άξονας του κινητήρα μένει κολλημένος στις 0 μοίρες .

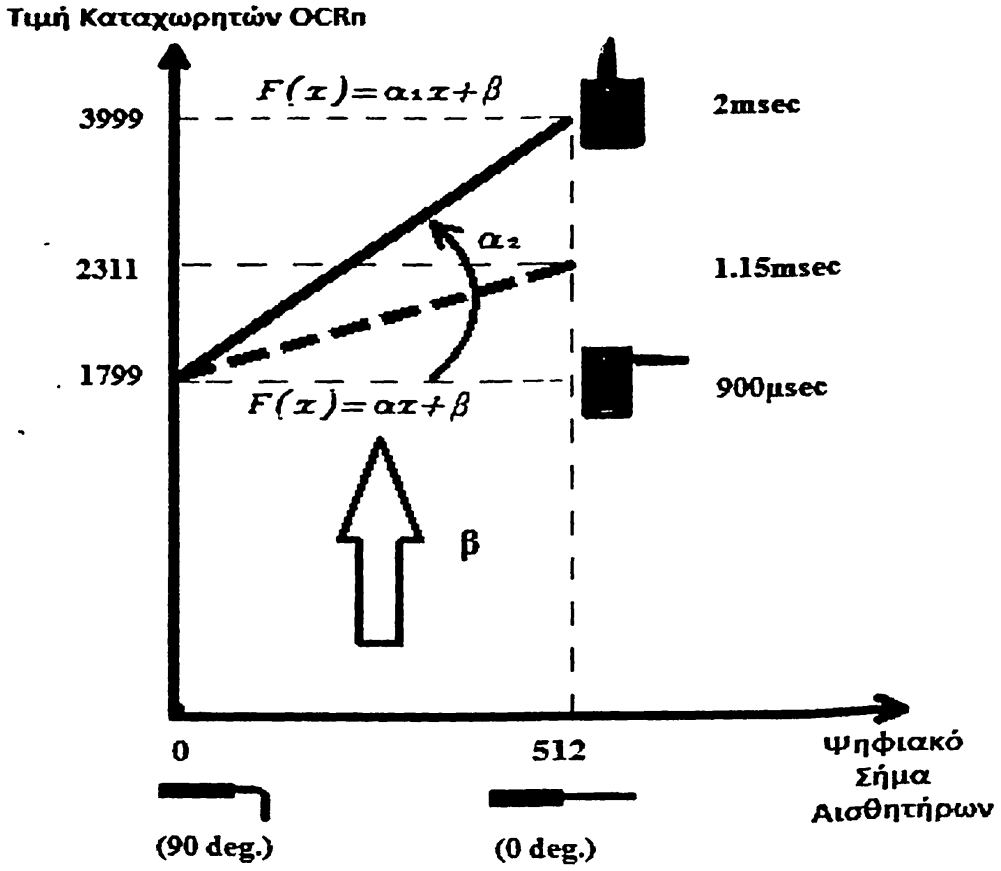
Η συνάρτηση που περιγράφθηκε τώρα και η οποία είναι λανθασμένη (είναι όμως πολύ σημαντική για την κατανόηση της σωστής συνάρτησης που πρόκειται να κατασκευαστεί) παρουσιάζεται γραφικά στη παρακάτω εικόνα (εικόνα 72).



Εικόνα 72: Μεταβολή ψηφιακού σήματος με την κάμψη του αισθητήρα χωρίς χρήση συνάρτησης

Όπως γίνεται φανερό, προκύπτει ένα κενό ανάμεσα στις τιμές που χρειάζονται οι σέρβο-κινητήρες για να γυρίσουν τους άξονες τους και στις τιμές που χρησιμοποιήθηκαν μέσω της συνάρτησης (κόκκινη γραμμή), βάσει της κάμψης των αισθητήρων.

Για να διορθωθεί αυτό το σφάλμα, αρχικά, η συνάρτηση του γραφήματος (εικόνα 72), $f(x) = \alpha x$, όπου $\alpha = 1$, πρέπει να αντικατασταθεί με μία συνάρτηση της μορφής $f(x) = \alpha x + \beta$, όπου το β θα είναι ίσο με την πρώτη τιμή που δέχεται ο σέρβο-κινητήρας, δηλαδή $\beta = 1799$ (εικόνα 73). Επειδή όμως η τελική τιμή της νέας συνάρτησης (διακεκομμένη κόκκινη γραμμή) για πλήρη κάμψη των αισθητήρων δεν φτάνει την τιμή 3999 αλλά την τιμή 2311 που αντιστοιχεί σε πλάτος παλμού 1.15 msec πρέπει να αλλαχθεί και η κλήση της συνάρτησης. Οπότε η τελική συνάρτηση παίρνει τη μορφή $f(x) = \alpha_1 x + \beta$. Στην εικόνα 73 παρουσιάζεται η διαδικασία εύρεσης της σωστής συνάρτησης όπως περιγράφθηκε παραπάνω.



Εικόνα 73: Κίνηση άξονα μετά τη χρήση της συνάρτησης

Για την εύρεση της κλίσης της συνάρτησης χρησιμοποιούμε τον γνωστό τύπο:

$$a_1 = \frac{Y_{τελικο} - Y_{αρχικο}}{X_{τελικο} - X_{αρχικο}}$$

Όπου $Y_{τελικο} = 3999,\ Y_{αρχικο} = \beta = 1799$

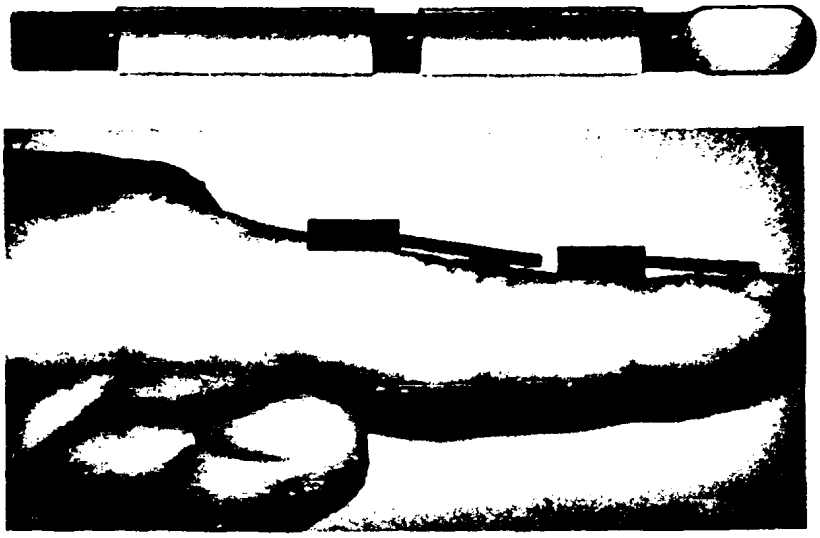
και $X_{τελικο} = 512,\ X_{αρχικο} = 0$

και προκύπτει ότι η κλίση είναι $a_1 = 4.3$

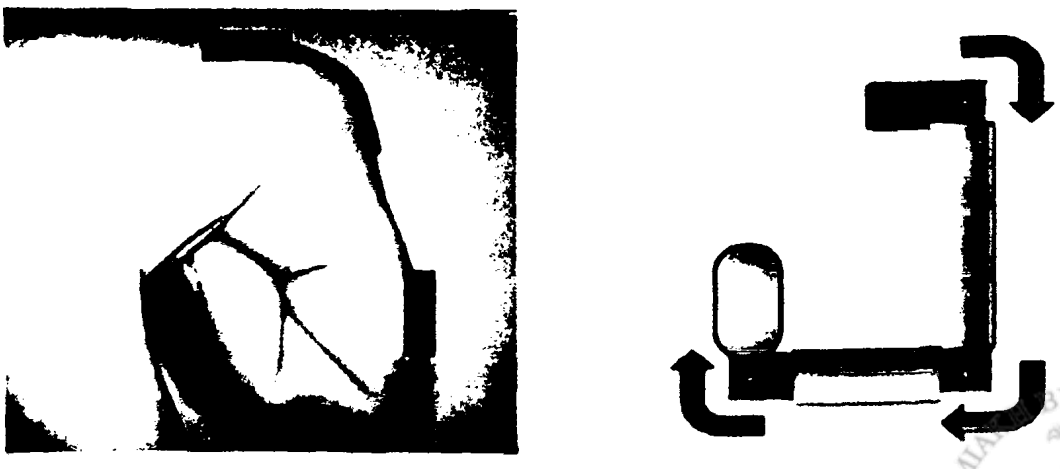
Έτσι η συνάρτηση η οποία ενώνει τα σήματα των αισθητήρων με την κίνηση των σέρβο-κινητήρων είναι η $f(x) = 4.3x + 1799$, ***όπου x το ψηφιακό σήμα ενός αισθητήρα το οποίο κυμαίνεται μεταξύ 0 (πλήρης κάμψη) και 512 (θέση ηρεμίας), και*** $f(x)$ ***η τιμή που παίρνει ο καταχωρητής OCRnA/B/C η οποία κυμαίνεται μεταξύ 1799(900 μsec) και 3999 (2 msec) ώστε να κινηθεί ο άξονας του κινητήρα στην επιθυμητή θέση.*** Η τιμή β = 1799 και α_1 = 4.3 στην πράξη, ενδέχεται μεταβάλλεται για κάθε αισθητήριο καθώς όπως αναφέρθηκε στην ενότητα 6.5.2 τα

αισθητήρια στην πλήρη τους έκταση μπορεί να μην δίνουν σήμα ακριβώς 2.5V αλλά περισσότερο η λιγότερο από αυτό. Έτσι η τιμή των β και α_1 της συνάρτησης στο πρόγραμμα θα προσαρμόζεται στην κάθε συνάρτηση ανάλογα με τη διαφορά από την επιθυμητή θέση. Αυτές οι αλλαγές έγιναν μέσω δοκιμών πάνω στη ρομποτική χείρα.

Έτσι λοιπόν, η λειτουργία ενός από τα δάχτυλα της ρομποτικής χείρας χρησιμοποιώντας την παραπάνω συνάρτηση τρείς φορές, μία για κάθε αισθητήρα κάμψης έχει σαν αποτέλεσμα την αντιγραφή των κινήσεων του δαχτύλου του χρήστη, ο οποίος φοράει και την διάταξη αισθητήρων (γάντι) στο χέρι του, όπως φαίνεται στην εικόνα 74.

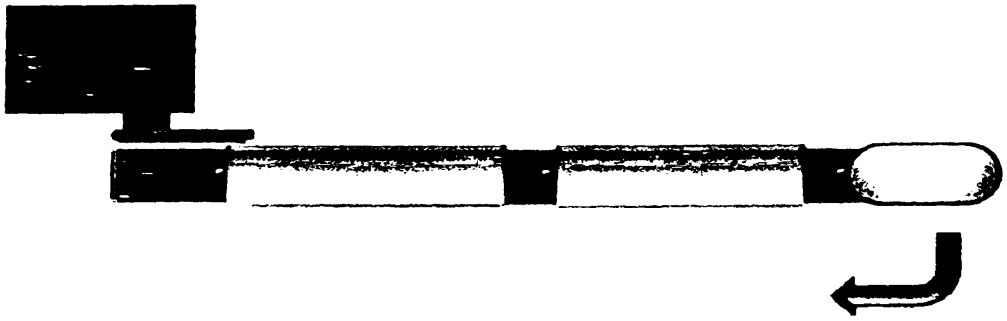


Εικόνα 74α: Αντιγραφή κινήσεων μέσω αισθητήρων



Εικόνα 74β: Κάμψη των αισθητήρων ισοδυναμεί με κίνηση των αξόνων των servo

Εκτός από τις συναρτήσεις της κάμψης και έκτασης των ρομποτικών δαχτύλων υπάρχουν και οι συναρτήσεις για τους κινητήρες πλάγιας κίνησης των δαχτύλων. Κάτω από κάθε δάχτυλο τοποθετήθηκε ένας σέρβο-κινητήρας ο οποίος αναλαμβάνει την κίνηση ολόκληρου του δαχτύλου δεξιά και αριστερά από τον άξονα ηρεμίας του (εικόνες 75α πλάγια όψη, και 75β κάτοψη).

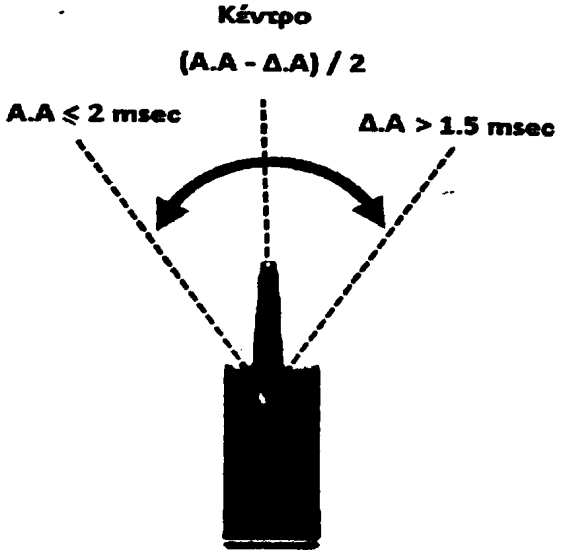


Εικόνα 75α: Τοποθέτηση κινητήρα πλάγιας κίνησης στο πίσω μέρος του δαχτύλου

Εικόνα 75β: Πλάγια κίνηση δαχτύλου

Για την κίνηση αυτή των ρομποτικών δαχτύλων χρησιμοποιήθηκαν τέσσερεις σέρβο-κινητήρες ροπής 2.2kg*cm ένας για κάθε δάχτυλο.

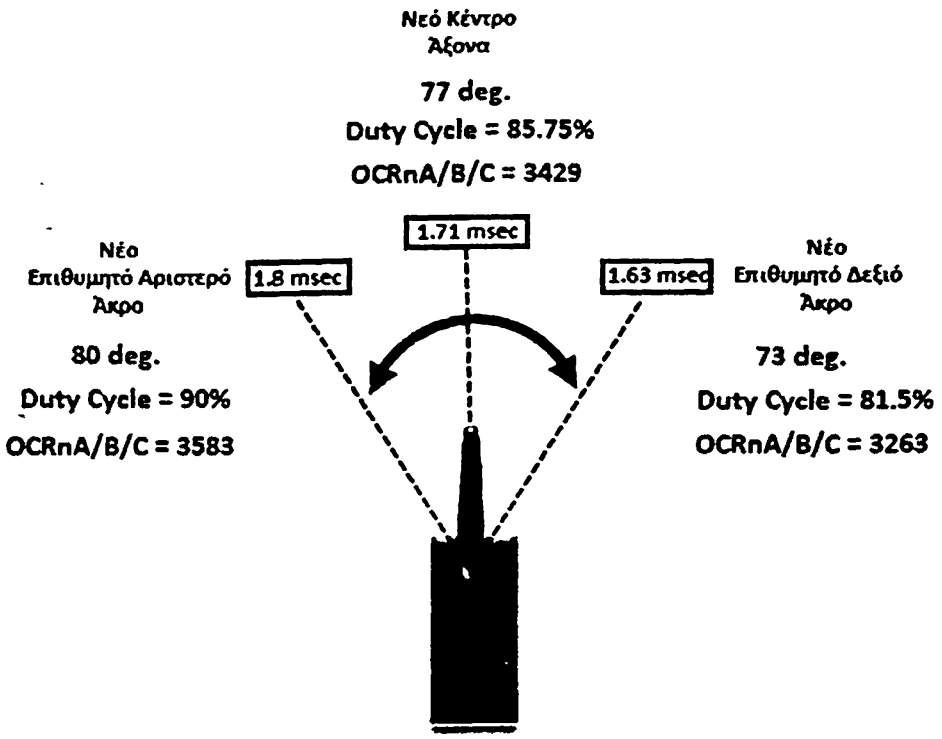
Για να είναι σε θέση τα δάχτυλα της ρομποτικής χείρας να γυρίζουν πλάγια (δεξιά-αριστερά) όπως φαίνεται στην εικόνα 75β, έπρεπε η τιμή ηρεμίας (το κέντρο) των αξόνων των νέων σέρβο-κινητήρων που τοποθετήθηκε να είναι μετατοπισμένο λίγο πιο δεξιά για σήμα PWM με t_{on} = 2 msec. Δηλαδή μέχρι τώρα ήταν επιθυμητό οι άξονες των servos να παίρνουν τις τελικές τους θέσεις όπως απεικονίζονται στην εικόνα 70. Τώρα όμως για την τοποθέτηση των τεσσάρων τελευταίων σέρβο-κινητήρων είναι επιθυμητό οι άξονες να είναι τοποθετημένοι όπως φαίνεται στην επόμενη εικόνα (εικόνα 76α) .



Εικόνα 76α: Μετατροπή άξονωντ των servo πλάγιας κίνησης των δαχτύλων

Τα ζητούμενα εδώ είναι το νέο κέντρο των αξόνων και οι επιθυμητές μοίρες απόκλισης από το νέο κέντρο. Δηλαδή πόσες μοίρες δεξιά - αριστερά θα γυριζει ο τέταρτος κινητήρας του ρομποτικών δαχτύλων από το νέο κέντρο του. Το νεο αριστερό άκρο (Α.Α) θα μπορεί να πάρει τιμές $t_{on} \leq 2$ msec καθώς $t_{on} = 2$ msec είναι η μέγιστη τιμή που καταλαβαίνει ο σέρβο-κινητήρας. Το νέο δεξιό άκρο (Δ.Α) θα μπορεί να πάρει τιμή ανάλογα με το πόσες θα είναι οι επιθυμητές μοίρες απόκλισης από το νέο κέντρο. Επειδή οι πλάγιες κινήσεις των δαχτύλων του ανθρώπινου χεριού δεν ξεπερνούν τις 45^{o} πλάγιας κίνησης, αποφασίσθηκε και οι άξονες των σερβοκινητήρων από το νέο κέντρο να μην κινούνται περισότερο από 45^{o}, δηλαδή το δεξί άκρο να μην πηγαίνει πιο κάτω από t_{on} = 1.5 msec.Το νέο κέντρο θα είναι η μέση τιμή των δύο νέων ακραίων τιμών (εικόνα 76α).

Μετά από δοκιμές των νέων αυτών θέσεων που έγιναν με τα ρομποτικά δάχτυλα, βρέθηκαν ως καταλληλότερες θέσεις νεόυ κέντρου και νέων ακραίων σημείων αυτά που φαίνονται στην εικόνα 76β . Εδώ πρέπει να σημειωθεί ότι το D.C. των θέσεων καθώς και οι μοίρες των κλίσεων που φαινονται στην εικόνα είναι μετρημένα, όπως γίνεται κατανοητό,ως προς την κατώτατη θέση που μπορεί να στραφεί ο άξονας του κινητήρα δηλαδή τα 900 μsec – 1 msec (η διαφορά της κίνησης για αυτό το έυρος είναι πάρα πολύ μικρή, σχεδόν μη ορατή στην κίνηση των αξόνων και επαληθεύθηκε μέσω δοκιμών).



Εικόνα 76β: Επιθυμητά χρονικα όρια για servos πλάγιας κίνησης

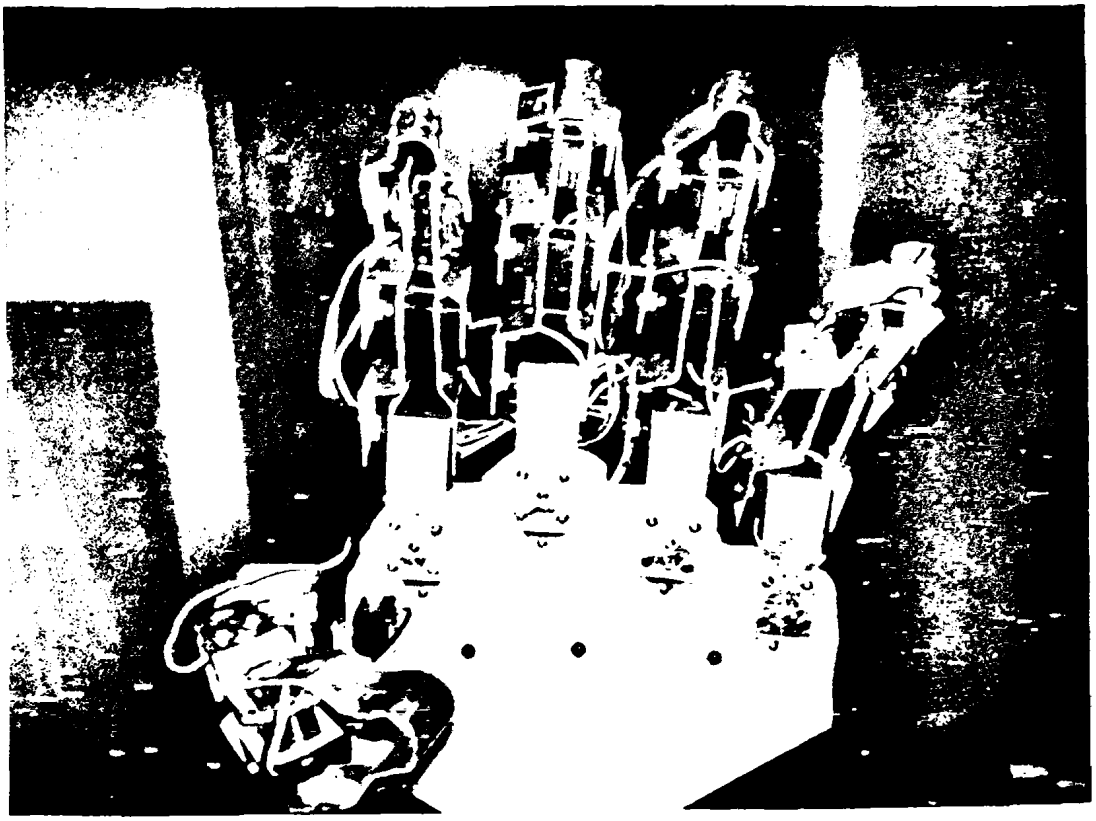
Έτσι λοιπόν, η κίνηση των ρομποτικών δαχτύλων θα κάνει μικρές κινήσεις δεξιά αριστερά με απόκλιση από το νέο κέντρο κατά 3.5^{o} περίπου. Κατά τον προγραμματισμό της κίνησης των κινητήρων αυτά τα μεγέθη μπορεί να μεταβληθούν λίγο γιατί σε ορισμένες περιπτώσεις τα αισθητήρια κάμψης ενδέχεται να μεταβάλλουν ελαφρώς το σήμα τους από το επιθυμητό κατά την τοποθέτηση του γαντιού από τον χρήστη.

Οι νέες συναρτήσεις σχεδιάσθηκαν με τον ίδιο ακριβώς τρόπο όπως έγινε και για τις προηγούμενες με τη διαφορά ότι εδώ τα αισθητήρια κάμψης δύο ιντσών έχουν ήδη καμφθεί από πριν (πίνακας 1 & εικόνα 36, 3^{o} Κεφ.) και το σήμα ηρεμίας είναι 2.5 V για αυτή τη θέση. Έτσι, όταν τα δάχτυλα του χρήστη ανοίγουν μεταξύ τους οι αντιστάσεις των αισθητήρων δύο ιντσών μικραίνουν και το σήμα γίνεται > 2.5V και έτσι κινούνται δεξιά- αριστερά τα δάχτυλα της ρομποτικής χείρας. Το αν θα κινούνται δεξιά αριστερά οι άξονες των κινητήρων εξαρτάται από τα πρόσημα που έχουμε βάλει μέσα στις συναρτήσεις. Για παράδειγμα, για την δεξιά κίνηση του δείκτη η συνάρτηση είναι η εξής: $f(x) = \text{ADC} + 3429$ ενώ του παράμεσου δαχτύλου για την αριστερή κίνηση η συνάρτηση είναι $f(x) = 3429 - \text{ADC}$, όπου x = ADC = ψηφιακό σήμα.

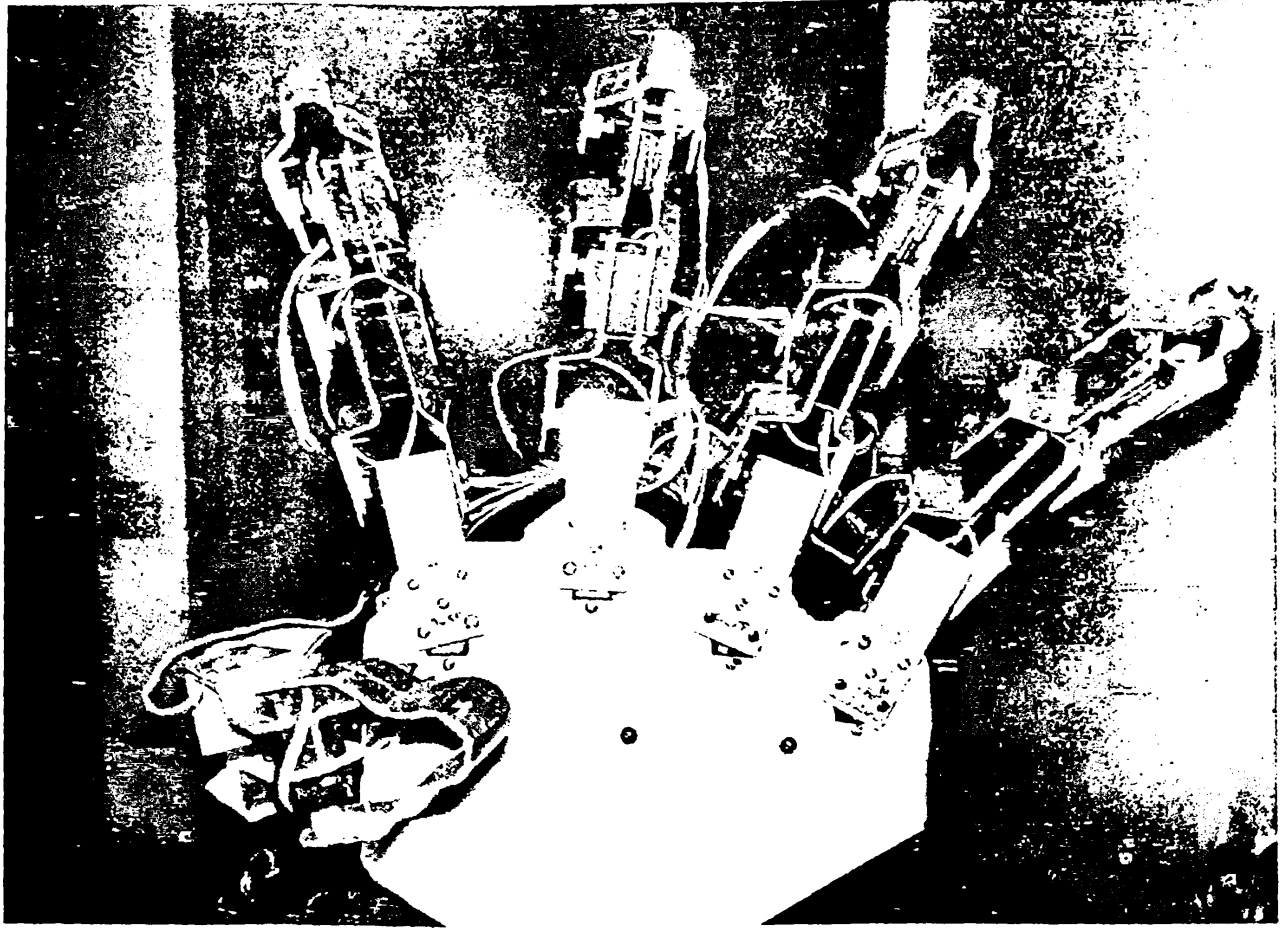
Όπως γίνεται φανερό, ο συντελεστής **α** στις συναρτήσεις αυτές είναι ίσος με μονάδα **(α=1)** και όχι 4.3 όπως ήταν στις προηγούμενες συναρτήσεις. Ο λόγος είναι ότι οι επιθυμητές πλάγιες κινήσεις είναι πάρα πολύ μικρές. Εάν βάζαμε τιμή στον συντελεστή **α > 1** (α= 2,3,4), στις συναρτήσεις τα ρομποτικά δάχτυλα θα χτυπούσαν μεταξύ τους με την παραμικρή κίνηση καθώς το ψηφιακό σήμα το οποίο λαμβάνεται από τα αισθητήρια θα ενισχύονταν πολύ.

Το αποτέλεσμα θα ήταν τα ρομποτικά δάχτυλα να σπάσουν και οι κινητήρες της πλάγιας κίνησης να καούν από το μεγάλο φορτίο που θα δεχόντουσαν.

Στην εικόνα 77 φαίνονται τα δάχτυλα της ρομποτικής χείρας πριν (77α) και μετά (77β) την εκτέλεση των πλαγίων κινήσεων.



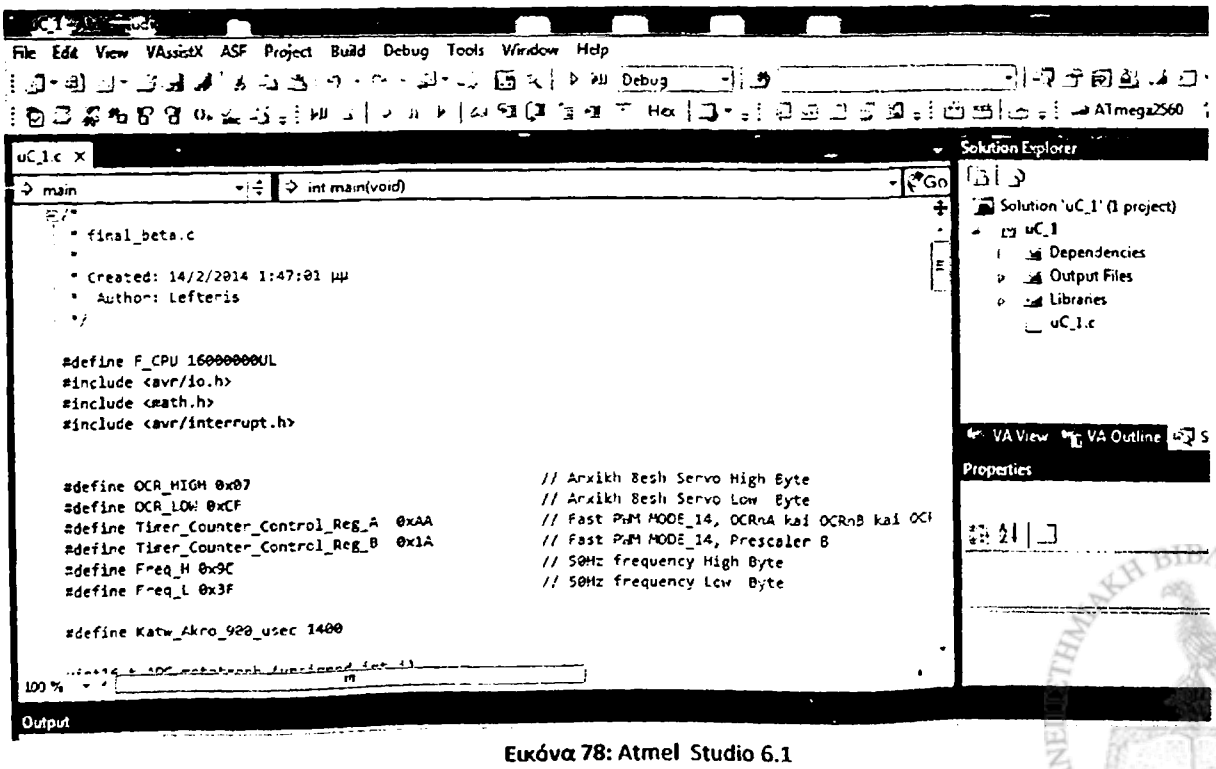
Εικόνα 77α: Δάχτυλα σε θέση Ισορροπίας



Εικόνα 77β: Δάχτυλα σε πλάγια κίνηση

6.9 Προγραμματισμός Συστήματος

Σε αυτό το σημείο θα αναλυθεί ο τρόπος με τον οποίο προγραμματίστηκαν οι δύο μικροελεγκτές. Για την υλοποίηση και μεταφορά του κώδικα στους δύο μικροελεγκτές χρησιμοποιήθηκε το πρόγραμμα Atmel Studio 6.1 (εικόνα 78).



Εικόνα 78: Atmel Studio 6.1

Για τον προγραμματισμό των μικροελεγκτών ήταν απαραίτητη η αρχικοποίηση των καταχωρητών των χρονιστών που είναι υπεύθυνοι για την παραγωγή σημάτων PWM.

Όπως αναφέρθηκε και στην παράγραφο 6.5.1, για να λειτουργήσουν οι χρονιστές των μικροελεγκτών ως γεννήτριες σημάτων PWM πρέπει οι καταχωρητές TCCRnA και TCCRnB να πάρουν τις ακόλουθες τιμές:

- **TCCRnA = 0xAA**; // Fast PWM Mode 14, Ενεργοποίηση OCRnA/B/C.
- **TCCRnB = 0x1A**; //Fast PWM Mode 14, Prescaler = **8**.

Στη συνέχεια για την παραγωγή της επιθυμητής συχνότητας των PWM σημάτων (50Hz), τοποθετούμε στον καταχωρητή ICRn την ακόλουθη τιμή:

- **ICRn = 0x9C3F** → Συχνότητα 50Hz.

Σε αυτό το σημείο, εφόσον οι χρονιστές και των δύο μικροελεγκτών έχουν αρχικοποιηθεί με τον τρόπο που περιγράφτηκε, αρχικοποιούμε τους ADC των μικροελεγκτών. Όπως αναλύθηκε και στις παραγράφους 6.6.2, 6.6.3 και 6.7, το πρώτο βήμα είναι η επιλογή της τάσης αναφοράς (V_{ref}) η ρύθμιση του Prescaler και η ενεργοποίηση των ADC. Οι ρυθμίσεις αυτές γίνονται μέσω των καταχωρητών ADMUX και ADCSRA τοποθετώντας τις ακόλουθες τιμές:

- **ADMUX = 0x40;** // Τάση Αναφοράς $V_{ref} = AV_{cc} = 5V$
- **ADCSRA = 0x87**; // ADC Prescaler = 128, Ενεργοποίηση ADC

Στη συνέχεια, δημιουργούμε μία συνάρτηση μέσω της οποίας επιλέγουμε το κανάλι (είσοδο) του ADC, στο οποίο πρόκειται να γίνει η μετατροπή αναλογικού σήματος σε ψηφιακό. Μέσω αυτής της συνάρτησης, παίρνουμε ως αποτέλεσμα το ψηφιακό σήμα του καναλιού που μετατρέπεται κάθε φορά.

Η συνάρτηση αυτή ονομάζεται **uint16_t ADC_metatroph (unsigned int i)** και δέχεται σαν όρισμα το κανάλι i (όπου i= 0,1,2,...,7) που επιλέγουμε για μία μετατροπή και επιστρέφει την ψηφιακή μορφή του αναλογικού σήματος τάσης του καναλιού που επιλέχθηκε. Για την επιλογή του καναλιού του ADC χρησιμοποιούμε τα bits MUX4/3/2/1/0 που βρίσκονται στον καταχωρητή ADMUX και το bit MUX5 που βρίσκεται στον καταχωρητή ADCSRB. Για να γίνει αυτό προσθέτουμε στον καταχωρητή ADMUX το όρισμα (i) χωρίς να πειράξουμε τα υπόλοιπα bits του καταχωρητή (τα οποία ρυθμίσαμε πιο πριν).

- **ADMUX = 0x40 + i;** // όπου i=0,1,...,7

Στη συνέχεια, για να ξεκινήσουμε την μετατροπή του καναλιού που επιλέχθηκε, κάνουμε το bit ADSC = "1", το οποίο βρίσκεται στον καταχωρητή ADCSRA. Για να γίνει αυτό χρησιμοποιούμε λογική πράξη OR (|), αφήνοντας έτσι τα υπόλοιπα bits του καταχωρητή όπως ήταν.

- **ADCSRA | = (1<<ADSC);**

Το επόμενο βήμα είναι ο έλεγχος της ολοκλήρωσης της μετατροπής. Αυτό γίνεται ελέγχοντας συνεχώς το bit ADIF που βρίσκεται στον καταχωρητή ADCSRA. Όσο το bit αυτό είναι "0" η μετατροπή του δεν έχει τελειώσει ακόμη και το πρόγραμμα παραμένει εκεί μέχρι αυτό να συμβεί. Μόλις γίνει "1" σημαίνει πως η μετατροπή έχει ολοκληρωθεί και το πρόγραμμα συνεχίζει με την εκτέλεση των επόμενων εντολών. Έτσι, ελέγχουμε μέσω μίας συνθήκης (while) πότε το bit αυτό θα γίνει "1".

- **while ((ADCSRA & (1<<ADIF)) == 0);**

Στη συνέχεια, παίρνουμε το αποτέλεσμα της ψηφιακής μετατροπής, το οποίο είναι αποθηκευμένο στους καταχωρητές ADCL και ADCH και το τοποθετούμε σε μία μεταβλητή μήκους 2 bytes, με όνομα adc_result, χρησιμοποιώντας τις παρακάτω εντολές:

- **adc_low_byte = ADCL;** // H adc_low_byte κρατάει προσωρινά το low
  // byte του αποτελέσματος της μετατροπής
- **adc_result = ADCH<<8 | adc_low_byte;**
- **return adc_result;**

Εδώ πρέπει να σημειωθεί πως χρησιμοποιούνται δύο ίδιες συναρτήσεις μετατροπής, μία για κάθε μικροελεγκτή καθώς, χρησιμοποιούνται για τον ίδιο ακριβώς λόγο και με τον ίδιο τρόπο. Έχοντας τελειώσει με τις αρχικοποιήσεις των χρονιστών και των ADC καθώς και με τη δημιουργία της συνάρτησης ADC_metatroph(i), μπορούμε πλέον να μετατρέψουμε τα αναλογικά σήματα των αισθητήρων που βρίσκονται στα κανάλια των ADC των μικροελεγκτών. Αυτό γίνεται επιλέγοντας τα επιθυμητό κανάλι κάθε φορά με κλήση της συνάρτησης ADC_metatroph(i) και την τοποθέτηση του αποτελέσματος της μετατροπής σε μία μεταβλητή με όνομα adc_result που ορίζουμε εμείς από το πρόγραμμα. Στο πρόγραμμα του πρώτου μικροελεγκτή η μετατροπή των καναλιών γίνεται με τη σειρά για τα 12 κανάλια που χρησιμοποιήθηκαν με τη χρήση δύο βρόχων for (for-loop), ξεκινώντας από το κανάλι ADC0 μέχρι το ADC7 (με bit MUX5 = 0) και από το κανάλι ADC8 μέχρι το ADC11 (με bit MUX5 = 1) αντίστοιχα. Τα 12 αυτά κανάλια

αντιστοιχούν στα αναλογικά σήματα που παράγονται από τους αισθητήρες κάμψης που βρίσκονται στις 12 αρθρώσεις των Δείκτη, Μέσου, Παράμεσου και Μικρού δαχτύλου όπως φαίνονται στη εικόνα 29 (σελ.53). Κάθε φορά που μία μετατροπή ολοκληρώνεται, το ψηφιακό σήμα αποθηκεύεται στην μεταβλητή adc_result. Αυτή η μεταβλητή τοποθετείται ως όρισμα στην αντίστοιχη συνάρτηση μετατροπής ψηφιακού σήματος σε παλμό PWM (όπως αναλύθηκαν στην παράγραφο 6.8). Ένα παράδειγμα της διαδικασίας που ακολουθήθηκε φαίνεται στο επόμενο κομμάτι κώδικα. Η διαδικασία αυτή είναι ίδια και για τα 12 κανάλια του ADC, του 1ου μικροελεγκτή.

```
for ( i=0; i<8; i++)
{
        adc_result =  ADC_metatroph(i);
        PRAXI_DIGITAL = (adc_result*4.3) + Katw_Akro_900_usec;
        *x[i] = (uint16_t) PRAXI_DIGITAL;
}
```

Στην πρώτη εντολή μέσα στη for επιλέγεται το κανάλι στο οποίο θα γίνει η μετατροπή ADC και το αποτέλεσμα αποθηκεύεται στην μεταβλητή adc_result.

Η δεύτερη εντολή είναι η συνάρτηση μετατροπής ψηφιακού σήματος σε σήμα PWM (παράγραφος 6.8). Η σταθερά **Katw_Akro_900_usec** αντιστοιχεί στο β της συνάρτησης $f(x)= a_1 x+\beta$ (όπως αυτή εξηγήθηκε στην παράγραφο 6.8), ο αριθμός **4.3** αντιστοιχεί στο a_1 της συνάρτησης και το **adc_result** αντιστοιχεί στη μεταβλητή x της συνάρτησης. Η μεταβλητή **PRAXI_DIGITAL** είναι αυτή στην οποία αποθηκεύεται το αποτέλεσμα της συνάρτησης.

Η τρίτη εντολή στέλνει την τιμή της μεταβλητής **PRAXI_DIGITAL** σε ένα συγκεκριμένο στοιχείο του πίνακα **x[i]** κάθε φορά. Ο πίνακας **x[i]** αποτελείται από τους καταχωρητές OCRnA/B/C οι οποίοι παράγουν και τα διαφορετικά σήματα PWM. Ο πίνακας αυτός αποτελείται από τους εξής καταχωρητές OCRnA/B/C.

```
unsigned int* x[]={&OCR1A, &OCR1B, &OCR1C, &OCR3A, &OCR3B,
                   &OCR3C, &OCR4A, &OCR4B } ;
```

Έτσι λοιπόν, ο Αισθητήρας **A1** ελέγχει τον καταχωρητή **OCR1A,** ο Αισθητήρας **A2** ελέγχει τον καταχωρητή **OCR1B,** κοκ. Συνολικά τοποθετήθηκαν δύο τέτοιοι πίνακες στο πρόγραμμα του 1ου μικροελεγκτή για τον έλεγχο όλων των καταχωρητών OCRnA/B/C/ (Παράρτημα Π1)

Στο πρόγραμμα του δεύτερου μικροελεγκτή η μετατροπή των καναλιών γίνεται με παρόμοιο τρόπο, μόνο που εδώ η συνάρτηση της μετατροπής καλείται μέσω των διακοπών των χρονιστών που παράγουν τα PWM σήματα. Κάθε φορά που ένας από τους καταχωρητές OCRnA/B/C εξισωθεί με τον καταχωρητή μέτρησης TCNTn, παράγεται και από μία διακοπή. Μέσα σε κάθε διακοπή, ανάλογα με τον OCRnA/B/C από τον οποίο και παράγεται , καλείται η συνάρτηση μετατροπής για ένα συγκεκριμένο κανάλι του δεύτερου μικροελεγκτή το οποίο αντιστοιχεί στον αισθητήρα που ελέγχει ένα από τους κινητήρες της ρομποτικής χείρας (εικόνα 79).

Επίσης δεν υπάρχουν βρόχοι for και η αποθήκευση των τιμών στους καταχωρητές OCRnA/B/C γίνεται άμεσα χωρίς χρήση πινάκων (όπως στο πρόγραμμα του 1^{ου} μικροελεγκτή). Ο δεύτερος μικροελεγκτής ελέγχει τα αναλογικά σήματα των αισθητήρων που ελέγχουν τις αρθρώσεις του αντίχειρα και των πλάγιων κινήσεων των δαχτύλων (εικόνα 29). Συνολικά χρησιμοποιήθηκαν 7 κανάλια του ADC του 2^{ου} μικροελεγκτή (ADC0 – ADC6). Ένα παράδειγμα της διαδικασίας που ακολουθήθηκε φαίνεται στο επόμενο κομμάτι κώδικα. Η διαδικασία αυτή είναι ίδια και για τα 7 κανάλια του ADC, του 2^{ου} μικροελεγκτή.

```
ISR(TIMER1_COMPC_vect)
{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL2;

        adc_result =  ADC_metatroph(2);
        PRAXI_DIGITAL2 = Katw_Akro_900_usec + (4*adc_result);
        OCR1C = (uint16_t) PRAXI_DIGITAL2;

}
```

Η κάθε διακοπή που προκαλείται από τους χρονιστές, όπως αυτή που φαίνεται παραπάνω, (τρείς διακοπές σε κάθε χρονιστή, όσοι και οι καταχωρητές OCRnA/B/C) αντιστοιχεί και στην μετατροπή ενός συγκεκριμένου καναλιού του ADC, του δεύτερου μικροελεγκτή.

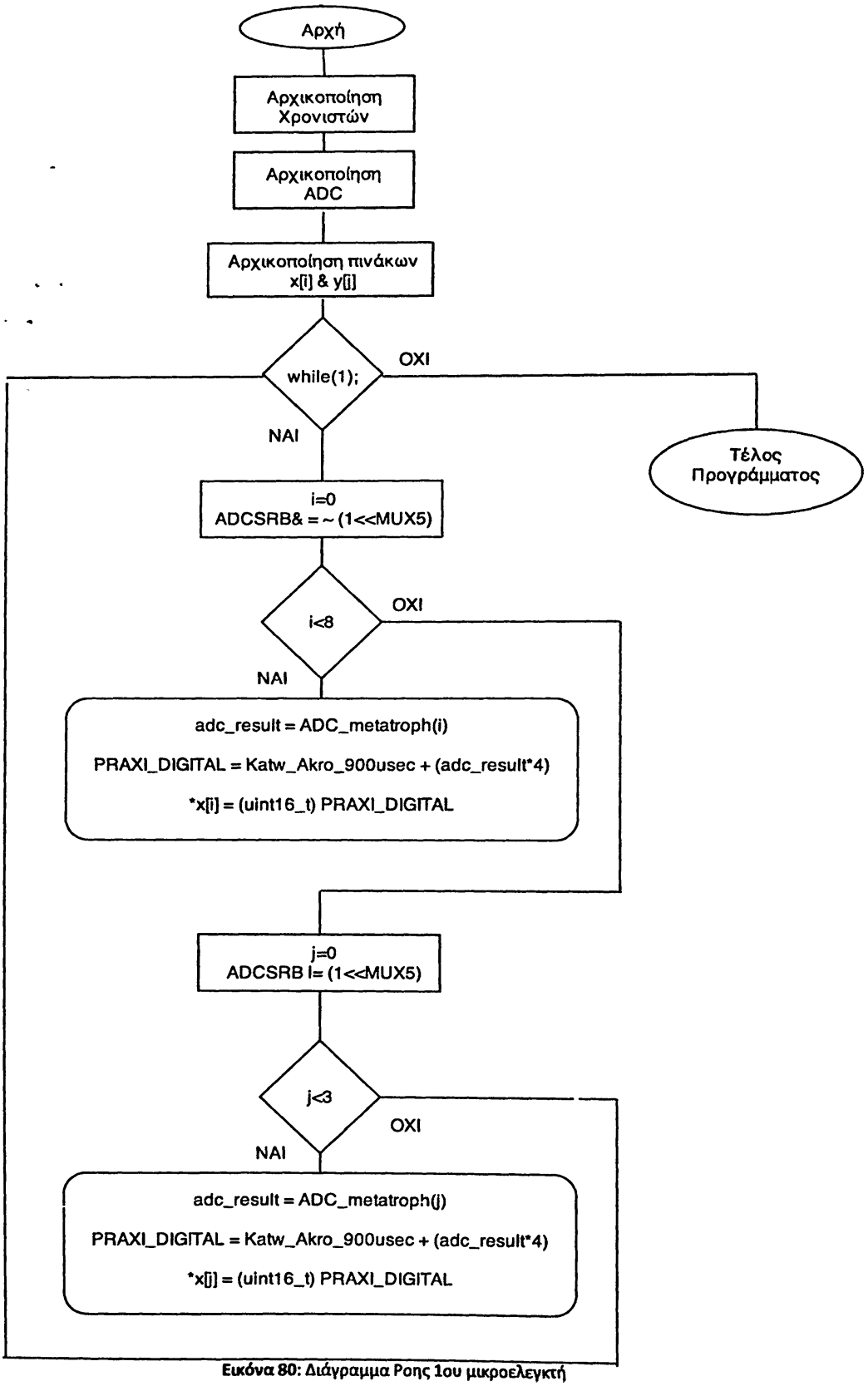
Έτσι σε κάθε διακοπή αντιστοιχεί και ένας συγκεκριμένος καταχωρητής OCRnA/B/C, όπως φαίνεται στην τελευταία εντολή του παραπάνω κώδικα. Κατά τα άλλα η διαδικασία μετατροπής των αναλογικών σημάτων είναι ίδια με το πρόγραμμα του πρώτου μικροελεγκτή.

Οι αντιστοιχίες Καναλιών ADC, καταχωρητών OCRnA/B/C και αρθρώσεων-κινητήρων ρομποτικής χείρας παρουσιάζονται στην εικόνα 79.

Ονομασία Αισθητήρα	Καταχωρητές OCRnA/B/C 1ου μικροελεγκτή	Αρθρώσεις Ρομποτικής Χείρας
A1	OCR1A	Ακραία Φαλαγγα Δείκτη
A2	OCR1B	Μεσαία Φαλαγγα Δείκτη
A3	OCR1C	Εγγύς Φαλαγγα Δείκτη
B1	OCR3A	Ακραία Φαλαγγα Μέσου
B2	OCR3B	Μεσαία Φαλαγγα Μέσου
B3	OCR3C	Εγγύς Φαλαγγα Μέσου
C1	OCR4A	Ακραία Φαλαγγα Παράμεσου
C2	OCR4B	Μεσαία Φαλαγγα Παράμεσου
C3	OCR4C	Εγγύς Φαλαγγα Παράμεσου
D2	OCR5A	Ακραία Φαλαγγα Μικρού
D2	OCR5B	Μεσαία Φαλαγγα Μικρού
D3	OCR5C	Εγγύς Φαλαγγα Μικρού
Ονομασία Αισθητήρα	Καταχωρητές OCRnA/B/C 2ου μικροελεγκτή	Αρθρώσεις Ρομποτικής Χείρας
T1	OCR1A	Ακραία Φαλαγγα Αντίχειρα
T2	OCR1B	Μεσαία Φαλαγγα Αντίχειρα
T3	OCR1C	Μετακάρπιο Αντίχειρα
A4	OCR3A	Πλάγια κίνηση Δείκτη
B4	OCR4A	Πλάγια κίνησηΜέσου
C4	OCR3B	Πλάγια κίνηση Παράμεσου
D4	OCR3C	Πλάγια κίνηση Μικρού

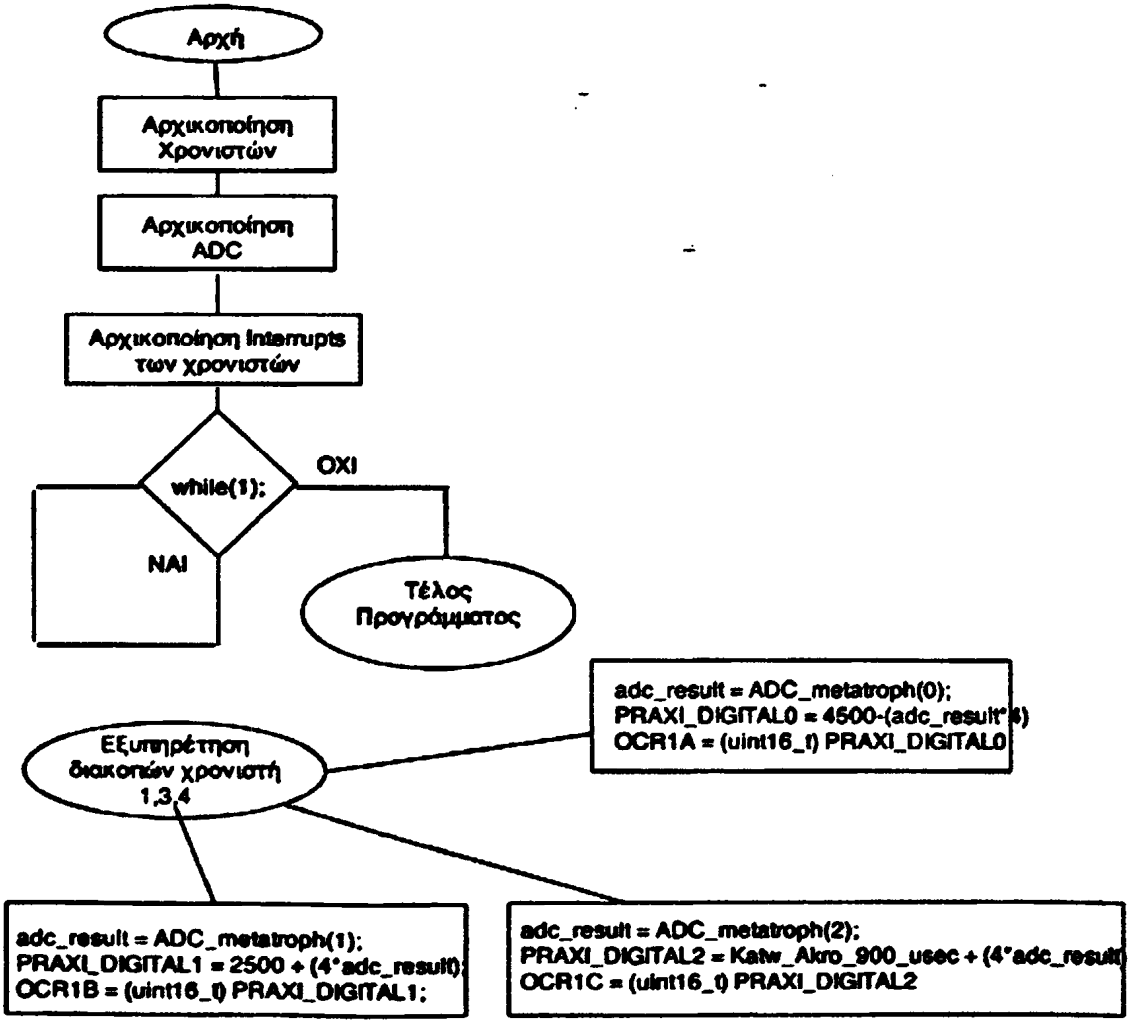
Εικόνα 79: Αντιστοιχία Αισθητήρων - Κινητήρων

Στην εικόνα 80 φαίνεται το διάγραμμα ροής για το πρόγραμμα του 1ου μικροελεγκτή.



Εικόνα 80: Διάγραμμα Ροης 1ου μικροελεγκτή

Στην εικόνα 81 φαίνεται το διάγραμμα ροής για το πρόγραμμα του 2ου μικροελεγκτή.



Εικόνα 81: Διάγραμμα ροής 2ου μικροελεγκτή

ΚΕΦΑΛΑΙΟ 7^{ο}

Κατασκευή, Ολοκλήρωση και Λειτουργία του γενικού Συστήματος

7.1 Εισαγωγή

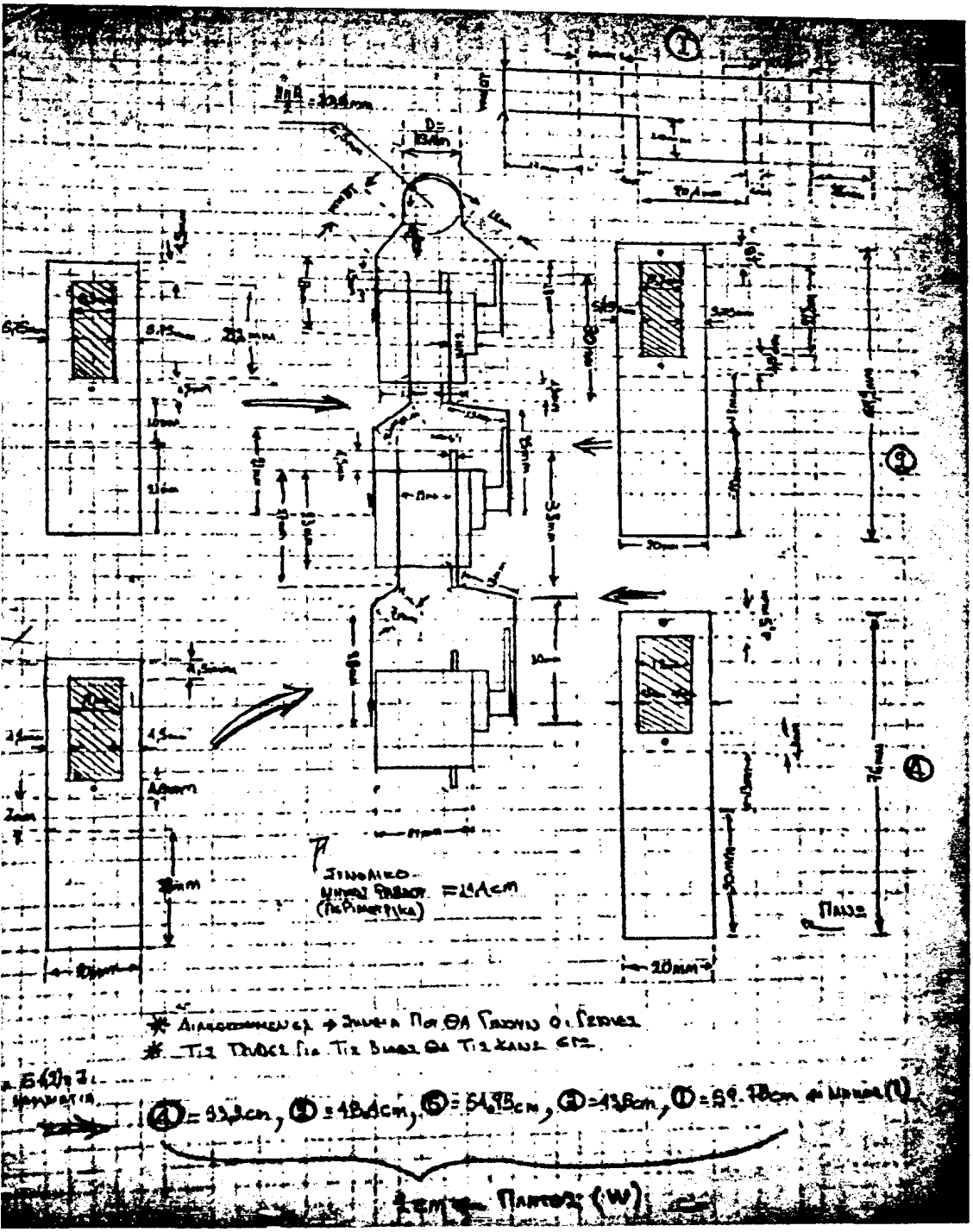
Σε αυτό το κεφάλαιο θα γίνει παρουσίαση της λειτουργίας της ρομποτικής χείρας ξεκινώντας με κάποιες πληροφορίες για την κατασκευή παραθέτοντας τα σχέδια του σκελετού.

7.2 Σχέδιο ρομποτικών δαχτύλων

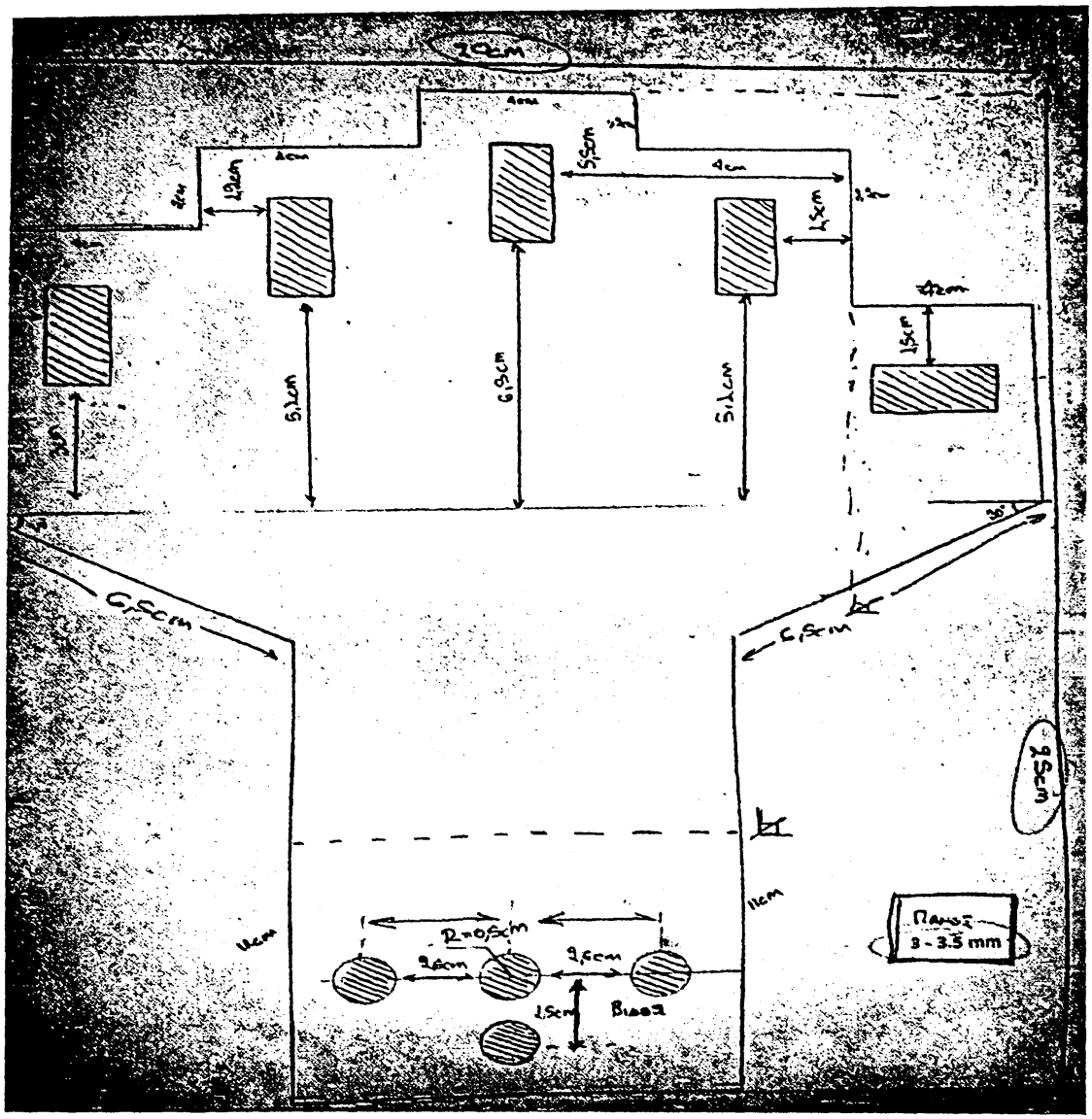
Όλα τα δάχτυλα της ρομποτικής χείρας κατασκευάστηκαν από το ίδιο σχέδιο. Ο λόγος γι'αυτό ήταν πως εφόσον οι σέρβο-κινητήρες για όλα τα δάχτυλα θα ήταν ίδιοι το μόνο που θα μπορούσε ίσως να αλλάξει στα σχέδια των δαχτύλων ήταν το μήκος τους καθώς και τα μήκη των δαχτύλων του ανθρώπινου χεριού είναι διαφορετικά.

Τελικά, μετά από αρκετή σκέψη αποφασίστηκε να χρησιμοποιηθεί για όλα τα δάχτυλα το ίδιο σχέδιο, κάτι που θα απλοποιούσε σε ένα βαθμό την διαδικασία της κατασκευής. Τα μήκη των δαχτύλων θα μπορούσαν να τροποποιηθούν έμμεσα, τοποθετώντας τα σε διαφορετικά ύψη επάνω στην παλάμη της ρομποτικής χείρας, όπως και έγινε.

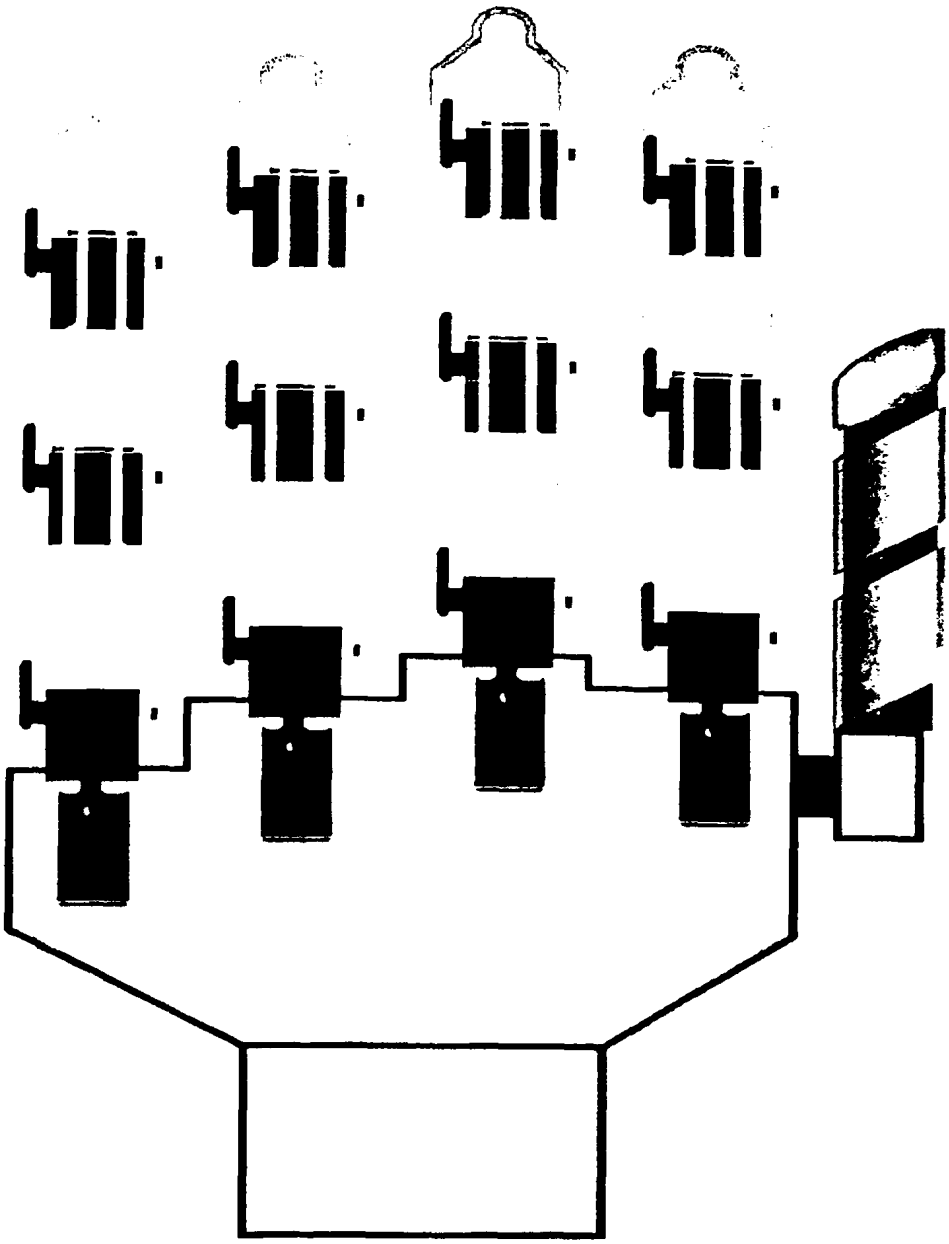
- Στην εικόνα 82 παρατίθεται σχέδιο της κάτοψης των ρομποτικών δαχτύλων. Το σχέδιο έγινε σε τετραγωνισμένο χαρτί όπως φαίνεται και στην εικόνα .
- Στην εικόνα 83 φαίνεται το σχέδιο της παλάμης με τις οπές για την τοποθέτηση των δαχτύλων.
- Στην εικόνα 84 φαίνεται το πρότυπο σχέδιο της ρομποτικής χείρας βάσει του οποίου και κατασκευάστηκε μετά.



Εικόνα82: Σχέδιο Κατασκευής δαχτύλων



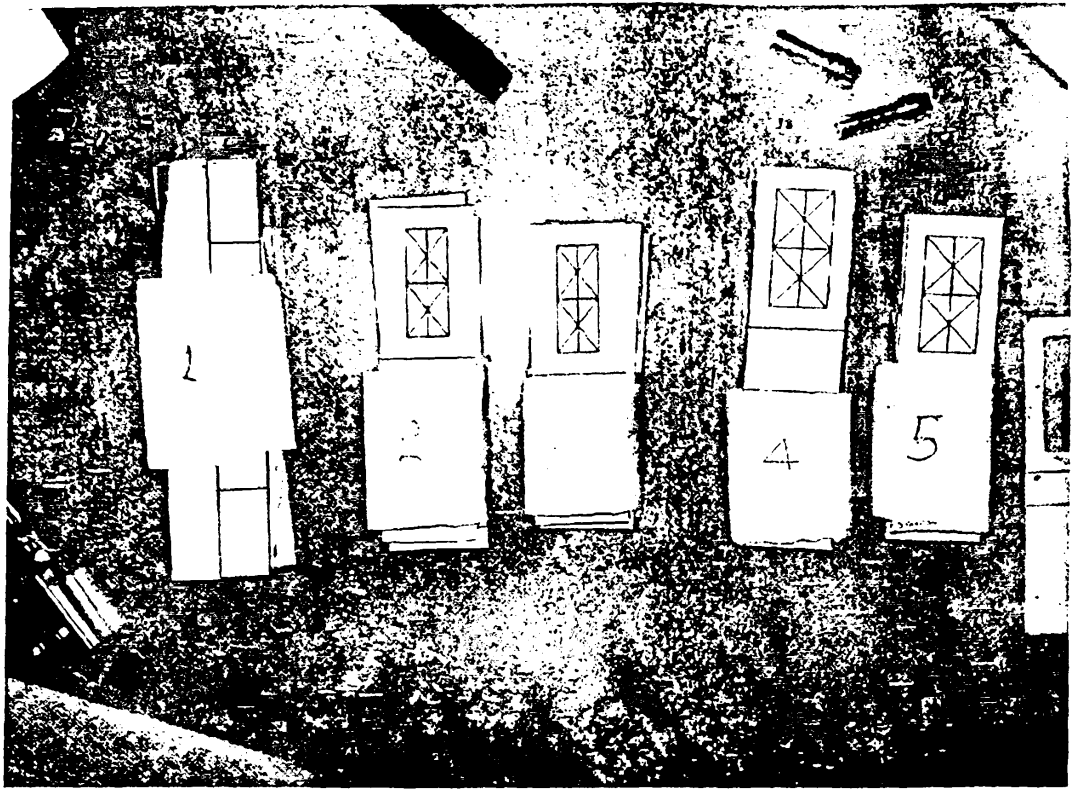
Εικόνα 83: Σχέδιο παλάμης ρομποτικής χείρας



Εικόνα 84: Πρότυπο σχέδιο

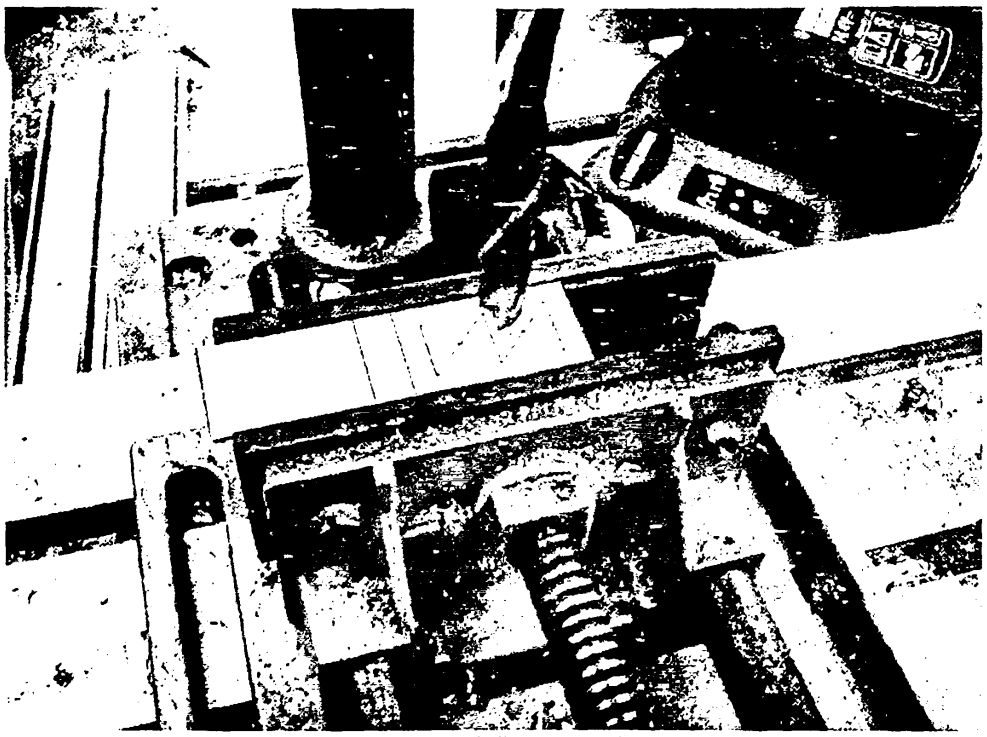
7.3 Κατασκευή Ρομποτικής Χείρας

Για την κατασκευή του σκελετού της ρομποτικής χείρας χρησιμοποιήθηκαν λωρίδες αλουμίνιου πάχους 0.8mm. Η διαδικασία που ακολουθήθηκε ήταν αρχικά το κόψιμο των λωρίδων αλουμινίου στα επιθυμητά μήκη από τις οποίες προέκυψαν τα κομμάτια που φαίνονται στην εικόνα 85.

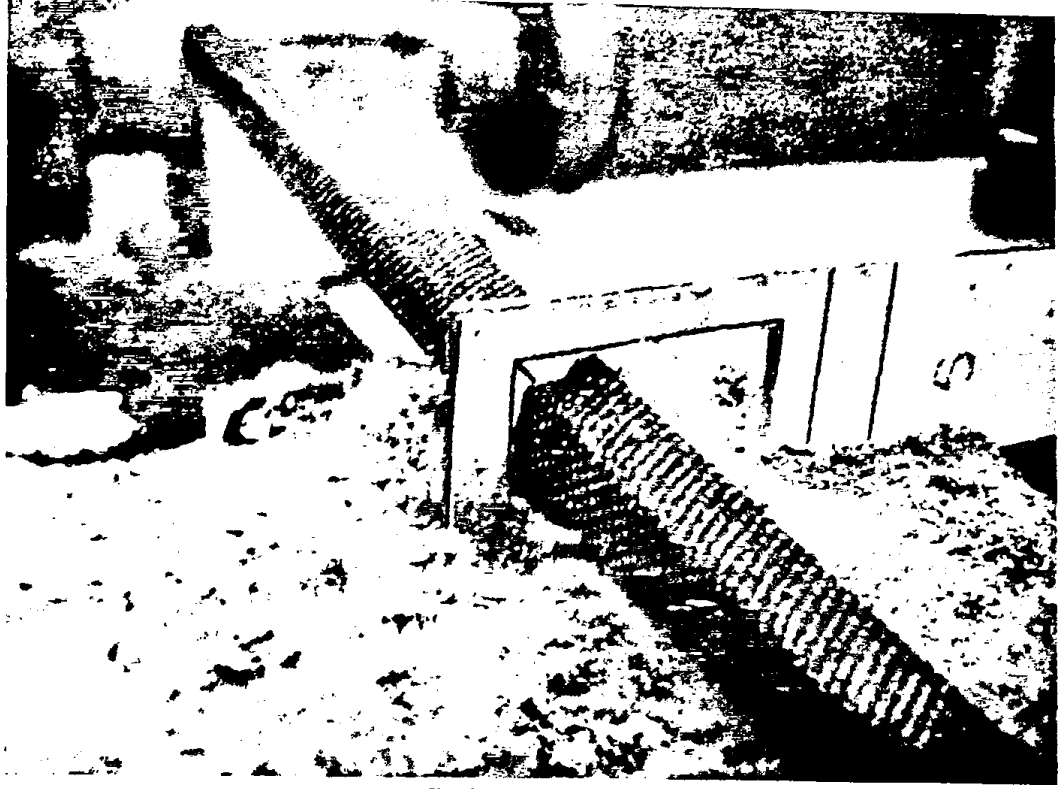


Εικόνα 85: Κομμάτια αλουμινιού

Αφού κόπηκαν τα κομμάτια μετά ανοίχθηκαν οι ορθογώνιες οπές στις προσαρμόστηκαν μετά οι σέρβο-κινητήρες. Στις εικόνες 86 και 87 φαίνεται ο τ με τον οποίο τρυπήθηκαν τα κομμάτια του αλουμινίου με χρήση τρυπανιού και στο εργαστήριο Φ.Υ.Ε του 5^{ου} ορόφου.

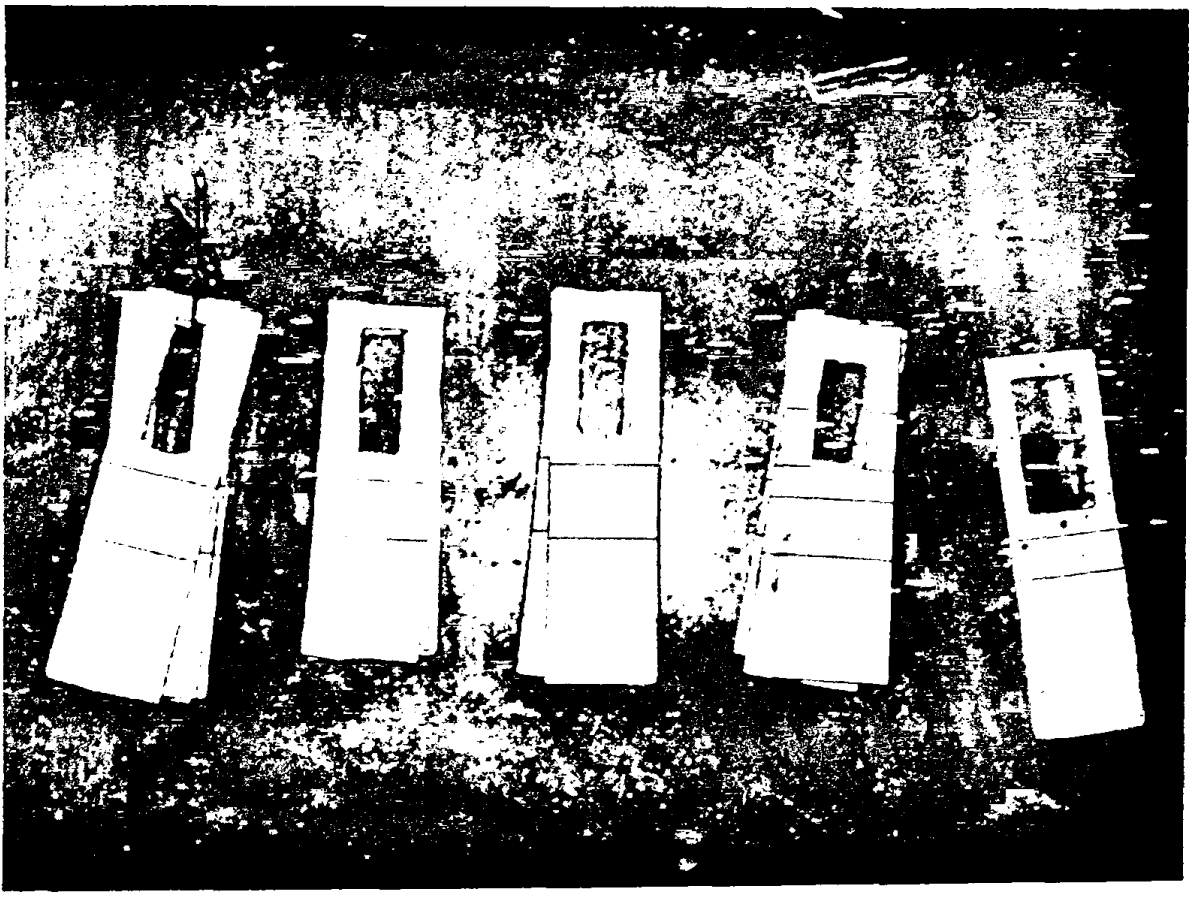


Εικόνα 86: Άνοιγμα οπών



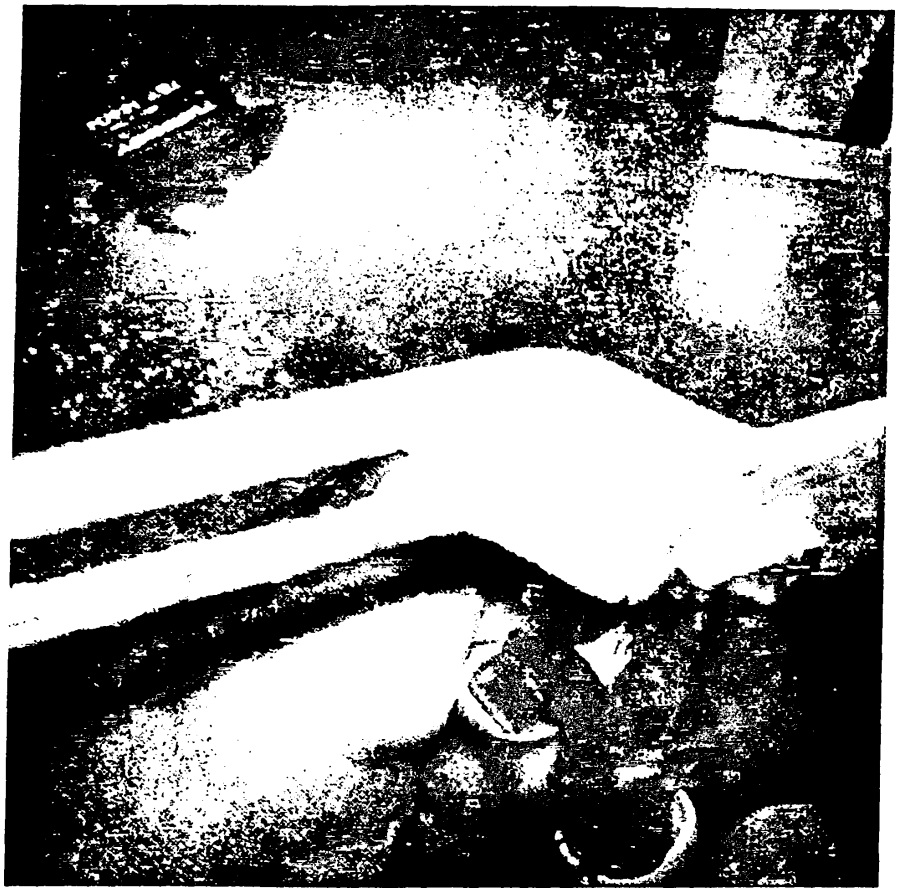
Εικόνα 87: Φινίρισμα οπών

ού ανοίχτηκαν οι ορθογώνιες οπές τα κομμάτια του σκελετού είχαν την παρακάτω
ψφή (εικόνα 88).



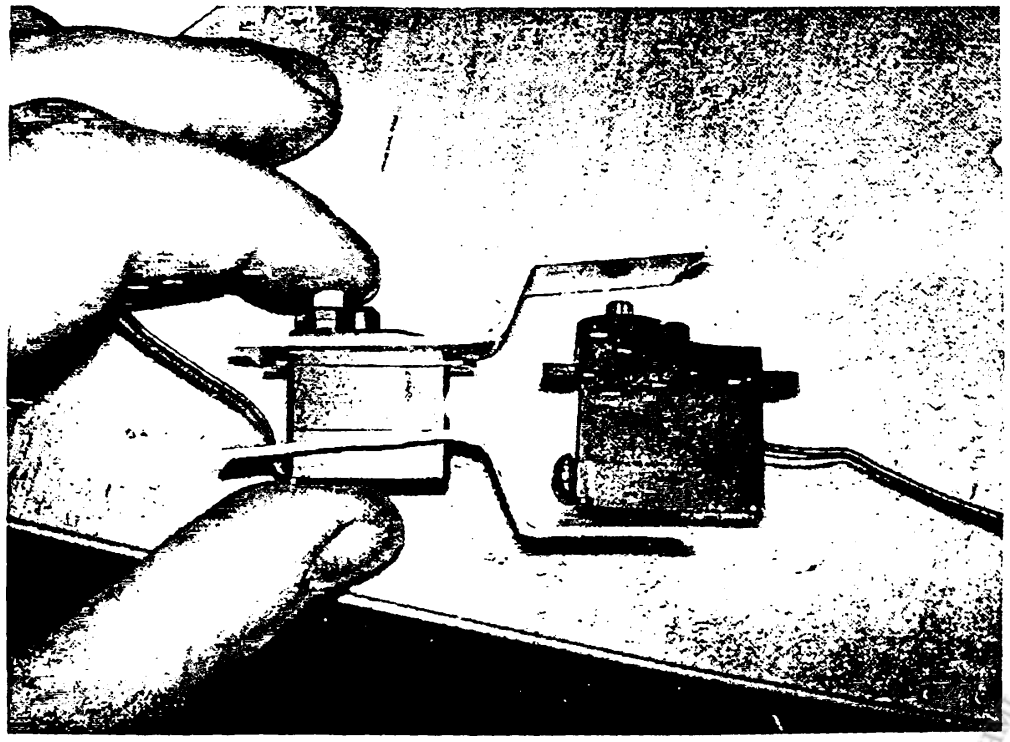
Εικόνα 88: Αποτέλεσμα επεξεργασίας

Στη συνέχεια τα κομμάτια αυτά διπλώθηκαν ώστε να σχηματιστούν τ
δαχτύλων όπως φαίνεται στην εικόνα 89.



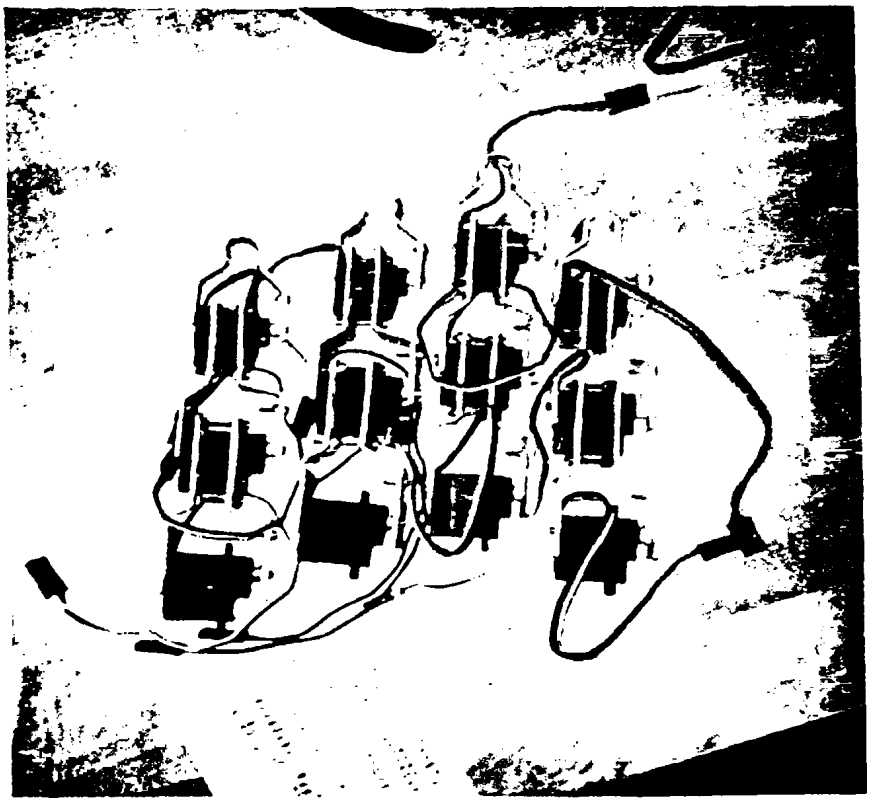
Εικόνα89: Μορφοποίηση των δαχτύλων

Αφού έγινε η κάμψη των κομματιών, τοποθετήθηκαν οι σέρβο-κινητήρες στ
πού έγιναν πριν όπως φαίνεται στην εικόνα 90.

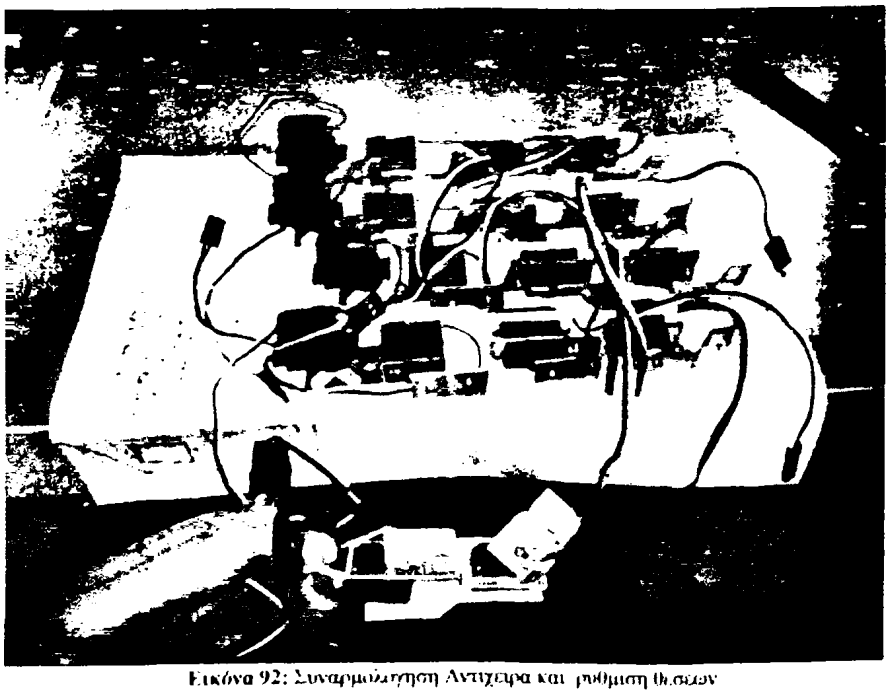


Εικόνα 90: Πριν την τοποθέτηση των servo

ιφού αυτό έγινε για όλα τα δάχτυλα της ρομποτικής χείρας, τα δάχτυλα λογήθηκαν με βίδες στους άξονες των σέρβο-κινητήρων και το αποτέλεσμα διαδικασίας φαίνεται στις εικόνες 91 και 92.



Εικόνα 91: Αποτέλεσμα συναρμολόγησης



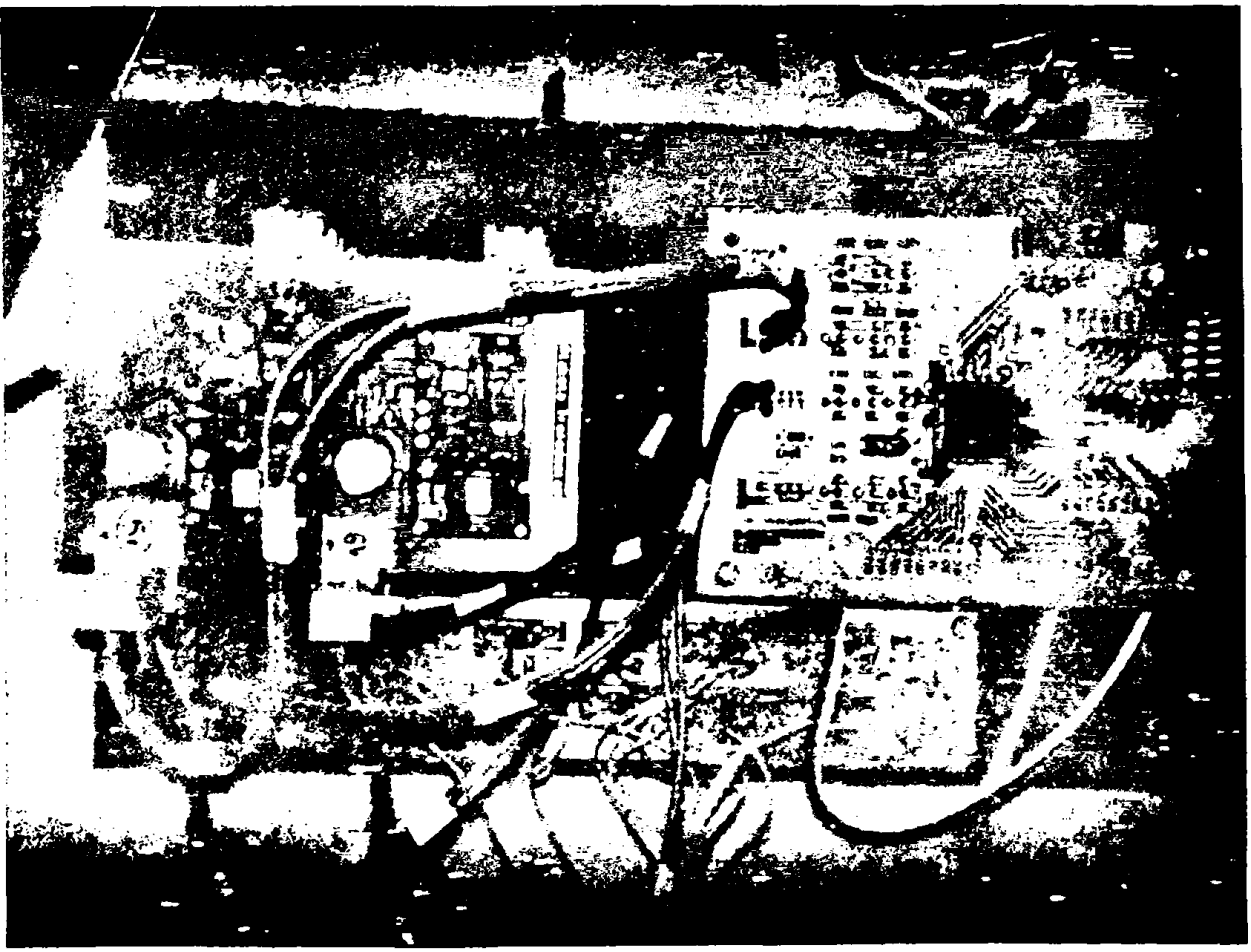
Εικόνα 92: Συναρμολόγηση Αντιχειρα και ρύθμιση θέσεων

παλάμη της ρομποτικής χειρός φαίνεται στις εικόνες 93 και 94, μετά την ποθέτηση των σέρβο-κινητήρων.

Εικόνα 93: Μέσα μέρος παλάμης

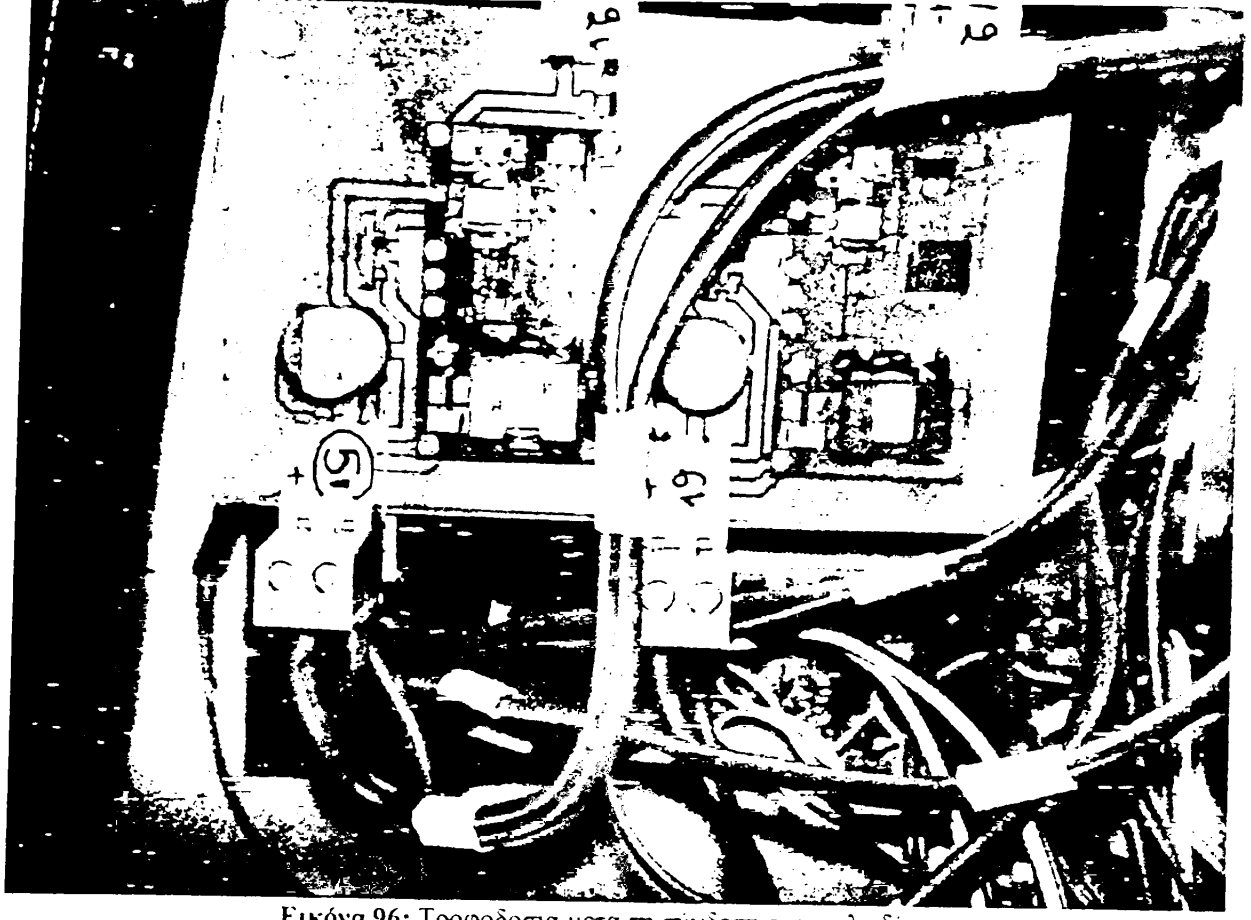
.4 Συνδεσμολογία Ρομποτικής Χείρας με σύστημα ελέγχου

Τελειώνοντας την κατασκευή και συναρμολόγηση της ρομποτικής χείρας, έγινε η συνδεσμολογία των επιμέρους πλακετών, όπως αυτές περιγράφθηκαν στα προηγούμενα κεφάλαια, με τη διάταξη αισθητήρων και τους κινητήρες της ρομποτικής χείρας. Όλο το σύστημα ελέγχου τοποθετήθηκε επάνω σε μεταλλική βάση (εικόνα 95). Στις εικόνες 96, 97 και 98 διακρίνονται οι πλακέτες της τροφοδοσίας των μικροελεγκτών και της πλακέτας παραγωγής αναλογικών σημάτων αντίστοιχα.

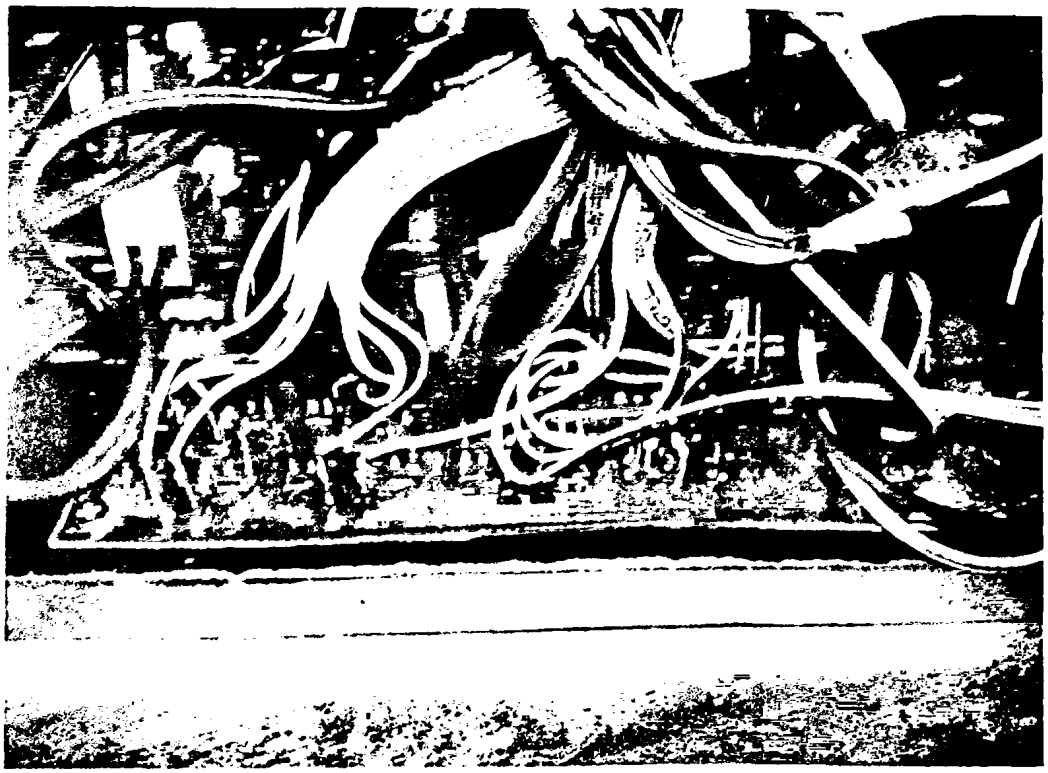


Εικόνα 95: Σύστημα τοποθετημένο στη βάση του

Καθώς το σύστημα δεν φαινόταν καλά από τα πολλά καλώδια του κυκλώματος, αυτά αφαιρέθηκαν και μετά τραβήχτηκε η φωτογραφία της εικόνας 90.



Εικόνα 96: Τροφοδοσια μετα τη σύνδεση των καλωδίων



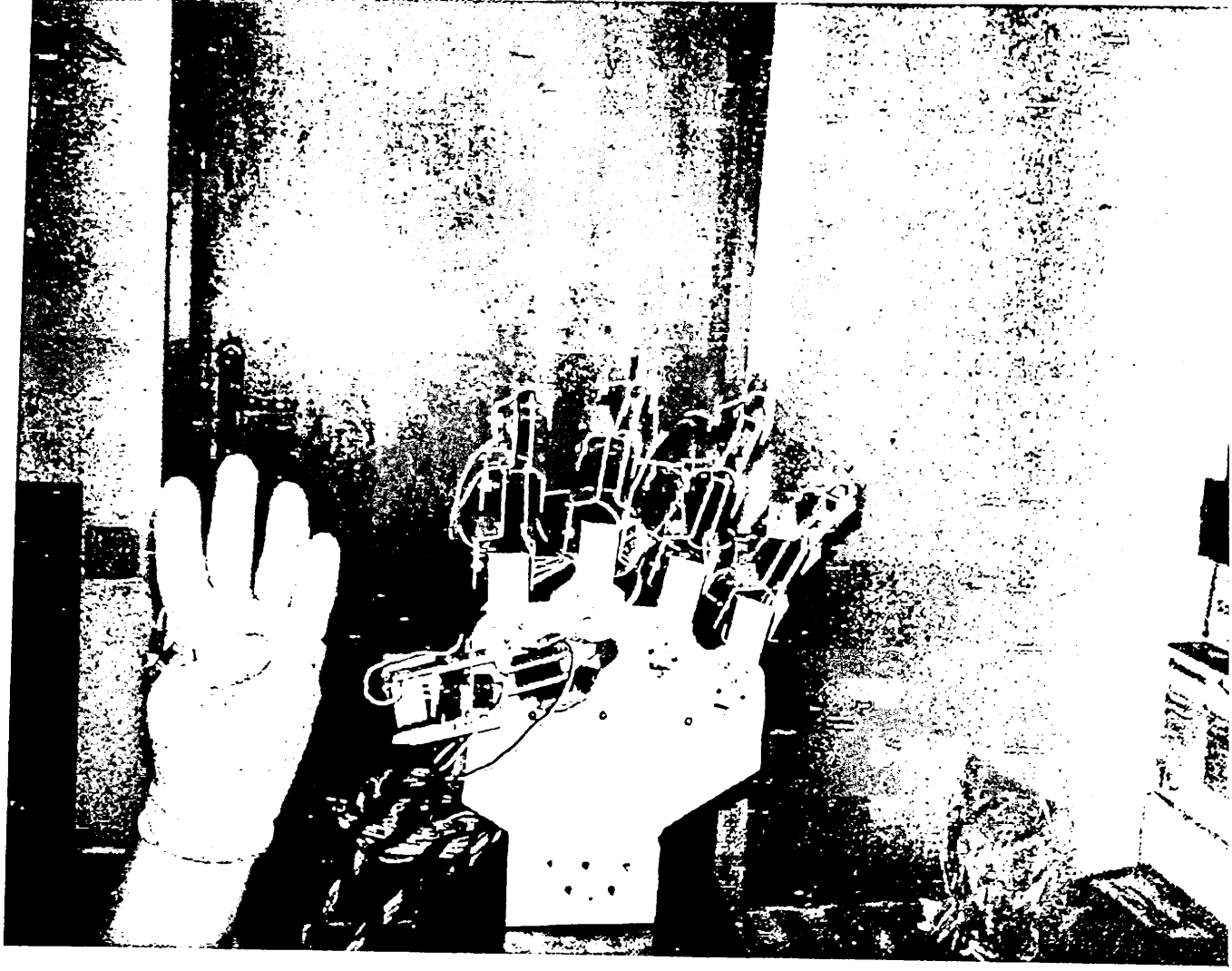
Εικόνα 98: Συνδεσμολογία πλακέτας παραγωγής σημάτων με ακροδέκτες αισθητήρων

7.5 Λειτουργία και Δοκιμές Ρομποτικής Χείρας

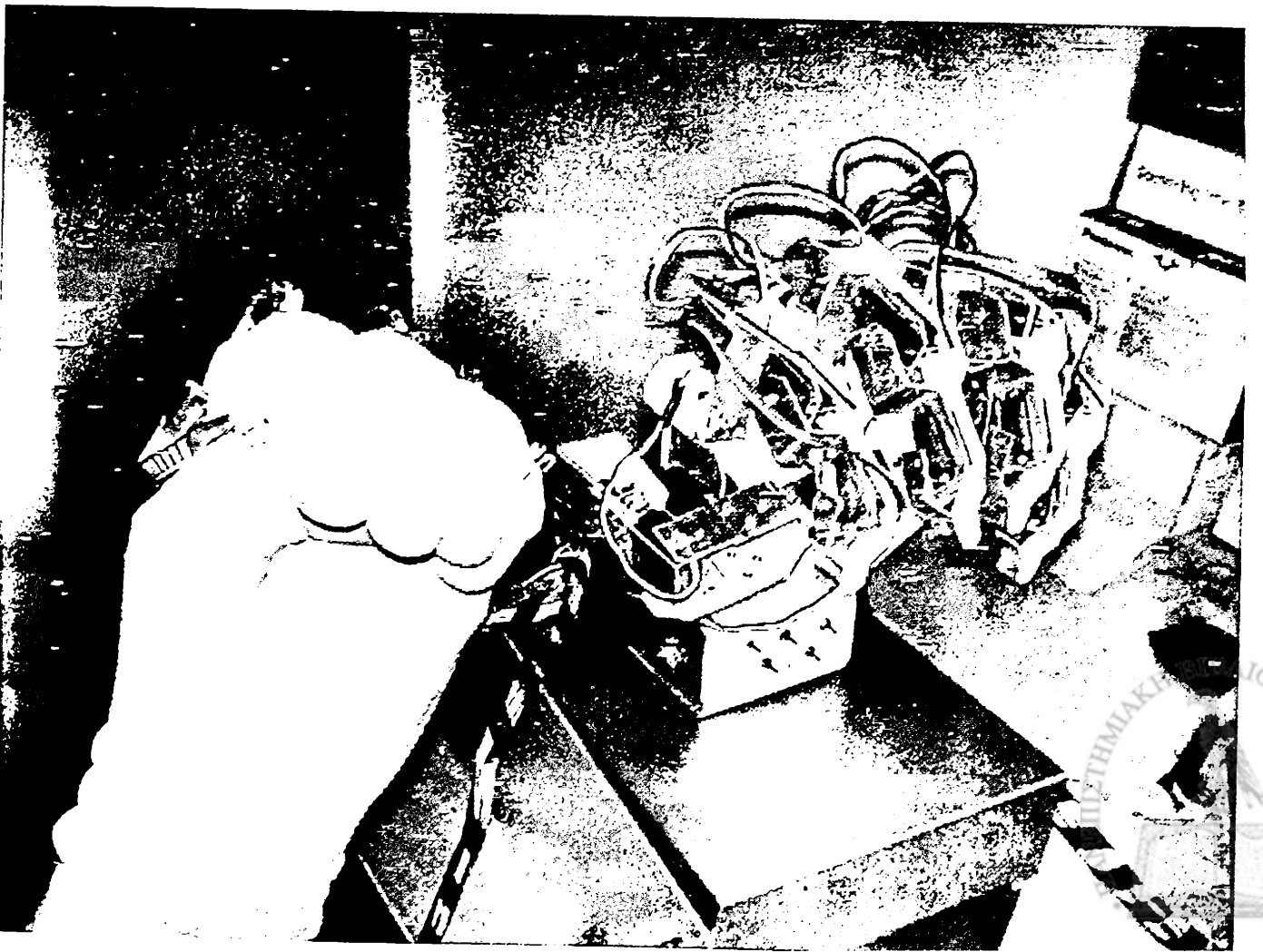
Τέλος, εφόσον το σύστημα της ρομποτικής χείρας έχει ολοκληρωθεί πλήρως, είναι δυνατό να γίνουν οι απαραίτητες δοκιμές και κάποιες ενδεικτικές κινήσεις της κρατώντας κάποια αντικείμενα διαφορετικού μεγέθους και βάρους και γενικότερα αντιγράφοντας τις κινήσεις του ανθρώπινου χεριού, μιας και αυτός ήταν και ένας από τους πρωταρχικούς στόχους.

Αρχικά ο χρήστης θα πρέπει να τοποθετήσει στο χέρι του το γάντι με τους αισθητήρες. Στη συνέχεια για την ενεργοποίηση του συστήματος κλείνουμε τους δύο διακόπτες που βρίσκονται τοποθετημένοι στο τροφοδοτικό της κατασκευής. Μόλις αυτό συμβεί, το πράσινο LED ένδειξης λειτουργίας ανάβει (το οποίο είναι τοποθετημένο πάνω στο τροφοδοτικό) και αμέσως η ρομποτική χείρα αρχίζει να αντιγράφει τις κινήσεις του χεριού του χρήστη.

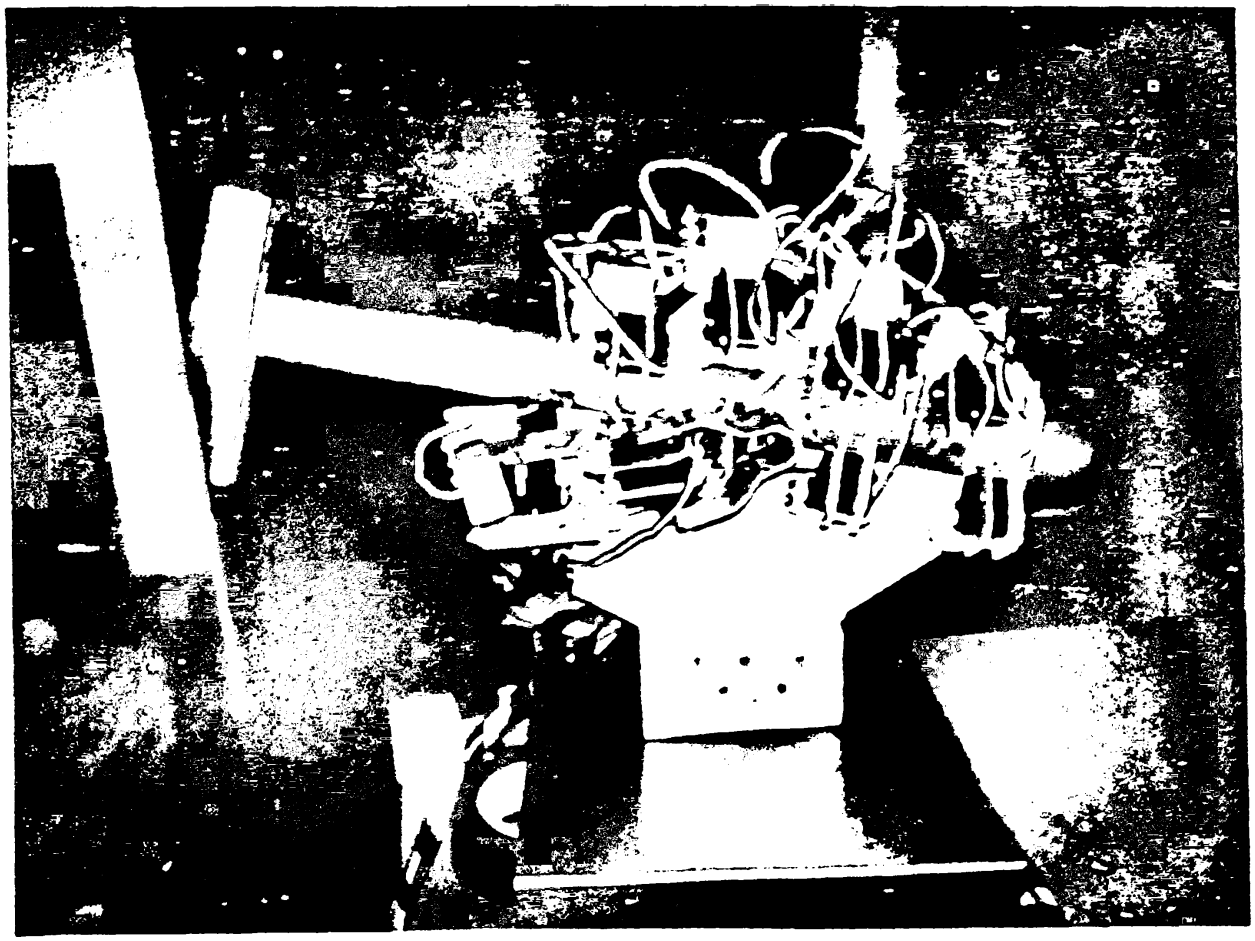
Πρώτα έγινε μία απλή κίνηση, έχοντας τα δάχτυλα σε πλήρη έκταση και τον αντίχειρα λυγισμένο προς την παλάμη, όπως αυτή που φαίνεται στην εικόνα 99. Στη συνέχεια τα δάχτυλα τοποθετήθηκαν πάνω από τον αντίχειρα όπως φαίνεται στην εικόνα 100.



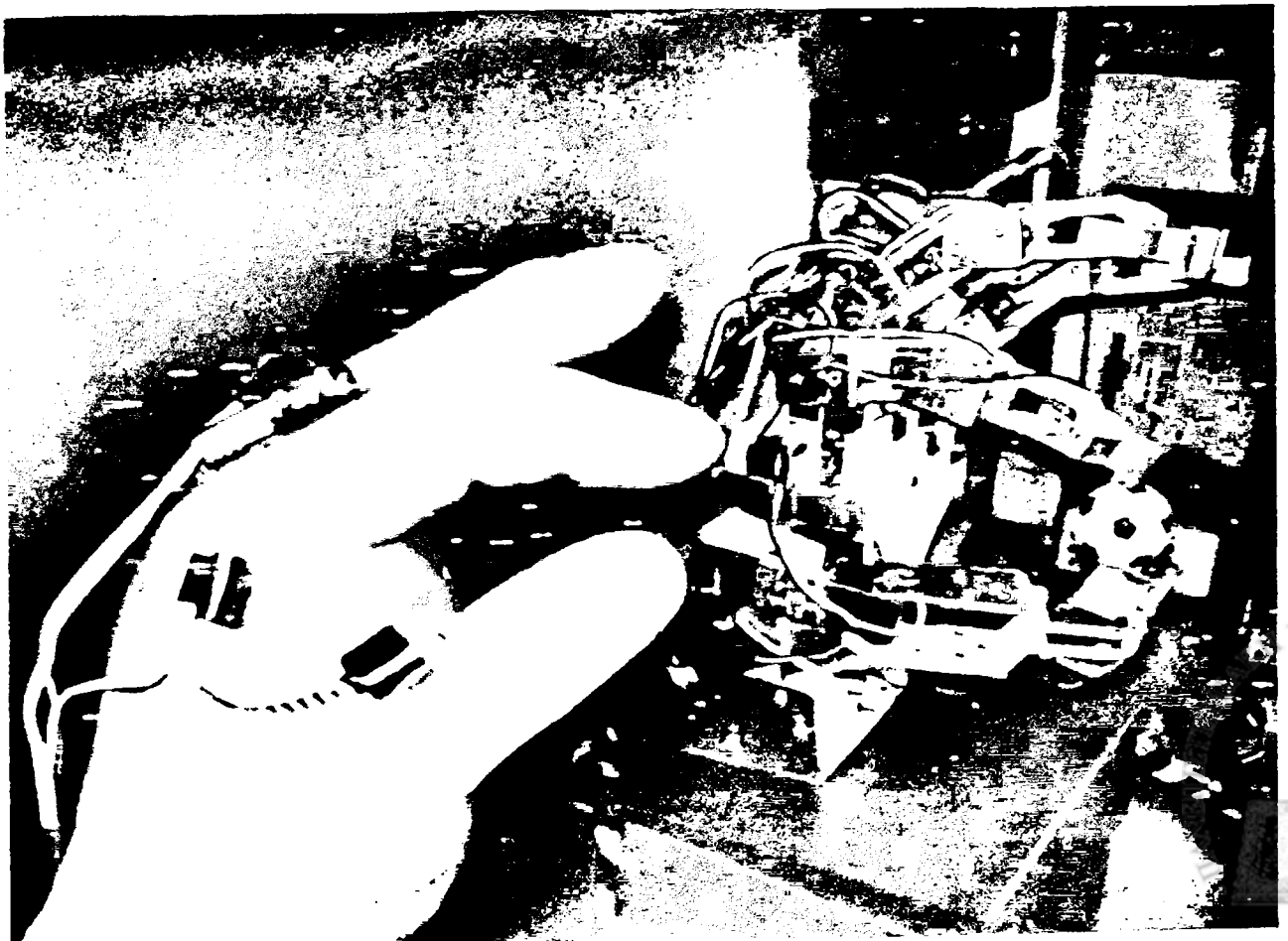
Εικόνα 99: Πλήρης Έκταση Δαχτύλων



νες 101 και 102 κρατήθηκαν δύο αντικείμενα, ένα σφυρί με μέτριο βάρος συνέχεια κρατήθηκε μία μικρή μπάλα αμελητέου βάρους. Και στις δύο σεις η ρομποτική χείρα ανταποκρίθηκε αρκετά καλά.



Εικόνα 101: Κρατημα βαρέως αντικειμενου



Παραρτήματα

Π1. Κώδικας του Πρώτου Μίκροελεγκτή

```
*
 * final.c
 *
 * Created: 14/2/2014 1:47:01 μμ
 *  Author: Lefteris
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <math.h>
#include <avr/interrupt.h>


#define OCR_HIGH 0x07                    // Arxikh 8esh Servo High Byte
#define OCR_LOW 0xCF                     // Arxikh 8esh Servo Low  Byte
#define Timer_Counter_Control_Reg_A  0xAA        // Fast PWM MODE_14, OCRnA kai
                                                 //OCRnB kai OCRnC ON se leitourgia
                                                 //NonInverting PWM
#define Timer_Counter_Control_Reg_B  0x1A        // Fast PWM MODE_14, Prescaler 8
#define Freq_H 0x9C                      // 50Hz frequency High Byte
#define Freq_L 0x3F                      // 50Hz frequency Low  Byte

#define Katw_Akro_900_usec 1400

uint16_t ADC_metatroph (unsigned int i) // synarthsh metatrophs
{
        uint16_t adc_result = 0;
        uint8_t adc_low_byte = 0;

        ADMUX = 0x40+i;
        ADCSRA |= (1<<ADSC)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); //Xekina metatroph me
                                                              // Prescaler 128
        while((ADCSRA & (1<<ADIF))==0);
        adc_low_byte = ADCL;
        adc_result = ADCH<<8 | adc_low_byte;  // Apo8ikeyse to apotelesma ths metatrophs sth
                                              // metavlhth adc_result


        return adc_result ;
}


int main(void)
{
        DDRD = 0x80;   // TEST POINT
        DDRB |= (1<<5) | (1<<6) | (1<<7);        // PWMs(A/B/C) Timer1 output enable
        DDRE |= (1<<3) | (1<<4) | (1<<5);        // PWMs(A/B/C) Timer3 output enable
        DDRH |= (1<<3) | (1<<4) | (1<<5);              // PWMs(A/B/C) Timer4 output enable
        DDRL |= (1<<3) | (1<<4) | (1<<5);        // PWMs(A/B/C) Timer5 output enable

        //OCR1AH = OCR_HIGH; //========================== Timer 1 A/B/C initialize
        //OCR1AL = OCR_LOW;
        //OCR1BH = OCR_HIGH;
        //OCR1BL = OCR_LOW;
        //OCR1CH = OCR_HIGH;
```

```
//OCR1CL = OCR_LOW;
TCCR1A = Timer_Counter_Control_Reg_A;
TCCR1B = Timer_Counter_Control_Reg_B;
ICR1H = Freq_H;
ICR1L = Freq_L;

//OCR3AH = OCR_HIGH; //========================== Timer 3 A/B/C initialize
//OCR3AL = OCR_LOW;
//OCR3BH = OCR_HIGH;
//OCR3BL = OCR_LOW;
//OCR3CH = OCR_HIGH;
//OCR3CL = OCR_LOW;
TCCR3A = Timer_Counter_Control_Reg_A;
TCCR3B = Timer_Counter_Control_Reg_B;
ICR3H = Freq_H;
ICR3L = Freq_L;

//OCR4AH = OCR_HIGH; //========================== Timer 4 A/B/C initialize
//OCR4AL = OCR_LOW;
//OCR4BH = OCR_HIGH;
//OCR4BL = OCR_LOW;
//OCR4CH = OCR_HIGH;
//OCR4CL = OCR_LOW;
TCCR4A = Timer_Counter_Control_Reg_A;
TCCR4B = Timer_Counter_Control_Reg_B;
ICR4H = Freq_H;
ICR4L = Freq_L;

//OCR5AH = OCR_HIGH; //========================== Timer 5 A/B/C initialize
//OCR5AL = OCR_LOW;
//OCR5BH = OCR_HIGH;
//OCR5BL = OCR_LOW;
//OCR5CH = OCR_HIGH;
//OCR5CL = OCR_LOW;
TCCR5A = Timer_Counter_Control_Reg_A;
TCCR5B = Timer_Counter_Control_Reg_B;
ICR5H = Freq_H;
ICR5L = Freq_L;


volatile unsigned int* x[] = { &OCR1A, &OCR1B, &OCR1C, &OCR3A, &OCR3B,
                               &OCR3C, &OCR4A, &OCR4B } ;
                        // registers A/B apo tous opoious orizetai h 8esh twn Dig.Servo

volatile unsigned int* y[] = { &OCR4C, &OCR5A, &OCR5B, &OCR5C };
                        // registers C apo tous opoious orizetai h 8esh twn Dig.Servo

float PRAXI_DIGITAL = 0; // Edw apo8hkeuetai h metatroph ths ADC metatrophs se xrono
                          //odhghshs gia ton Psifiako servo.
//float PRAXI_ANALOG = 0; // Edw apo8hkeuetai h metatroph ths ADC metatrophs se xrono
                          //odhghshs gia ton Analogiko servo.
int i = 0;                // Metavlith pou epilegei to ADC0:7 kanali kai thn exodo PWM se
                          // syndiasmo me ton pinaka x[]
int j = 0;                // Metavlith pou epilegei to ADC8:11 kanali kai thn exodo PWM
                          // se syndiasmo me ton pinaka y[]
uint16_t adc_result = 0;

ADCSRA |= (1<<ADEN);      //TO bit ADEN to kanw 1 exw apo thn for( h kai exw apo thv
                          //while). An to kanw mesa sth for prin to start,
                          //opws to ekana prin dhladh, 8a pernei san metrhsh thn 1h h
                          //opoia einai panta lan8asmenh
```

```
//TIMSK1 = (1<<TOIE1);
//sei();

while(1)
{

    ADCSRB &= 0x00;   // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
                      //kai afhnw ta ypoloipa bits opws htan

    for ( i=0; i<8; i++)          diavasma 8 prwtwn kanaliwn
    {
        adc_result =  ADC_metatroph(i);  //Klhsh synarthshs metatrophs
        PRAXI_DIGITAL = Katw_Akro_900_usec + (adc_result*4);
        *x[i] = (uint16_t) PRAXI_DIGITAL;
    }

    ADCSRB |=(1<<MUX5);
    for ( j=0; j<3; j++)          //diavasma twn 3 teleutaiwn kanaliwn

        if( j == 2)
        {
            adc_result =  ADC_metatroph(2); //Klhsh synarthshs metatrophs
            PRAXI_DIGITAL = Katw_Akro_900_usec + (adc_result*4);
            *y[2] = (uint16_t) PRAXI_DIGITAL;
            *y[3] = (uint16_t) PRAXI_DIGITAL;
        }
        else
        {
            adc_result =  ADC_metatroph(j); //Klhsh synarthshs metatrophs
            PRAXI_DIGITAL = Katw_Akro_900_usec + (adc_result*4);
            *y[j] = (uint16_t) PRAXI_DIGITAL;

        }
}
}
```

Π2. Κώδικας του Δεύτερου Μικροελεγκτή

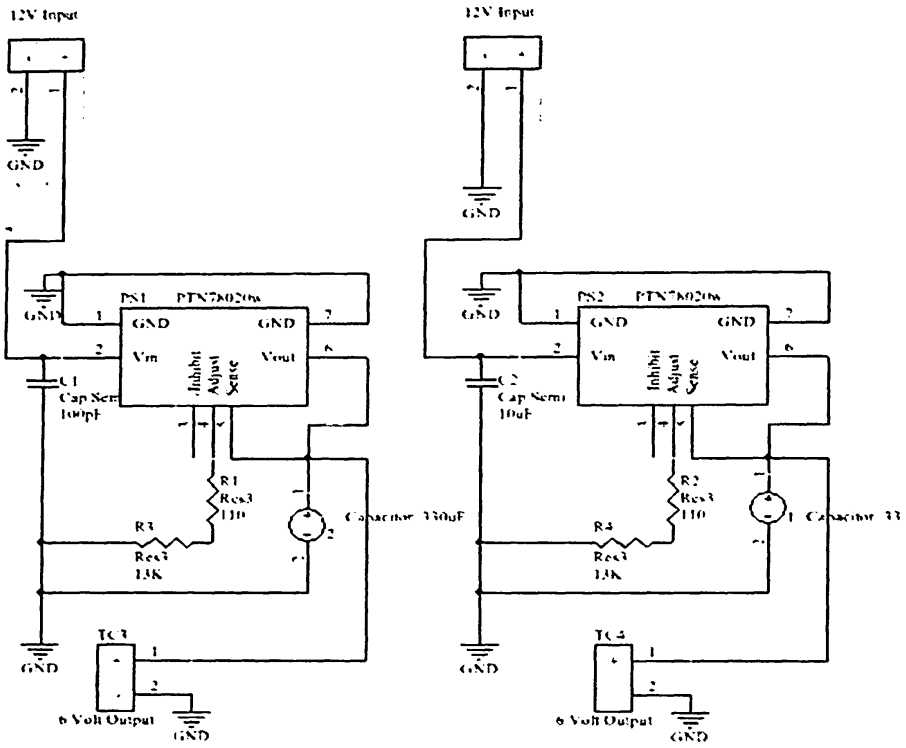
```
/*
 * final2.c
 *
 * Created: 14/2/2014 1:47:01 μμ
 *  Author: Lefteris
 */

#define F_CPU 16000000UL
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
#include <math.h>



#define OCR_HIGH 0x0F                       // Arxikh 8esh Servo High Byte
#define OCR_LOW 0x9F                        // Arxikh 8esh Servo Low  Byte
#define Timer_Counter_Control_Reg_A  0xAA        // Fast PWM MODE_14, OCRnA kai OCRnB
                                                 //kai OCRnC ON se leitourgia NonInverting
                                                 //PWM
#define Timer_Counter_Control_Reg_B  0x1A   // Fast PWM MODE_14, Prescaler 8
#define Freq_H_50Hz 0x9C                     // 50Hz frequency High Byte
#define Freq_L_50Hz 0x3F                    // 50Hz frequency Low  Byte

#define Katw_Akro_900_usec 1400

uint16_t ADC_metatroph (unsigned int i)
{
        uint16_t adc_result = 0;
        uint8_t adc_low_byte = 0;

        ADMUX = 0x40+i;
        ADCSRA |= (1<<ADSC)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); //Xekina metatroph me
                                                                //Prescaler 128
        while((ADCSRA & (1<<ADIF))==0);
        adc_low_byte = ADCL;
        adc_result = ADCH;
        adc_result = ADCH<<8 | adc_low_byte;


        return adc_result ;   // epestrepse to apotelesma
}
int main(void)
{
        DDRC = 0xFF;
        DDRA = 0xFF;
        DDRG = 0x03;
        DDRD = 0x80;   // TEST POINT
        DDRB |= (1<<5) | (1<<6) | (1<<7);        // PWMs(A/B/C) Timer1 output enable
        DDRE |= (1<<3) | (1<<4) | (1<<5);       // PWMs(A/B/C) Timer3 output enable
        DDRH |= (1<<3) | (1<<4) | (1<<5);              // PWMs(A/B/C) Timer4 output enable
        //DDRL |= (1<<3) | (1<<4) | (1<<5);      // PWMs(A/B/C) Timer5 output enable


        OCR1AH = OCR_HIGH; //========================= Timer 1 A/B/C initialize
        OCR1AL = OCR_LOW;
        OCR1BH = OCR_HIGH;
        OCR1BL = OCR_LOW;
```

```
        OCR1CH = OCR_HIGH;
        OCR1CL = OCR_LOW;
        TCCR1A = Timer_Counter_Control_Reg_A;
        TCCR1B = Timer_Counter_Control_Reg_B;
        ICR1H = Freq_H_50Hz;
        ICR1L = Freq_L_50Hz;

        OCR3AH = OCR_HIGH; //========================= Timer 3 A/B/C initialize
        OCR3AL = OCR_LOW;
        OCR3BH = OCR_HIGH;
        OCR3BL = OCR_LOW;
        OCR3CH = OCR_HIGH;
        OCR3CL = OCR_LOW;
        TCCR3A = Timer_Counter_Control_Reg_A;
        TCCR3B = Timer_Counter_Control_Reg_B;
        ICR3H = Freq_H_50Hz;
        ICR3L = Freq_L_50Hz;

        OCR4AH = 0x0D; //========================= Timer 4 A initialize
        OCR4AL = 0x5F;
        TCCR4A = Timer_Counter_Control_Reg_A;
        TCCR4B = Timer_Counter_Control_Reg_B;
        ICR4H = Freq_H_50Hz;
        ICR4L = Freq_L_50Hz;


        //TIMSK4 |= (1<<OCIE4A) ; // Apo auth thn interrupt 8a ry8mizw to dexia aristera tou
                                  //mesou daxtylou

        TIMSK3 = (1<<OCIE3C) | (1<<OCIE3B) | (1<<OCIE3A) ;  // Energopoiw ta interrupts
                                                            //      gia τους timers 1, 3
        TIMSK1 = (1<<OCIE1C) | (1<<OCIE1B) | (1<<OCIE1A) ;


        //uint16_t adc_result;
        //unsigned int i;


  ADCSRA |= (1<<ADEN);    // Energopoiw ton ADC

  ADCSRB &= 0x00;          // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels


        sei();              // energopoiw ta global interrupts

        while(1)
        {


        }
}


ISR(TIMER1_COMPA_vect)    // Interrupt sygrishs TCNT1 gia OCRA
{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL0;

        adc_result = ADC_metatroph(0);
```

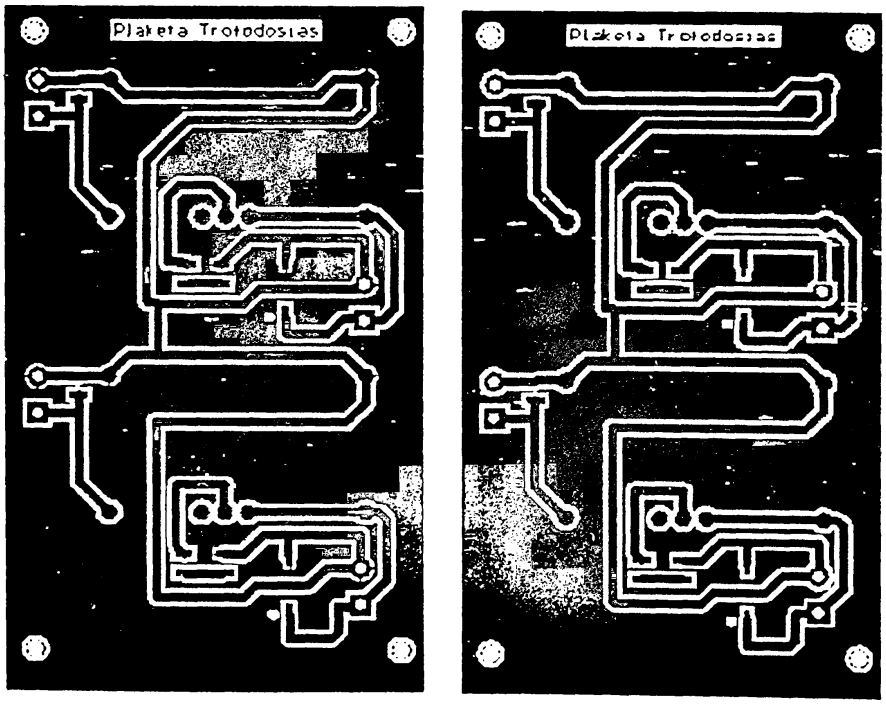
```
        PRAXI_DIGITAL0 = 4500-(adc_result*4);      // adc 0
        OCR1A = (uint16_t) PRAXI_DIGITAL0;   // megalh ar8rwsh tou antixeira
}

ISR(TIMER1_COMPB_vect)    // Interrupt sygrishs TCNT1 gia OCRB

{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL1;

        adc_result =  ADC_metatroph(1);
        PRAXI_DIGITAL1 = 2500 + (4*adc_result);    // adc 1
        OCR1B = (uint16_t) PRAXI_DIGITAL1;   // mesaia ar8rwsh tou antixeira

}



ISR(TIMER1_COMPC_vect)    // Interrupt sygrishs TCNT1 gia OCRC

{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL2;

        adc_result =  ADC_metatroph(2);
        PRAXI_DIGITAL2 = Katw_Akro_900_usec + (4*adc_result);  // adc 2
        OCR1C = (uint16_t) PRAXI_DIGITAL2;   // mikrh ar8rwsh tou antixeira

}
ISR(TIMER3_COMPA_vect)      // Interrupt sygrishs TCNT3 gia OCRA
{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL3;

        adc_result = ADC_metatroph(3);
        PRAXI_DIGITAL3= 1800 + (4*adc_result);  // adc 3
        OCR3A = (uint16_t) PRAXI_DIGITAL3;      // to dexia-aristera srv tou deikth
}
ISR(TIMER3_COMPB_vect)      // Interrupt sygrishs TCNT3 gia OCRB
{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL4;

        adc_result = ADC_metatroph(4);
        PRAXI_DIGITAL4= 7400 - (8*adc_result);  // adc 4
        OCR3B = (uint16_t) PRAXI_DIGITAL4;   // dexia - aristera srv tou paramesou daxtylou
}
ISR(TIMER3_COMPC_vect)      // Interrupt sygrishs TCNT3 gia OCRC
{
        uint16_t adc_result = 0;
        float PRAXI_DIGITAL5;
        adc_result = ADC_metatroph(5);
        PRAXI_DIGITAL5= 7400 - (8*adc_result);  // adc 5
        OCR3C = (uint16_t) PRAXI_DIGITAL5;   // dexia - aristera srv tou mikrou daxtylou

}
```

Π3 Σχηματικά και Πλακέτες συστήματος

Π3.1 Πλακέτες Τροφοδοσίας

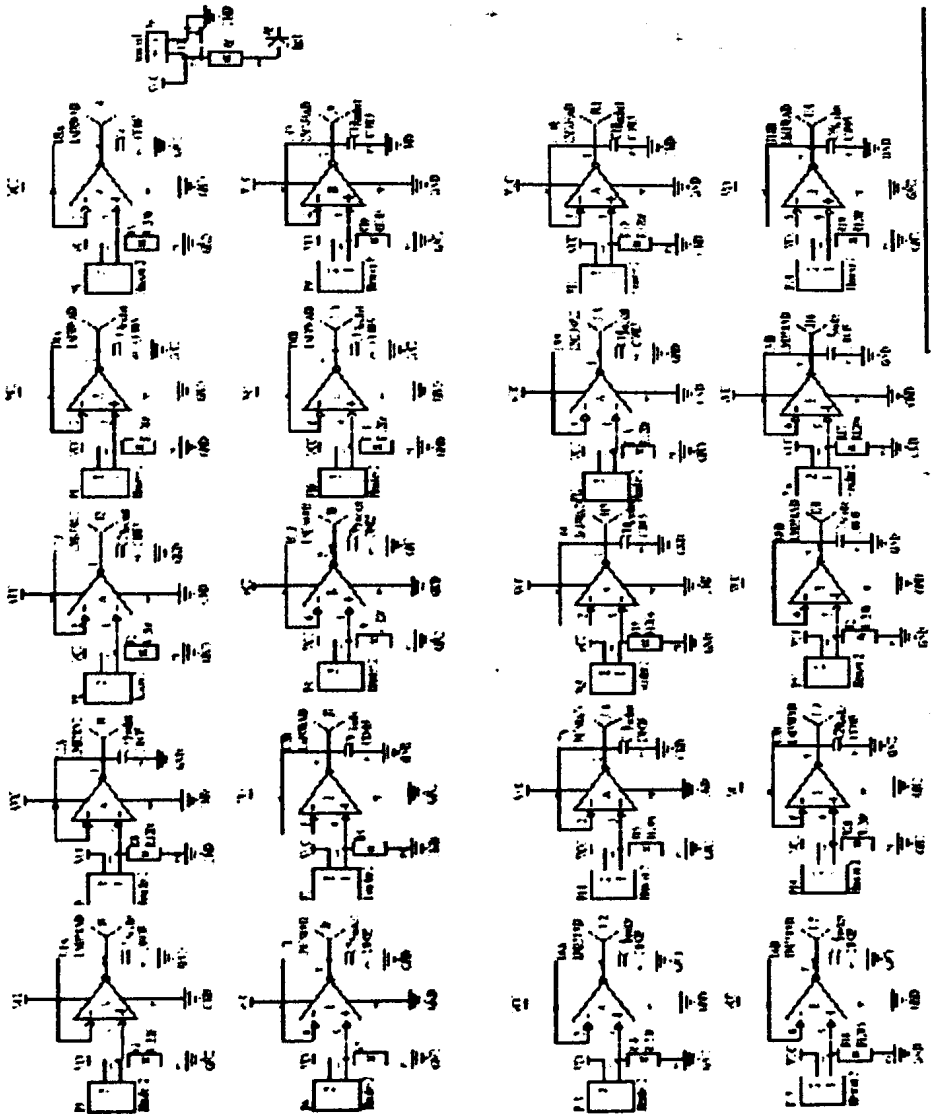


Εικόνα 103: Σχηματικό πλακέτας τροφοδοσίας συστήματος



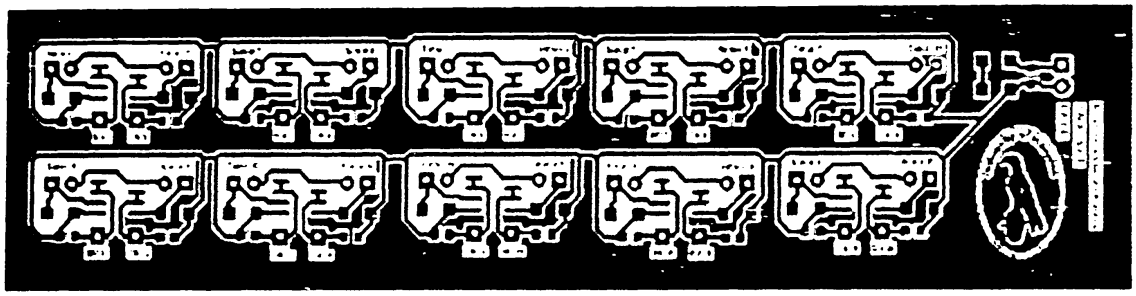
Εικόνα 104 : Πλακέτες Τροφοδοσίας

Π3.2 Πλακέτα παραγωγής αναλογικών σημάτων



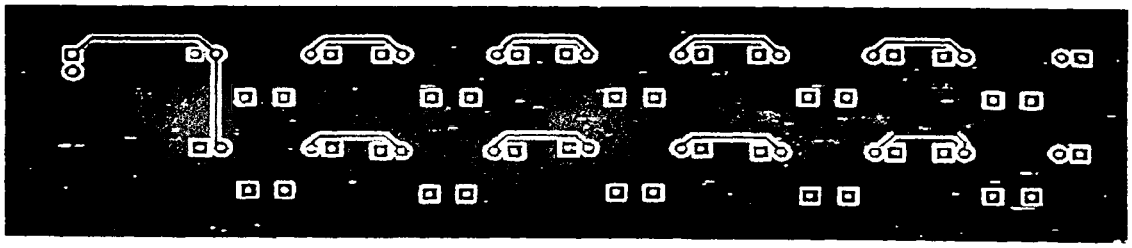
Εικόνα 105: Σχηματικό Πλακέτας παραγωγής Σημάτων

Μπροστινή Οψη



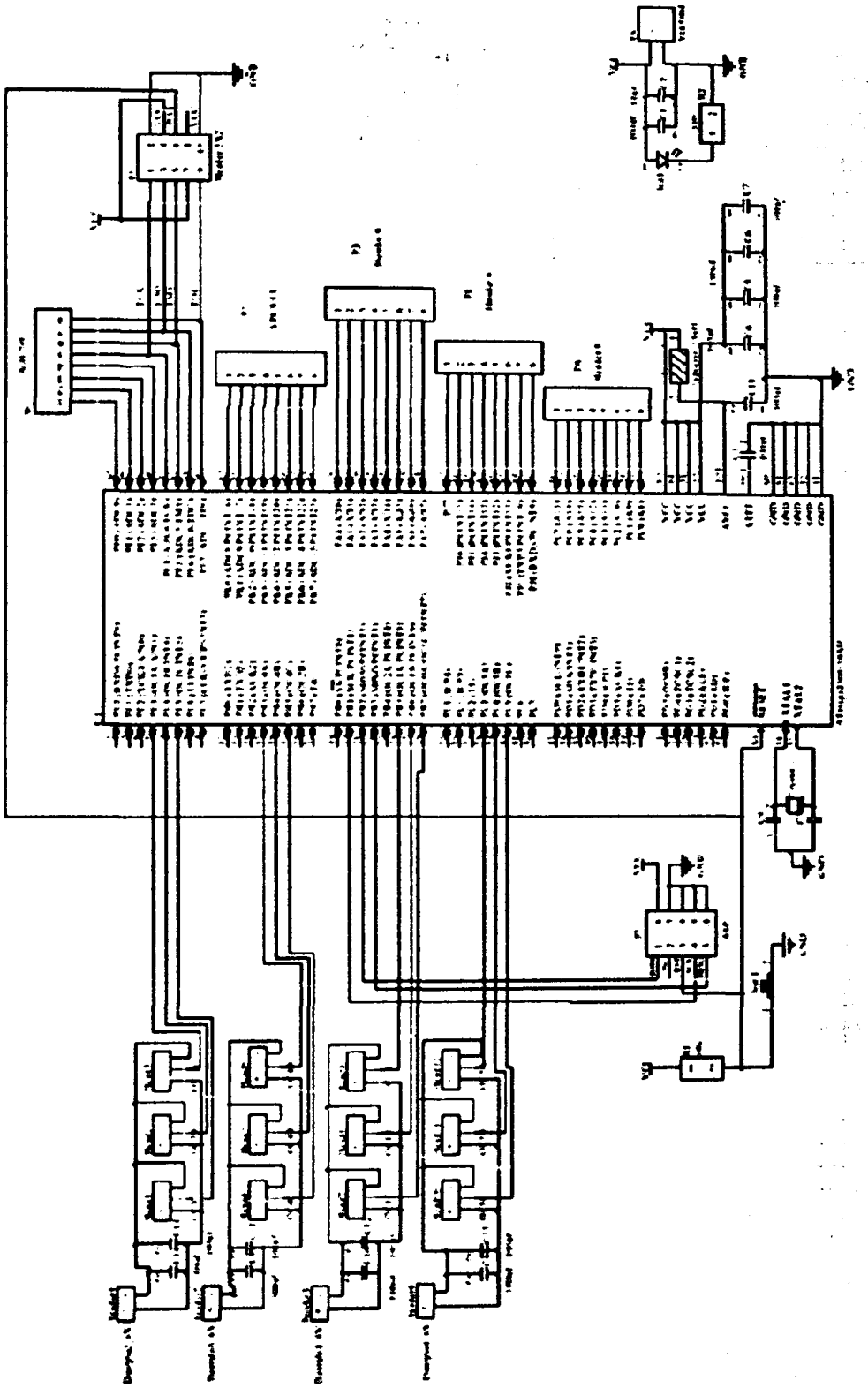
Εικόνα 106: Πλακέτα παραγωγής σημάτων

Πίσω Όψη

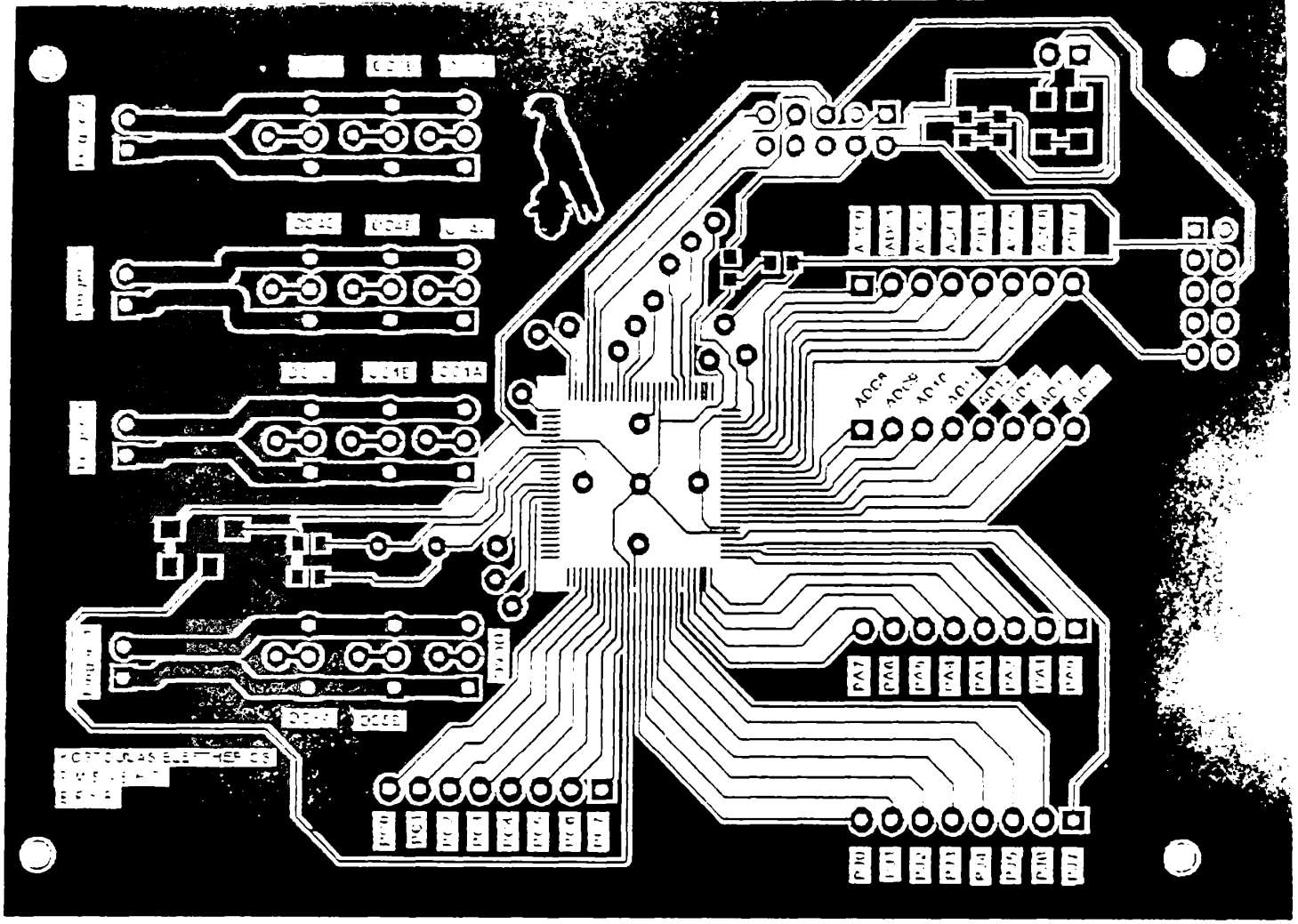


Εικόνα 107: Πίσω Όψη πλακέτας παραγωγής σημάτων

Π3.3 Πλακέτα 1ου μικροελεγκτή

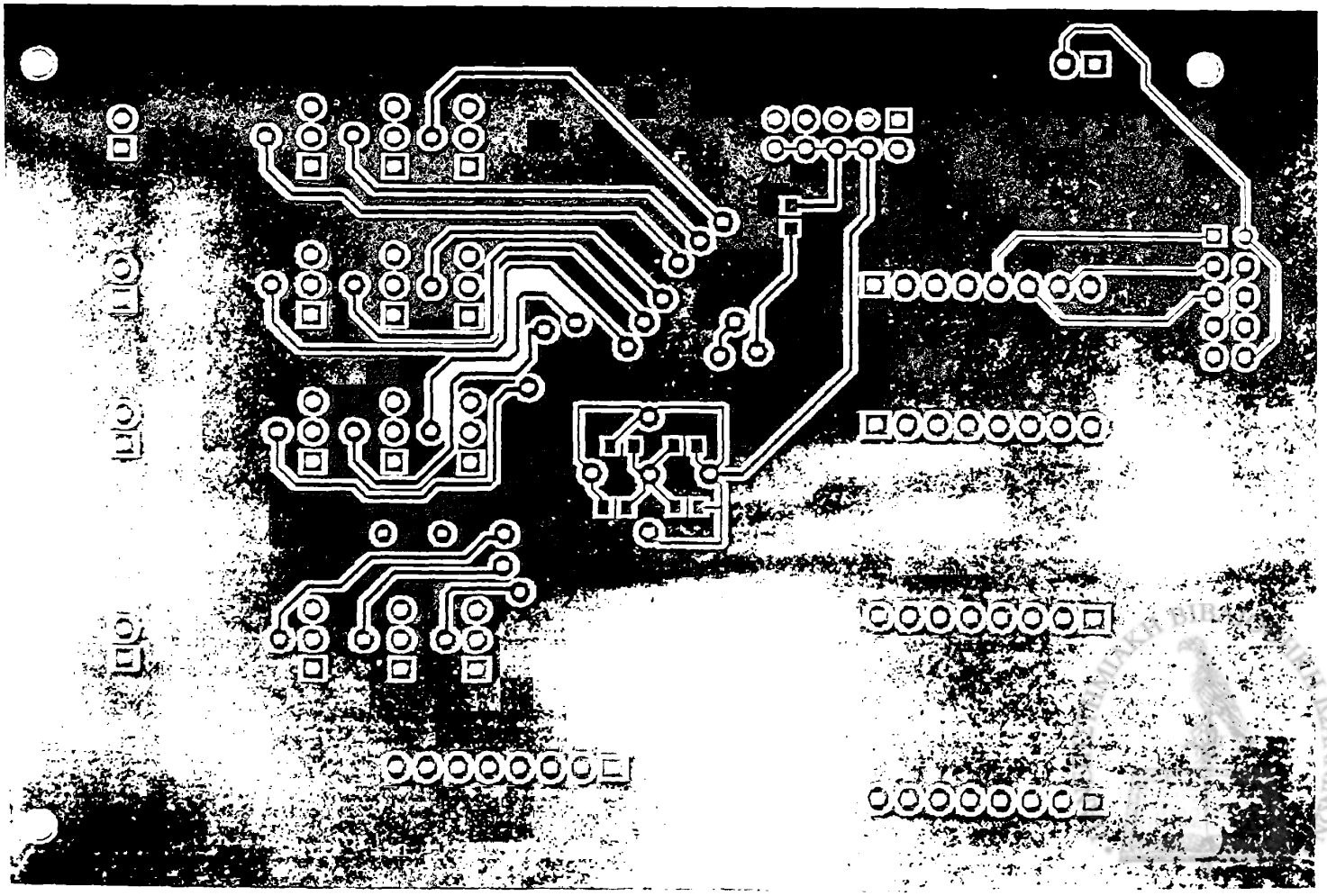


Εικόνα 108: Σχηματικό πλακέτας μικροελεγκτή



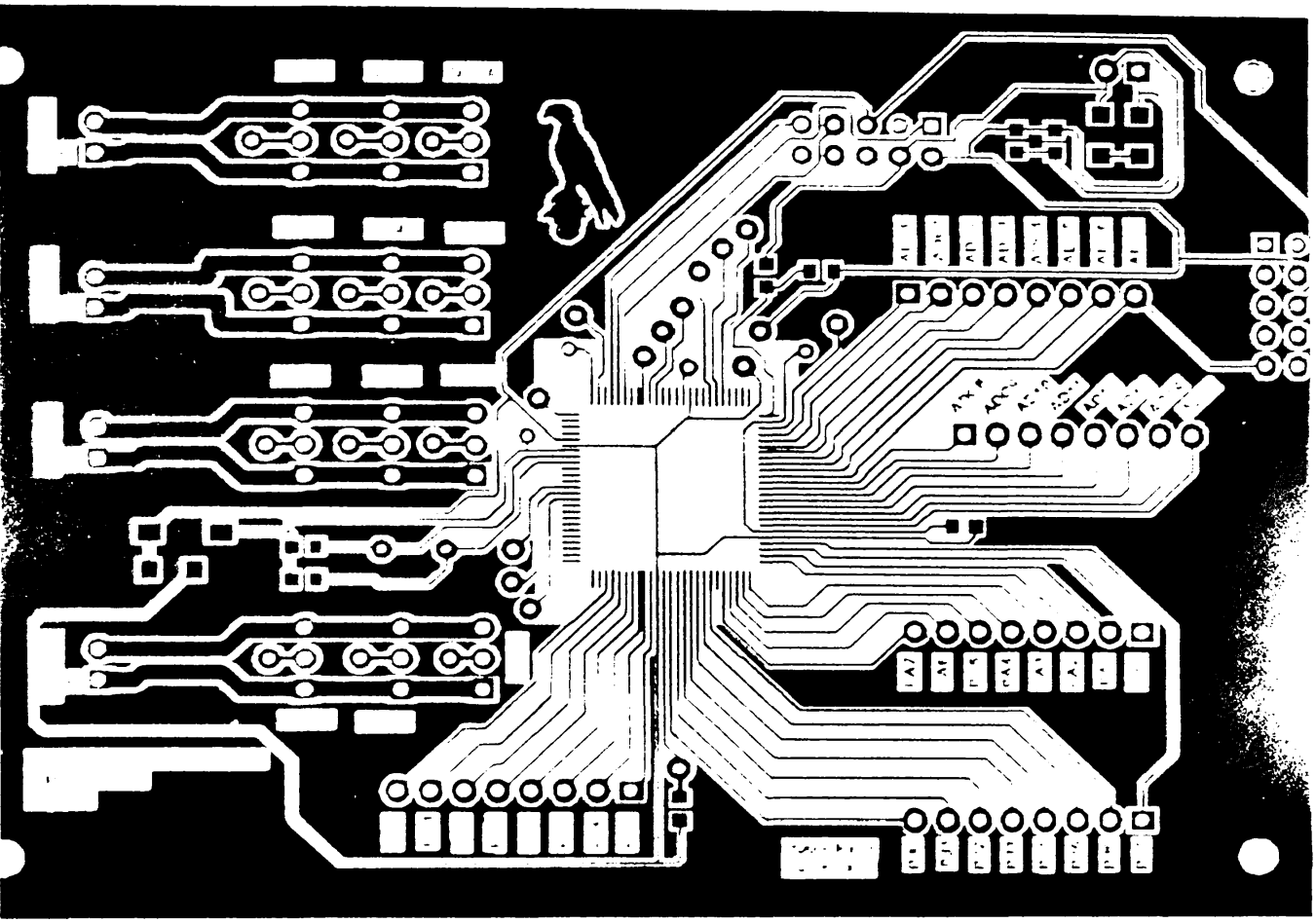
Εικόνα 109: Πλακέτα Μικροελεγκτή

σω Όψη



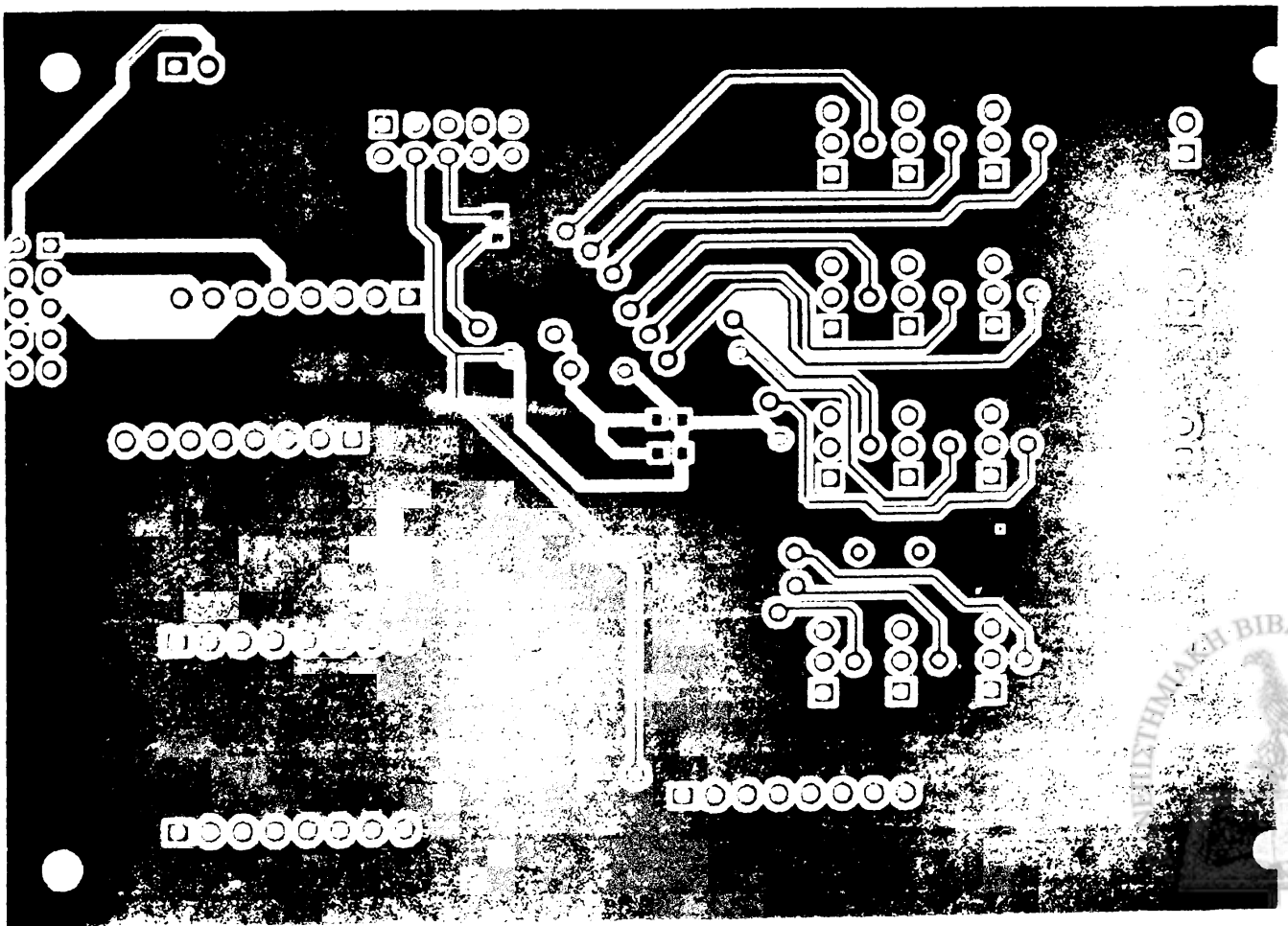
αλλαξαν καποιοι αγωγιμοι και vias που βρισκονταν κατω απο τον μικροελε

οστινή Όψη



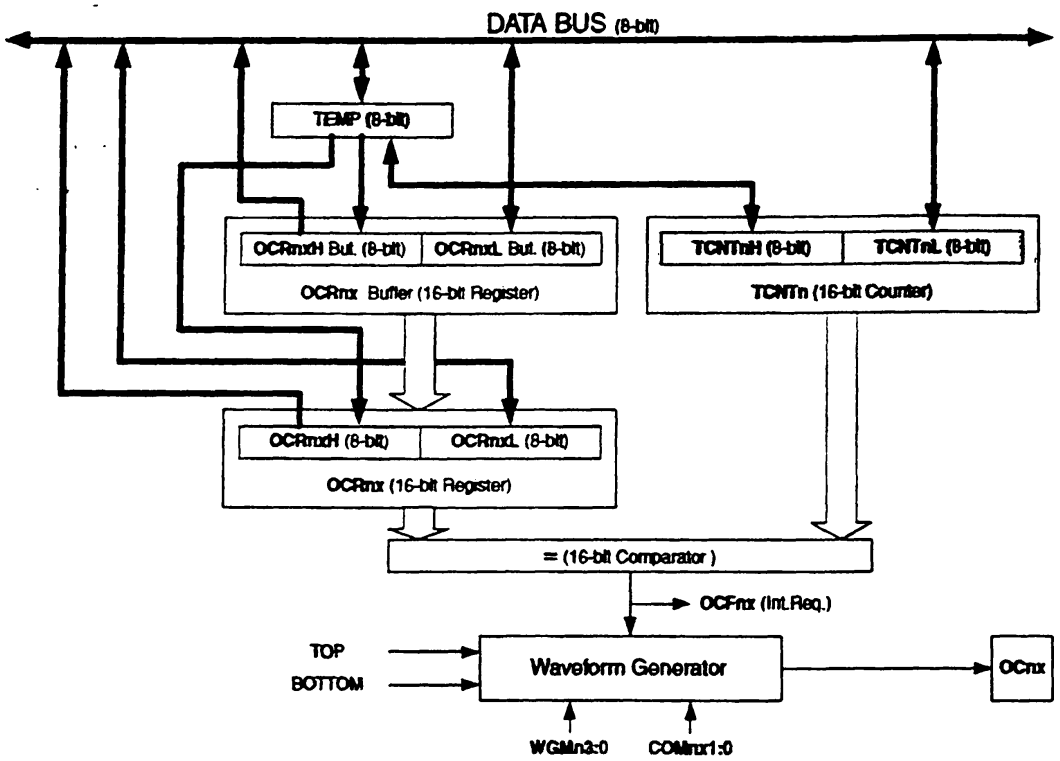
Εικονα III: [illegible]

ω Όψη



Π4 Καταχωρητές των 16-bit Χρονιστών

Σχηματικό σύγκρισης των καταχωρητών OCRnA/B/C και TCNTn



Εικόνα 113: Σύγκριση τιμών των καταχωρητών OCRnA/B/C και TCNTn

Π4.1 Επιλογή λειτουργίας Χρονιστών μέσω WGMn3:0 bits

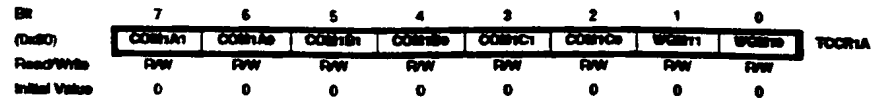
Table 17-2. Waveform Generation Mode Bit Description[1]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

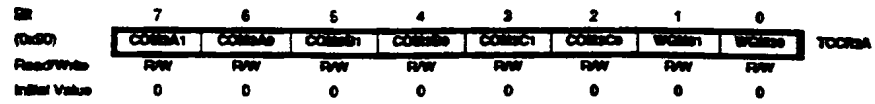
Εικόνα 114

Π4.2 Καταχωρητές Ελέγχου των χρονιστών

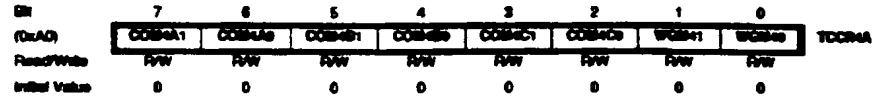
TCCR1A – Timer/Counter 1 Control Register A



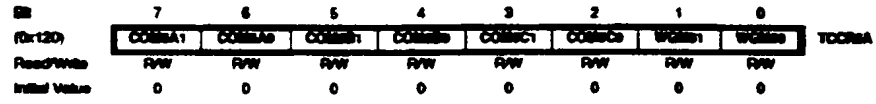
TCCR3A – Timer/Counter 3 Control Register A



TCCR4A – Timer/Counter 4 Control Register A



TCCR5A – Timer/Counter 5 Control Register A



Εικόνα 115: Καταχωρητές TCCRnA

- **Bit 7:6 – COMnA1:0: Compare Output Mode for Channel A**
- **Bit 5:4 – COMnB1:0: Compare Output Mode for Channel B**
- **Bit 3:2 – COMnC1:0: Compare Output Mode for Channel C**

The COMnA1:0, COMnB1:0, and COMnC1:0 control the output compare pins (OCnA, OCnB, and OCnC respectively) behavior. If one or both of the COMnA1:0 bits are written to one, the OCnA output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COMnB1:0 bits are written to one, the OCnB output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COMnC1:0 bits are written to one, the OCnC output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OCnA, OCnB or OCnC pin must be set in order to enable the output driver.

Table 17-4. Compare Output Mode, Fast PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM (inverting mode)

Εικόνα 116 : Mode Λειτουργιας του Fast PWM

TCCR1B – Timer/Counter 1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR3B – Timer/Counter 3 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x91)	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR4B – Timer/Counter 4 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xA1)	ICNC4	ICES4	–	WGM43	WGM42	CS42	CS41	CS40	TCCR4B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR5B – Timer/Counter 5 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x121)	ICNC5	ICES5	–	WGM53	WGM52	CS52	CS51	CS50	TCCR5B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Εικόνα 117: Καταχωρητές ελέγχου TCCRnB

- **Bit 7 – ICNCn: Input Capture Noise Canceler**

Setting this bit (to one) activates the Input Capture Noise Canceler. When the Noise Canceler is activated, the input from the Input Capture Pin (ICPn) is filtered. The filter function requires four successive equal valued samples of the ICPn pin for changing its output. The input capture is therefore delayed by four Oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICESn: Input Capture Edge Select**

This bit selects which edge on the Input Capture Pin (ICPn) that is used to trigger a capture event. When the ICESn bit is written to zero, a falling (negative) edge is used as trigger, and when the ICESn bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICESn setting, the counter value is copied into the Input Capture Register (ICRn). The event will also set the Input Capture Flag (ICFn), and this can be used to cause an Input Capture Interrupt, if this interrupt is enabled.

When the ICRn is used as TOP value (see description of the WGMn3:0 bits located in the TCCRnA and the TCCRnB Register), the ICPn is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCRnB is written.

- **Bit 4:3 – WGMn3:2: Waveform Generation Mode**

See TCCRnA Register description.

- **Bit 2:0 – CSn2:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter

Π4.2.1 Επιλογή Ρολογιού Και Prescaler για τους χρονιστών

Table 17-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Εικόνα 118: Bits επιλογής Prescaler

Π4.3 Καταχωρητές Μέτρησης των χρονιστών

TCNT1H and TCNT1L – Timer/Counter 1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCNT3H and TCNT3L – Timer/Counter 3

Bit	7	6	5	4	3	2	1	0	
(0x95)	TCNT3[15:8]								TCNT3H
(0x94)	TCNT3[7:0]								TCNT3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCNT4H and TCNT4L –Timer/Counter 4

Bit	7	6	5	4	3	2	1	0	
(0xA5)	TCNT4[15:8]								TCNT4H
(0xA4)	TCNT4[7:0]								TCNT4L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCNT5H and TCNT5L –Timer/Counter 5

Bit	7	6	5	4	3	2	1	0	
(0x125)	TCNT5[15:8]								TCNT5H
(0x124)	TCNT5[7:0]								TCNT5L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Εικόνα 119: Καταχωρητές Μέτρησης

Π4.4 Καταχωρητές σύγκρισης OCRnA/B/C

OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR1CH and OCR1CL – Output Compare Register 1 C

Bit	7	6	5	4	3	2	1	0	
(0x8D)	OCR1C[15:8]								OCR1CH
(0x8C)	OCR1C[7:0]								OCR1CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR3AH and OCR3AL – Output Compare Register 3 A

Bit	7	6	5	4	3	2	1	0	
(0x99)	OCR3A[15:8]								OCR3AH
(0x98)	OCR3A[7:0]								OCR3AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR3BH and OCR3BL – Output Compare Register 3 B

Bit	7	6	5	4	3	2	1	0	
(0x9B)	OCR3B[15:8]								OCR3BH
(0x9A)	OCR3B[7:0]								OCR3BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR3CH and OCR3CL – Output Compare Register 3 C

Bit	7	6	5	4	3	2	1	0	
(0x9D)	OCR3C[15:8]								OCR3CH
(0x9C)	OCR3C[7:0]								OCR3CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR4AH and OCR4AL – Output Compare Register 4 A

Bit	7	6	5	4	3	2	1	0	
(0xA9)	OCR4A[15:8]								OCR4AH
(0xA8)	OCR4A[7:0]								OCR4AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR4BH and OCR4BL – Output Compare Register 4 B

Bit	7	6	5	4	3	2	1	0	
(0xAA)	OCR4B[15:8]								OCR4BH
(0xAB)	OCR4B[7:0]								OCR4BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

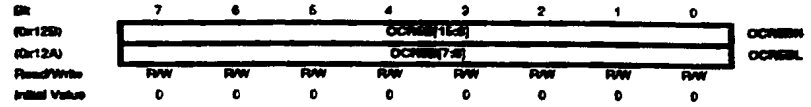
OCR4CH and OCR4CL –Output Compare Register 4 C

Bit	7	6	5	4	3	2	1	0	
(0xAD)	OCR4C[15:8]								OCR4CH
(0xAC)	OCR4C[7:0]								OCR4CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

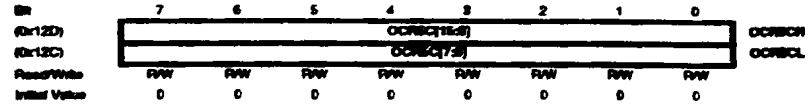
OCR5AH and OCR5AL – Output Compare Register 5 A

Bit	7	6	5	4	3	2	1	0	
(0x129)	OCR5A[15:8]								OCR5AH
(0x128)	OCR5A[7:0]								OCR5AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCR5BH and OCR5BL – Output Compare Register 5 B



OCR5CH and OCR5CL –Output Compare Register 5 C

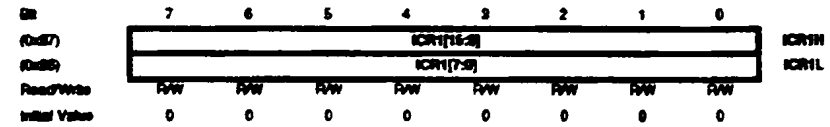


The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNTn). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OCnx pin.

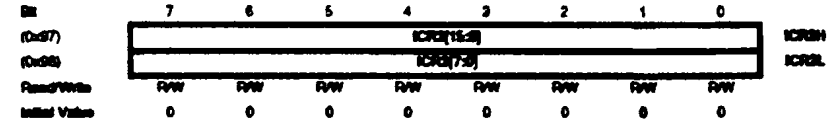
Εικόνα 120: Καταχωρητές Σύγκρισης

Π4.5 Καταχωρητές ρύθμισης της συχνότητας των PWM σημάτων

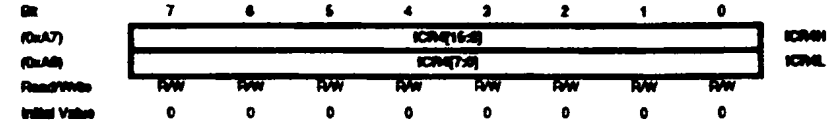
ICR1H and ICR1L – Input Capture Register 1



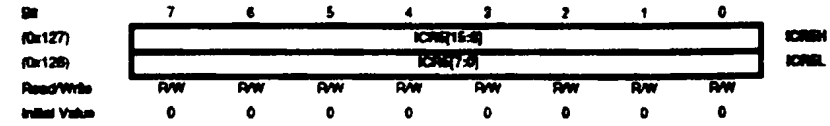
ICR3H and ICR3L – Input Capture Register 3



ICR4H and ICR4L – Input Capture Register 4



ICR5H and ICR5L – Input Capture Register 5



Εικόνα 121: Καταχωρητές Ρύθμισης της συχνότητας για PWM

The Input Capture is updated with the counter (TCNTn) value each time an event occurs on the ICPn pin (or optionally on the Analog Comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

Π4.6 Καταχωρητές ρύθμισης των interrupts

TIMSK1 – Timer/Counter 1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x6F)			ICIE1		OCIE1C	OCIE1B	OCIE1A	TOIE1
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TIMSK3 – Timer/Counter 3 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x71)			ICIE3		OCIE3C	OCIE3B	OCIE3A	TOIE3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TIMSK4 – Timer/Counter 4 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x72)			ICIE4		OCIE4C	OCIE4B	OCIE4A	TOIE4
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TIMSK5 – Timer/Counter 5 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
(0x73)			ICIE5		OCIE5C	OCIE5B	OCIE5A	TOIE5
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Εικόνα 122 : Καταχωρητές ενεργοποίησης Interrupts

- **Bit 5 – ICIEn: Timer/Countern, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Input Capture interrupt is enabled. The corresponding Interrupt Vector is executed when the ICFn Flag, located in TIFRn, is set.

- **Bit 3 – OCIEnC: Timer/Countern, Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCFnC Flag, located in TIFRn, is set.

- **Bit 2 – OCIEnB: Timer/Countern, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCFnB Flag, located in TIFRn, is set.

- **Bit 1 – OCIEnA: Timer/Countern, Output Compare A Match Interrupt Enable**

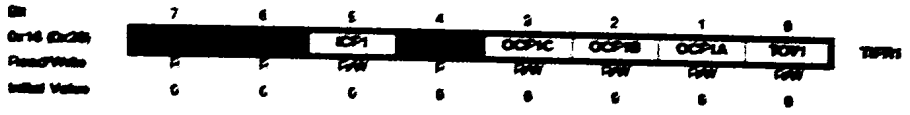
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector is executed when the OCFnA Flag, located in TIFRn, is set.

- **Bit 0 – TOIEn: Timer/Countern, Overflow Interrupt Enable**

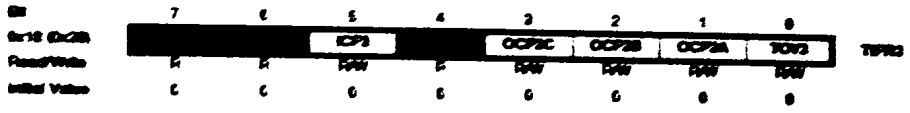
When this bit is written to one, and the I-flag in the Status Register is set (interrupts globall) enabled), the Timer/Countern Overflow interrupt is enabled. The corresponding Interrupt Vector is executed when the TOVn Flag, located in TIFRn, is set.

Π4.6.1 Καταχωρητές σημαιών των interrupts των χρονιστών

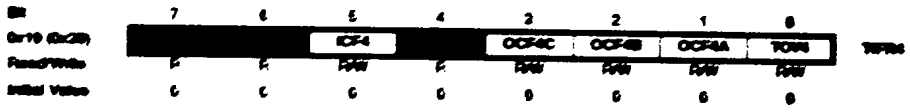
TIFR1 – Timer/Counter1 Interrupt Flag Register



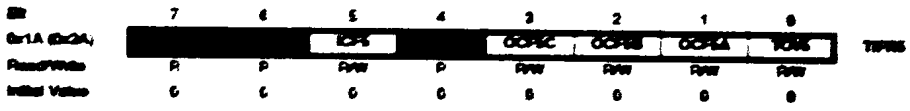
TIFR3 – Timer/Counter3 Interrupt Flag Register



TIFR4 – Timer/Counter4 Interrupt Flag Register



TIFR5 – Timer/Counter5 Interrupt Flag Register



Εικόνα 123: Καταχωρητές TIFRn

- **Bit 5 – ICFn: Timer/Countern, Input Capture Flag**

This flag is set when a capture event occurs on the ICPn pin. When the Input Capture Register (ICRn) is set by the WGMn3:0 to be used as the TOP value, the ICFn Flag is set when the counter reaches the TOP value.

ICFn is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICFn can be cleared by writing a logic one to its bit location.

- **Bit 2– OCFnC: Timer/Countern, Output Compare C Match Flag**

This flag is set in the timer clock cycle after the counter (TCNTn) value matches the Output Compare Register C (OCRnC).

Note that a Forced Output Compare (FOCnC) strobe will not set the OCFnC Flag.

OCFnC is automatically cleared when the Output Compare Match C Interrupt Vector is executed. Alternatively, OCFnC can be cleared by writing a logic one to its bit location.

- **Bit 2 – OCFnB: Timer/Counter1, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNTn) value matches the Output Compare Register B (OCRnB).

Note that a Forced Output Compare (FOCnB) strobe will not set the OCFnB Flag.

OCFnB is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCFnB can be cleared by writing a logic one to its bit location.

- **Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNTn value matches the Output Compare Register A (OCRnA).

Note that a Forced Output Compare (FOCnA) strobe will not set the OCFnA Flag.

OCFnA is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCFnA can be cleared by writing a logic one to its bit location.

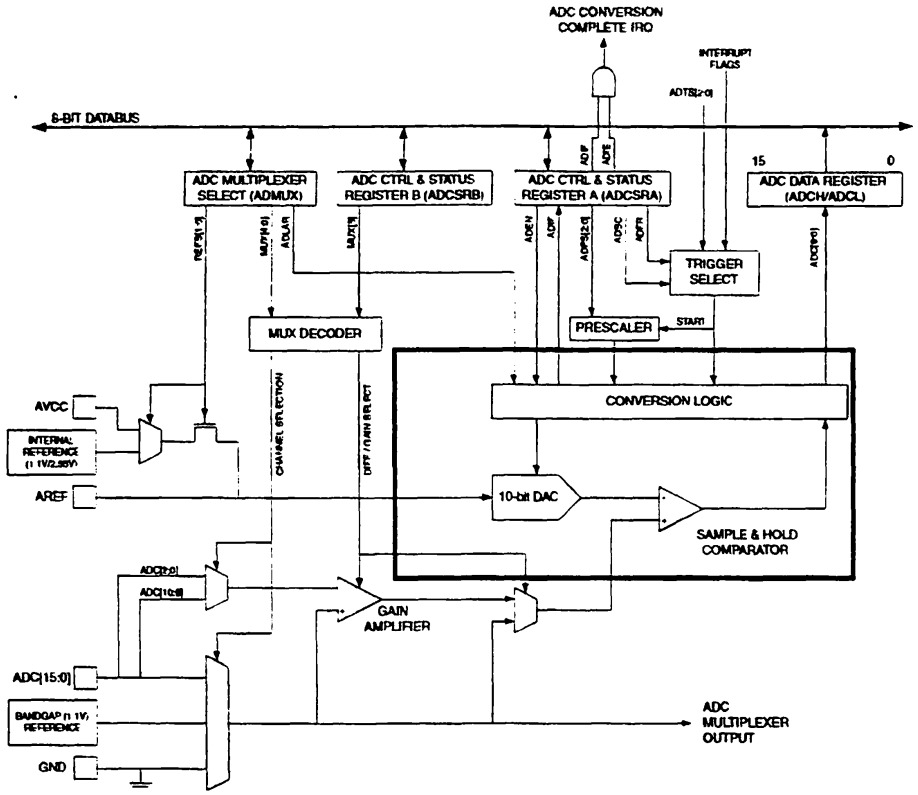
- **Bit 0 – TOVn: Timer/Countern, Overflow Flag**

The setting of this flag is dependent of the WGMn3:0 bits setting. In Normal and CTC modes, the TOVn Flag is set when the timer overflows.

TOVn is automatically cleared when the Timer/Countern Overflow Interrupt Vector is executed. Alternatively, TOVn can be cleared by writing a logic one to its bit location.

Π5 Καταχωρητές του ADC του μικροελεγκτή Atmega2560

Π5.1 Σχηματικό Διάγραμμα του ADC



Εικόνα 124 : Κύκλωμα ADC

Π5.1.1 Μέθοδος Διαδοχικής Προσέγγισης (Succesive Aproximation)

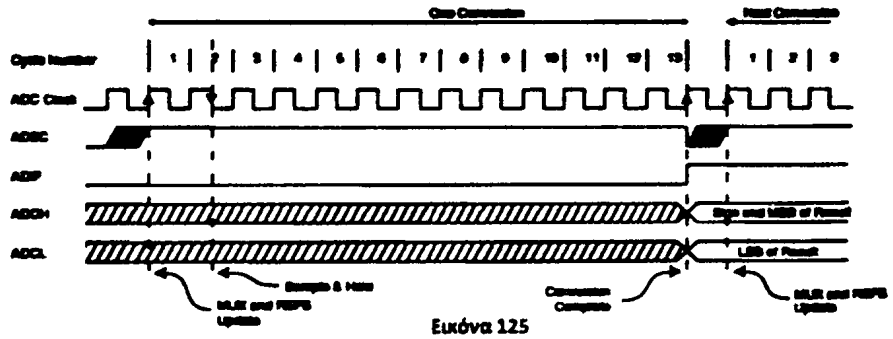
Με τη μέθοδο S.A. (εικόνα 124) ο μικροελεγκτής μπορεί να κάνει μετατροπές αναλογικού σήματος σε ψηφιακό. Η μέθοδος αυτή χρησιμοποιεί ένα μετατροπέα Ψηφιακού σήματος σε Αναλογικό (DAC) 10-bit, ένα καταχωρητή 10-bit (SAR) στον οποίο αποθηκεύεται μία διαδικασία διαδοχικής σύγκρισης, ένα ελεγκτή (μικροελεγκτής) και ένα συγκριτή δειγματοληψίας (Sample & Hold). Η διαδικασία είναι η εξής: Αφού ο μικροελεγκτής αρχικοποιήσει τον ADC ξεκινάει μία μετατροπή. Κατά τη μετατροπή αυτή ο SAR μηδενίζεται και στη συνέχεια κάνει λογικό "1" το πιο σημαντικό του bit (10^{o} bit - MSB). Στη συνέχεια, αυτή η ψηφιακή τιμή του SAR (0b1000000000 → 512) μεταφέρεται στην είσοδο του DAC. Καθώς ο DAC έχει την ίδια τάση αναφοράς V_{REF} με αυτή του ADC, αναγνωρίζει την τιμή 512 ως το μισό της V_{REF}. Έτσι η ψηφιακή τιμή μετατρέπεται σε αναλογική τάση 2.5V στην έξοδο του DAC. Στη συνέχεια, αυτή η τιμή (2.5V) μεταφέρεται στην αναστρέφουσα είσοδο του

συγκριτή δειγματοληψίας και συγκρίνεται με την αναλογική τάση του καναλιού που πρόκειται να μετατραπεί σε ψηφιακή, η οποία βρίσκεται στη μη αναστρέφουσα είσοδο του συγκριτή αυτού.

Η έξοδος του συγκριτή πηγαίνει ως είσοδος στον καταχωρητή SAR. Εάν η τιμή του καναλιού προς μετατροπή είναι μεγαλύτερη από την τιμή του DAC (2.5V), η έξοδος του συγκριτή γίνεται λογικό "1" και το bit που ήταν "1" στον SAR μένει όπως είναι. Διαφορετικά, η έξοδος του συγκριτή γίνεται λογικό "0" και το bit του SAR που ήταν "1" γίνεται τώρα "0".

Στη συνέχεια, το 9° bit του SAR γίνεται "1". Η νέα ψηφιακή τιμή του SAR μεταφέρεται στον DAC. Ο DAC με τη σειρά του μετατρέπει αυτή την τιμή σε αναλογική τάση και η διαδικασία επαναλαμβάνεται μέχρι να γίνει "1" και το τελευταίο bit του SAR (1° bit - LSB). Με το τέλος της διαδικασίας αυτής, η ψηφιακή τιμή του SAR μας δίνει και το αποτέλεσμα της μετατροπής της αναλογικής τάσης του καναλιού σε ψηφιακή τιμή.

Π5.2 Διάγραμμα χρόνου μετατροπής του ADC

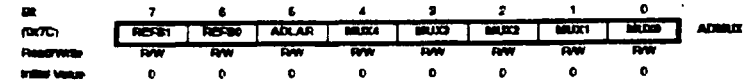


Εικόνα 125

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Εικόνα 126 : Χρόνοι μετατροπής

ADMUX – ADC Multiplexer Selection Register



- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 26-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Εικόνα 127: Καταχωρητής Επιλογής Vref και καναλιού

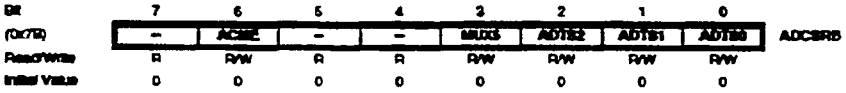
- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 26-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

ADCSRB – ADC Control and Status Register B



- **Bit 3 – MUX5: Analog Channel and Gain Selection Bit**

This bit is used together with MUX4:0 in ADMUX to select which combination in of analog inputs are connected to the ADC. See Table 26-4 for details. If this bit is changed during a conversion, the change will not go in effect until this conversion is complete.

This bit is not valid for ATmega1281/2561.

Εικόνα 128: Καταχωρητής ελέγχου B

Π5.2.1 Επιλογή καναλιών του ADC

Input Channel Selections

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			
000110	ADC6			
000111	ADC7			
100000	ADC8	N/A		
100001	ADC9			
100010	ADC10			
100011	ADC11			
100100	ADC12			
100101	ADC13			
100110	ADC14			
100111	ADC15			

Εικόνα 129: Επιλογή Καναλιών

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Εικόνα 130: Επιλογή Prescaler

Π5.3 Καταχωρητής αποθήκευσης του αποτελέσματος της μετατροπής

ADCL and ADCH – The ADC Data Register

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision (7 bit + sign bit for differential input channels) is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 288.

Εικόνα 131

Π5.4 Επιλογή Τρόπου Λειτουργίας του ADC

ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when ADCSRB is written.

- **Bit 2:0 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trig-

ger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 26-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Εικόνα 132: Bits επιλογής λειτουργίας του ADC

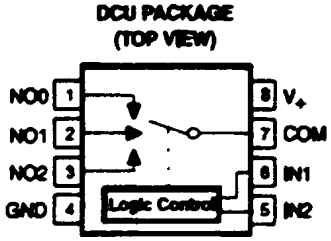
Π6 Δεύτερη (εναλλακτική) Υλοποίηση του συστήματος

Π6.1 Λειτουργία μεθόδου

Σε αυτό το σημείο γίνεται αναφορά στον δεύτερο τρόπο με τον οποίο υλοποιήθηκε το σύστημα ελέγχου της ρομποτικής χείρας.

Για την πραγματοποίηση του ελέγχου της ρομποτικής χείρας ήταν απαραίτητα τουλάχιστον 18 σήματα PWM. Το πρόβλημα όμως εδώ ήταν ότι ο μικροελεγκτής ATmega2560 μπορεί να βγάλει στις εξόδους του μέχρι 12 σήματα PWM.
Έτσι, θα έπρεπε να χρησιμοποιηθούν δύο μικροελεγκτές (όπως και έγινε τελικά) ή να χρησιμοποιηθεί ένας μικροελεγκτής συνδεδεμένος με ένα εξωτερικό κύκλωμα το οποίο θα έδινε τα επιπλέον 6 σήματα PWM που απαιτούνταν για την οδήγηση των κινητήρων.

Για την υλοποίηση της δεύτερης μεθόδου χρησιμοποιήθηκαν , ένας μικροελεγκτής ATmega2560 και τρείς De-multiplexers SP3T (1:3) οι οποίοι μπορούνε να μεταφέρουν το σήμα που βρίσκεται στην είσοδο τους σε τρείς διαφορετικές εξόδους ο καθένας, μέσω δύο bit ελέγχου. Οι De-multiplexers που χρησιμοποιήθηκαν είναι της εταιρίας Texas Instruments με κωδικό TS5A3359 [15] και φαίνονται στην εικόνα 133.



Εικόνα 133α: Ακροδέκτες DeMUX

FUNCTION TABLE

IN2	IN1	COM TO NO, NO TO COM
L	L	OFF
L	H	COM = NO0
H	L	COM = NO1
H	H	COM = NO2

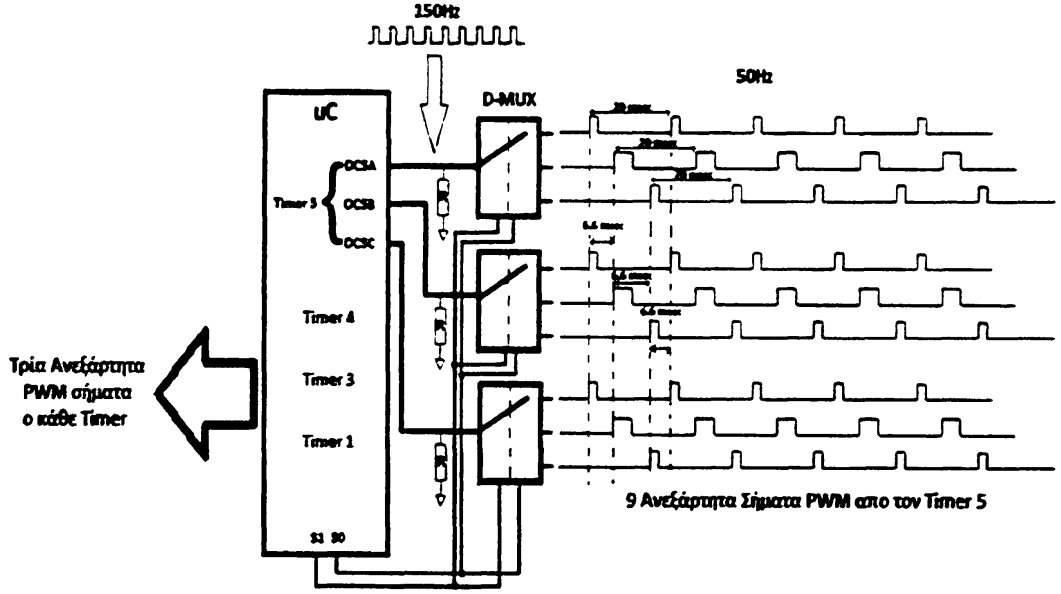
Εικόνα 133β: Πίνακας Αληθείας

Η λογική της μεθόδου αυτής ήταν ότι, τρία σήματα PWM προερχόμενα από τρείς εξόδους του μικροελεγκτή μπορούνε να βγάλουν εννέα σήματα PWM τελείως ανεξάρτητα το ένα από το άλλο, δίνοντας έτσι την δυνατότητα οδήγησης εννέα σέρβο-κινητήρων στις εξόδους .Δηλαδή ο κάθε Demultiplexer θα οδηγούσε τρείς σέρβο-κινητήρες.

Κάθε ένας 16-bit χρονιστής μπορεί να παράγει μέχρι τρία σήματα PWM. Έτσι στην κάθε έξοδο PWM αυτού του χρονιστή τοποθετήθηκαν και από ένας Demultiplexer. Από τις τρείς εξόδους του κάθε De-multiplexer θα παίρνουμε τρία διαφορετικά σήματα PWM.Άρα συνολικά θα παράγονται 9 σήματα PWM.

Στο τέλος κάθε περιόδου, του ενός από τα τρία σήματα που παράγει ο ένας χρονιστής, ο Demultiplexer θα αλλάζει την έξοδο στην οποία μεταφέρεται το σήμα αυτό κυκλικά, μέσω των δύο bits ελέγχου του και έτσι θα τροφοδοτεί και τις τρείς εξόδους του συνεχώς με σήματα PWM. Το πρόβλημα σε αυτό το σημείο είναι πως

αφού το σήμα PWM που παράγεται από την μία έξοδο του χρονιστή πηγαίνει σε τρείς εξόδους , μοιράζεται δηλαδή σε τρία μέρη, το ίδιο συμβαίνει και με τη συχνότητα του. Δηλαδή τα 50Hz που είναι η απαραίτητη συχνότητα για την οδήγηση των σέρβο-κινητήρων και η οποία βγαίνει από το αρχικό σήμα PWM στην έξοδο του μικροελεγκτή, θα επιμερίζεται σε τρία μέρη και θα γίνεται 16.6Hz. Δηλαδή σε κάθε σέρβο-κινητήρα το σήμα PWM θα έχει συχνότητα 16.6Hz, κάτι που είναι εντελώς λάθος εφόσον ο σέρβο-κινητήρας λειτουργεί στα 50Hz. Για να διορθωθεί αυτό το σφάλμα, αυτό που μπορεί να γίνει είναι να αυξηθεί η συχνότητα του αρχικού σήματος PWM, το οποίο βγαίνει από την έξοδο του χρονιστή του μικροελεγκτή, στο τριπλάσιο. Δηλαδή το αρχικό σήμα θα έχει συχνότητα 150Hz. Έτσι τα παραγόμενα σήματα από τον Demultiplexer θα έχουν συχνότητα 50Hz που είναι και η σωστή συχνότητα για την οδήγηση των κινητήρων. Στην επόμενη εικόνα παρουσιάζεται η λειτουργία του συστήματος με αυτή τη μέθοδο.



Εικόνα 134: Κύκλωμα παραγωγής 9 PWM σημάτων απο τρία

Τα τρία αρχικά σήματα PWM που παράγονται από τον χρονιστή (έστω Timer 5) έχουν την ίδια συχνότητα (150Hz) εφόσον ανήκουν στον ίδιο χρονιστή. Τα bits S1 και S0 είναι τα bits επιλογής της εξόδου των De-multiplexers και τα οποία είναι κοινά και για τους τρείς De-multiplexers. Τα bits αυτά θα αλλάζουν από "01" σε "10" σε "11" σηματοδοτώντας την αλλαγή των διακοπτών των De-multiplexers.

Στους καταχωρητές του χρονιστή για την παραγωγή των σημάτων PWM με συχνότητα **150Hz** και πλάτη παλμών **900usec -2 msec**, βάλαμε τις εξής τιμές

Στον καταχωρητή ICR5 που είναι υπεύθυνος για τη μέγιστη τιμή στην οποία θα φτάνει ο χρονιστής καθώς θα μετράει, και άρα θα παράγει τα 150Hz, η τιμή του βρίσκεται από τον τύπο

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

Με F_{clk} = 16MHz, N=Prescaler=8, F_{OC5}=150Hz

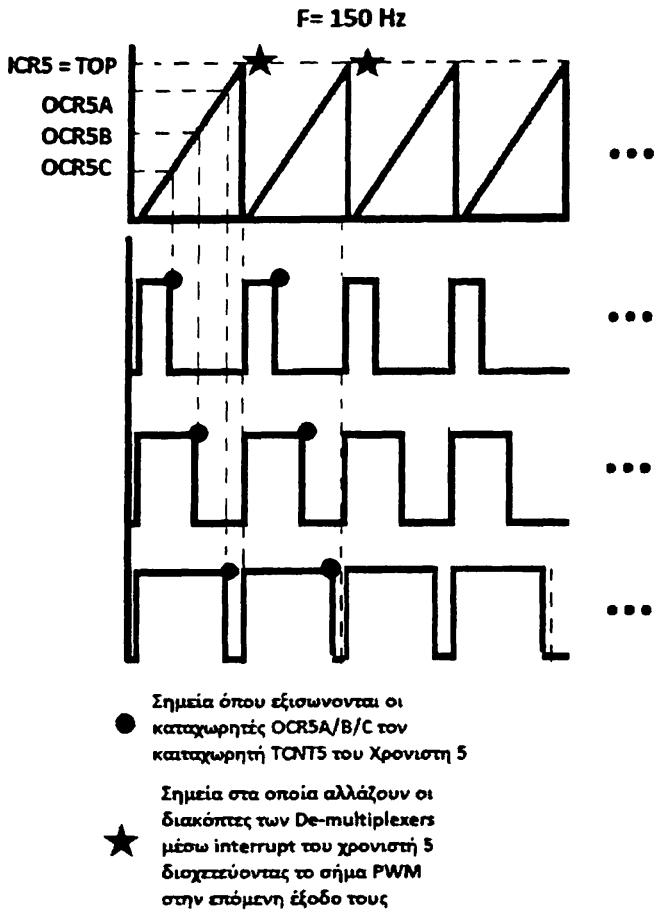
Έτσι η τιμή του καταχωρητή βρίσκεται να είναι 13332. Δηλαδή **ICR5 = 13332** (0x3414 σε hex).

Στους καταχωρητές OCR5A/B/C βάζουμε τις τιμές 3999 και 1999 για 2msec και 900usec αντίστοιχα όπως προκύπτουν και για την πρώτη μέθοδο. Δηλαδή **OCR5A/B/C = 1999** και **OCR5A/B/C = 3999** (0x07CF και 0x0F9F σε hex αντίστοιχα).

Κάθε φορά που ο χρονιστής φτάνει την τιμή 13332 , μηδενίζεται και ξεκινάει από την αρχή . Όταν ο καταχωρητής μέτρησης του χρονιστή 5, TCNT5, εξισωθεί με ένα από τους καταχωρητές σύγκρισης OCR5A/B/C, το σήμα εξόδου γίνεται από λογικο¨1¨ σε λογικό ¨0¨ και ξαναγίνεται λογικό ¨1¨ μόλις ο χρονιστής ξαναφτάσει στο μηδέν. Έτσι παράγεται το PWM σήμα συχνότητας 150 Hz.

Κάθε φορά που ο TCNT5 εξισώνεται με τους OCR5A/B/C ενεργοποιείται και από μία διακοπή (συνολικά 3 interrupts) και μέσω software ανανεώνουμε τις τιμές των OCR5A/B/C με τις τιμές που παίρνουμε από τις συναρτήσεις όπως αναλύθηκε στο 6° κεφάλαιο. Όταν ο χρονιστής εξισωθεί με την μέγιστη τιμή που μπορεί να πάρει δηλαδή TCNT5 =ICR5 = TOP τότε παράγεται μία διακοπή (interrupt) μέσα στην οποία αλλάζουμε τα bits επιλογής S1 και S0 των Demultiplexers. Έτσι κάθε φορά που ο χρονιστής 5 φτάνει στο τέλος της περιόδου του αλλάζει τους διακόπτες των Demultiplexers παράγοντας τα εννέα διαφορετικά σήματα PWM συχνότητας 50 Hz.

Στην επόμενη εικόνα παρουσιάζεται η λειτουργία του χρονιστή όπως αναλύθηκε προηγουμένως καθώς και τα σημεία που (κόκκινοι κύκλοι) που παράγονται οι διακοπές των καταχωρητών TCNT5 και OCR5A/B/C.



Εικόνα 135: Χρονικά Σημεία αποθήκευσης νέων τιμών στους OCRnA/B/C

Με το πέρας της υλοποίησης της δεύτερης μεθόδου ελέγχου του συστήματος δοκιμάστηκαν και οι δύο μέθοδοι (Δύο μικροελεγκτές, Ένας μικροελεγκτής –Τρείς De-multiplexers) και αποφασίστηκε η τελική υλοποίηση του συστήματος ελέγχου να γίνει με την πρώτη μέθοδο.

Οι λόγοι που οδήγησαν σε αυτή την επιλογή ήταν οι εξής:

Πρώτον, με τη χρήση De-multiplexers και ενός μικροελεγκτή, παρατηρήθηκε πως ενώ τα εννέα παραγόμενα σήματα PWM στις εξόδους τους ακολουθούσαν τούς χρονικούς περιορισμούς των 900usec – 2msec για την οδήγηση των κινητήρων (όπως παρατηρήθηκαν στον παλμογράφο) ,όταν συνδεόταν οι σέρβο-κινητήρες κάποιοι από αυτούς άρχιζαν να τρέμουν έντονα ενώ οι υπόλοιποι καθόλου. Δοκιμάστηκαν διάφοροι μέθοδοι για την εξάλειψη του σφάλματος αυτού, όπως τοποθέτηση πυκνωτών για την απαλοιφή πιθανού θορύβου στις εισόδους του σήματος των σέρβο-κινητήρων καθώς και τοποθέτηση pull-down αντιστάσεων σε αυτά τα σημεία γιατί όταν οι διακόπτες των De-multiplexers άλλαζαν, οι προηγούμενες έξοδοι ήταν σε κατάσταση idle και κάνανε float (άγνωστη κατάσταση μεταξύ λογικού ¨1¨ και λογικού¨0¨). Ύστερα αλλάχθηκαν οι σέρβο-κινητήρες της ρομποτικής χείρας που έτρεμαν , στην περίπτωση που κάτι έφταιγε με τους ίδιους τους σέρβο-κινητήρες καθώς, το πρόβλημα αυτό δεν εμφανιζόταν και στους εννέα. Δυστυχώς καμία από τις πιο πάνω προσπάθειες δεν ευδοκίμησαν.

Αντίθετα με την χρήση των δύο μικροελεγκτών οι σέρβο-κινητήρες της ρομποτικής χείρας δεν έτρεμαν αλλά λειτουργούσαν ομαλά.

Δεύτερον, με τη χρήση Demultiplexers και ενός μικροελεγκτή, παρατηρήθηκε πως περίπου μετά από ένα λεπτό από την ενεργοποίηση του συστήματος, και τα εννέα παραγόμενα PWM σήματα αποσυντονίζονταν μη μπορώντας να κρατήσουν τα απαραίτητα χρονικά όρια 900μsec και 2msec για την οδήγηση των κινητήρων. Μετά από ένα λεπτό τα σήματα λειτουργούσαν ξανά κανονικά. Αυτό το φαινόμενο επαναλαμβανόταν συνεχώς. Αυτό προφανώς ήταν κάποιο λάθος που έγινε κατά τον προγραμματισμό του μικροελεγκτή. Το πρόγραμμα τελικά διορθώθηκε έμμεσα με την χρήση του Watchdog timer μέσω του οποίου εκτελούνταν επανεκκίνηση όλου του συστήματος κάθε ένα λεπτό. Με αυτό τον τρόπο τα σήματα δεν παρουσίαζαν πλέον αυτό το πρόβλημα. Παρουσιάστηκε όμως ένα άλλο πρόβλημα με τη χρήση της επανεκκίνησης μέσω software. Κάθε φορά που γινόταν επανεκκίνηση του συστήματος οι χρονιστές, που παρήγαγαν τα σήματα PWM, μηδενίζονταν και ξανά αρχικοποιούνταν. Αυτό είχε ως αποτέλεσμα τον απότομο αποσυντονισμό των σέρβο-κινητήρων για ένα πάρα πολύ μικρό χρονικό διάστημα (λιγότερο από 1 sec). Αυτό όμως αν και φαινομενικά δεν ήταν τόσο σοβαρό, αν εκείνη τη στιγμή η ρομποτική χείρα κρατούσε κάποιο αντικείμενο, οι κινητήρες με τον απότομο αποσυντονισμό πιθανότατα (σίγουρα) θα άφηναν το αντικείμενο να πέσει κάτω. Φυσικά αυτό δεν είναι ένα αμελητέο σφάλμα.

Αντίθετα με την χρήση των δύο μικροελεγκτών αυτά τα προβλήματα δεν παρουσιάστηκαν και όλα εξελίχθηκαν ομαλά κατά τις δοκιμές λειτουργίας.

Π6.2 Κώδικας μικροελεγκτή δεύτερης μεθόδου

```
/*
 * final_beta.c
 *
 * Created: 14/2/2014 1:47:01 μμ
 *  Author: Lefteris
 */

#define F_CPU 16000000UL
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
#include <math.h>
#include <avr/wdt.h>


#define OCR_HIGH 0x0F                    // Arxikh 8esh Servo High Byte
#define OCR_LOW 0x9F                     // Arxikh 8esh Servo Low  Byte
#define Timer_Counter_Control_Reg_A  0xAA
#define Timer_Counter_Control_Reg_B  0x1A  // Fast PWM MODE_14, Prescaler 8
#define Freq_H_50Hz 0x9C                   // 50Hz frequency High Byte
#define Freq_L_50Hz 0x3F
                                           // 50Hz frequency Low  Byte
#define Freq_H_150Hz 0x34
                                           // 150Hz frequency High Byte
```

```
#define Freq_L_150Hz 0x14
                                        // 150Hz frequency Low  Byte

#define Katw_Akro_900_usec 1400

volatile char g ;
volatile int k;
volatile char w ;

void WDT_Init(void)
{
   cli();    //disable interrupts

   wdt_reset();  //reset watchdog

   WDTCSR = (1<<WDCE)|(1<<WDE);       //set up WDT interrupt

   WDTCSR =(1<<WDE)|(1<<WDP3); //(1<<WDIE)  //Start watchdog timer with 4s prescaller

    sei();       //Enable global interrupt


}


uint16_t ADC_metatroph (unsigned int i)  // synarthsh metatrophs
{
uint16_t adc_result = 0;
uint8_t adc_low_byte = 0;

ADMUX = 0x40+i;
ADCSRA |= (1<<ADSC)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
while((ADCSRA & (1<<ADIF))==0);
adc_low_byte = ADCL;
adc_result = ADCH<<8 | adc_low_byte;
return adc_result ;
}

uint16_t ADC_metatroph_2 (unsigned int j)
{
uint16_t adc_result = 0;
uint8_t adc_low_byte = 0;
ADMUX = 0x40+j;
ADCSRA |= (1<<ADSC)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
while((ADCSRA & (1<<ADIF))==0);
adc_low_byte = ADCL;
adc_result = ADCH<<8 | adc_low_byte;
return adc_result ;
}

int main(void)
{
DDRC = 0xFF;
DDRA = 0xFF;
DDRG = 0x03;
DDRD = 0x80;   // TEST POINT
DDRB |= (1<<5) | (1<<6) | (1<<7);        // PWMs(A/B/C) Timer1 output enable
DDRE |= (1<<3) | (1<<4) | (1<<5);        // PWMs(A/B/C) Timer3 output enable
DDRH |= (1<<3) | (1<<4) | (1<<5);        // PWMs(A/B/C) Timer4 output enable
DDRL |= (1<<3) | (1<<4) | (1<<5);        // PWMs(A/B/C) Timer5 output enable
```

```
//OCR1AH = OCR_HIGH; //========================== Timer 1 A/B/C initialize
//OCR1AL = OCR_LOW;
//OCR1BH = OCR_HIGH;
//OCR1BL = OCR_LOW;
//OCR1CH = OCR_HIGH;
//OCR1CL = OCR_LOW;
TCCR1A = Timer_Counter_Control_Reg_A;
TCCR1B = Timer_Counter_Control_Reg_B;
ICR1H = Freq_H_50Hz;
ICR1L = Freq_L_50Hz;

//OCR3AH = OCR_HIGH; //========================== Timer 3 A/B/C initialize
//OCR3AL = OCR_LOW;
//OCR3BH = OCR_HIGH;
//OCR3BL = OCR_LOW;
//OCR3CH = OCR_HIGH;
//OCR3CL = OCR_LOW;
TCCR3A = Timer_Counter_Control_Reg_A;
TCCR3B = Timer_Counter_Control_Reg_B;
ICR3H = Freq_H_50Hz;
ICR3L = Freq_L_50Hz;

//OCR4AH = OCR_HIGH; //========================== Timer 4 A/B/C initialize
//OCR4AL = OCR_LOW;
//OCR4BH = OCR_HIGH;
//OCR4BL = OCR_LOW;
//OCR4CH = OCR_HIGH;
//OCR4CL = OCR_LOW;
TCCR4A = Timer_Counter_Control_Reg_A;
TCCR4B = Timer_Counter_Control_Reg_B;
ICR4H = Freq_H_50Hz;
ICR4L = Freq_L_50Hz;

//OCR5AH = OCR_HIGH; //========================== Timer 5 A/B/C initialize
//OCR5AL = OCR_LOW;
//OCR5BH = OCR_HIGH;
//OCR5BL = OCR_LOW; // <<========= oi arxikes times gia A/B/C prepei na allaxtoun
//OCR5CH = OCR_HIGH;
//OCR5CL = OCR_LOW;
TCCR5A = Timer_Counter_Control_Reg_A;
TCCR5B = Timer_Counter_Control_Reg_B;
ICR5H = Freq_H_150Hz;
ICR5L = Freq_L_150Hz;

//TCNT0 = 0x00;
//TCCR0A = 0x00;
//TCCR0B = 0x05;
//TIMSK0 = (1<<TOIE0);

TIMSK5 =  (1<<ICIE5) ;
TIMSK4 = (1<<OCIE4C) | (1<<OCIE4B) | (1<<OCIE4A) | (1<<ICIE4) ;
TIMSK3 = (1<<OCIE3C) | (1<<OCIE3B) | (1<<OCIE3A) | (1<<ICIE3) ;
TIMSK1 = (1<<OCIE1C) | (1<<OCIE1B) | (1<<OCIE1A) | (1<<ICIE1) ;

ADCSRA |= (1<<ADEN);


sei();
```

```
while(1)
{

  }
}

ISR(TIMER1_COMPA_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL0;

ADCSRB &= 0x00;              // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result =  ADC_metatroph(0);          //ADC 0
PRAXI_DIGITAL0 = Katw_Akro_900_usec + (adc_result*4);
OCR1A = (uint16_t) PRAXI_DIGITAL0;   // Vale sth 8esh tou pinaka pou einai h dieuthinsi (tou
                                      //OCRnX) pou deixnei o pointer to apotelesma ths
                                      //praxhs_digital

}

ISR(TIMER1_COMPB_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL1;

ADCSRB &= 0x00;    // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result =  ADC_metatroph(1);  //ADC 1
PRAXI_DIGITAL1 = Katw_Akro_900_usec + (adc_result*4);
OCR1B = (uint16_t) PRAXI_DIGITAL1; // Vale sth 8esh tou pinaka pou einai h dieuthinsi (tou
                                      //OCRnX) pou deixnei o pointer to apotelesma ths
                                      //praxhs_digital

}

ISR(TIMER1_COMPC_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL2;

ADCSRB &= 0x00;       // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result =  ADC_metatroph(2);  //ADC 2
PRAXI_DIGITAL2 = Katw_Akro_900_usec + (adc_result*4);
OCR1C = (uint16_t) PRAXI_DIGITAL2;
}
ISR(TIMER3_COMPA_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL3;

ADCSRB &= 0x00;              // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result =  ADC_metatroph(3);  //ADC 3
PRAXI_DIGITAL3 = Katw_Akro_900_usec + (adc_result*4);
OCR3A = (uint16_t) PRAXI_DIGITAL3;

}

ISR(TIMER3_COMPB_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL4;
```

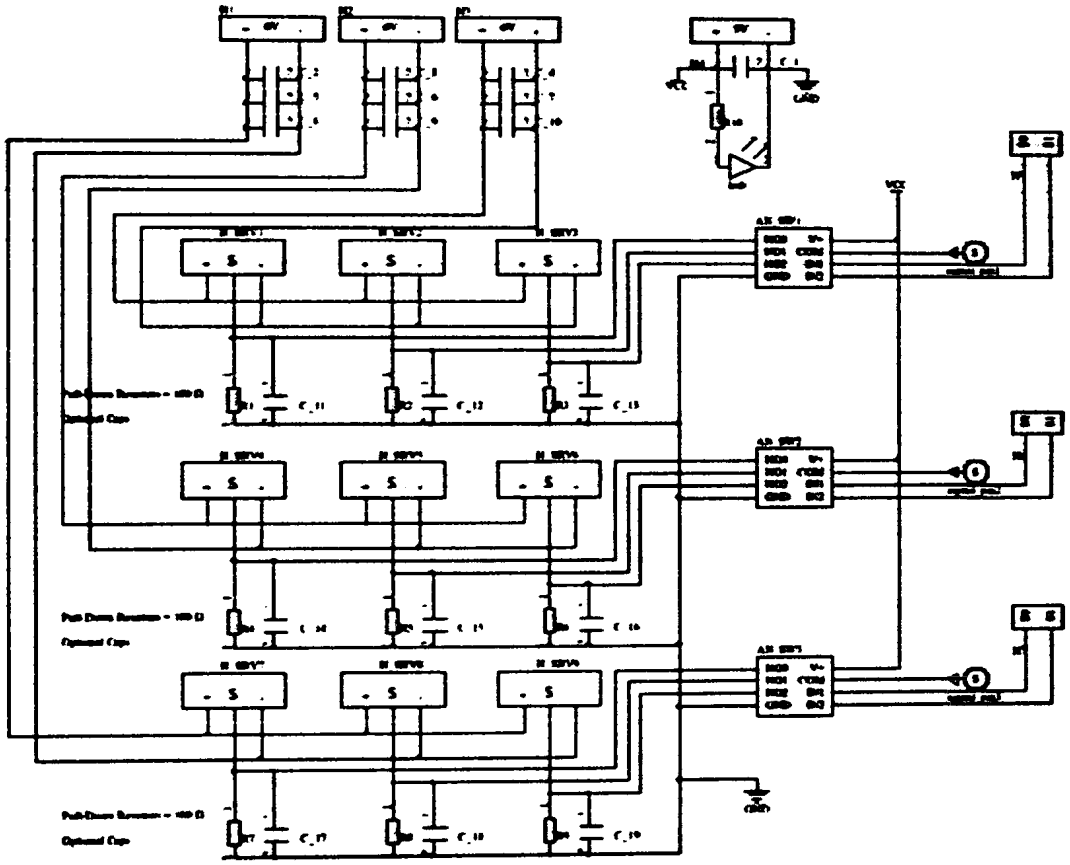
```
ADCSRB &= 0x00;     // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result = ADC_metatroph(4); //ADC 4
PRAXI_DIGITAL4 = Katw_Akro_900_usec + (adc_result*4);
OCR3B = (uint16_t) PRAXI_DIGITAL4;
}

ISR(TIMER3_COMPC_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL5;

ADCSRB &= 0x00;        // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result = ADC_metatroph(5); //ADC 5
PRAXI_DIGITAL5 = Katw_Akro_900_usec + (adc_result*4);
OCR3C = (uint16_t) PRAXI_DIGITAL5;
}

ISR(TIMER4_COMPA_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL6;

ADCSRB &= 0x00;     // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result = ADC_metatroph(6); //ADC 6
PRAXI_DIGITAL6 = Katw_Akro_900_usec + (adc_result*4);
OCR4A = (uint16_t) PRAXI_DIGITAL6;

}



ISR(TIMER4_COMPB_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL7;

ADCSRB &= 0x00;        // Kanw to bit MUX5 = 0 gia na diavasw ta ADC0:7 channels
adc_result = ADC_metatroph(7); //ADC 7
PRAXI_DIGITAL7 = Katw_Akro_900_usec + (adc_result*4);
OCR4B = (uint16_t) PRAXI_DIGITAL7;
}

ISR(TIMER4_COMPC_vect)
{
uint16_t adc_result = 0;
float PRAXI_DIGITAL8;

ADCSRB |= (1<<MUX5);    // Kanw to bit MUX5 = 1 gia na diavasw ta ADC8:15 channels
adc_result = ADC_metatroph(0); //ADC 8
PRAXI_DIGITAL8 = Katw_Akro_900_usec + (adc_result*4);
OCR4C = (uint16_t) PRAXI_DIGITAL8;
}



ISR(TIMER5_CAPT_vect)  //==========Edw ginetai to demultiplexing tou timer 5 gia
antixeira , dexia-aristera daxtylwn , kai mikro daxtylo
{
int var_A ;
int var_B ;
int var_C ;
```

```
uint16_t adc_result;

ADCSRB = (1<<MUX5);

if(PORTA == 0x00)
{
PORTA = 0x03;

adc_result = ADC_metatroph_2 (1);
var_A = Katw_Akro_900_usec + (4*adc_result);  // adc9

OCR5A = (uint16_t) var_A;   //   megalh ar8rwsh tou mikrou daxtylou

adc_result = ADC_metatroph_2 (2);
var_B = Katw_Akro_900_usec + (4*adc_result);  // adc10

OCR5B = (uint16_t) var_B;  // Mesaia ar8rwsh mikrou daxtylou

OCR5C = (uint16_t) var_B;  // Mikrh ar8rwsh mikrou daxtylou
}
else if (PORTA >= 0x03)
{
PORTA = 0x01;

adc_result = ADC_metatroph_2 (3);
var_A = 4500-(ADC*4);      // adc11
OCR5A = (uint16_t) var_A;   // megalh ar8rwsh tou antixeira

adc_result = ADC_metatroph_2 (4);
var_B = 2500 + (4*adc_result);    // adc12
OCR5B = (uint16_t) var_B;   // mesaia ar8rwsh tou antixeira

adc_result = ADC_metatroph_2 (5);
var_C = Katw_Akro_900_usec + (4*adc_result);  // adc13
OCR5C = (uint16_t) var_C;   // mikrh ar8rwsh tou antixeira
}
else if(PORTA == 0x01)
{
PORTA = 0x02;

adc_result = ADC_metatroph_2 (6);
var_A = 1800 + (3*ADC);  // adc14    <<<<<==== OK
OCR5A = (uint16_t) var_A;     // to dexia-aristera srv tou deikth

adc_result = ADC_metatroph_2 (7);
var_B = 7400 - (8*ADC);  // adc15     <<<<========= OK
OCR5B = (uint16_t) var_B;   // dexia - aristera srv tou paramesou daxtylou

OCR5C = (uint16_t) var_B;   // dexia aristera  ar8rwsh tou mikrou daxtylou
}
else if(PORTA == 0x02)
{
PORTA = 0x03;

adc_result = ADC_metatroph_2 (1);
var_A = Katw_Akro_900_usec + (4*adc_result);  // adc9
OCR5A = (uint16_t) var_A;   //   megalh ar8rwsh tou mikrou daxtylou

adc_result = ADC_metatroph_2 (2);
var_B = Katw_Akro_900_usec + (4*adc_result);  // adc10
```

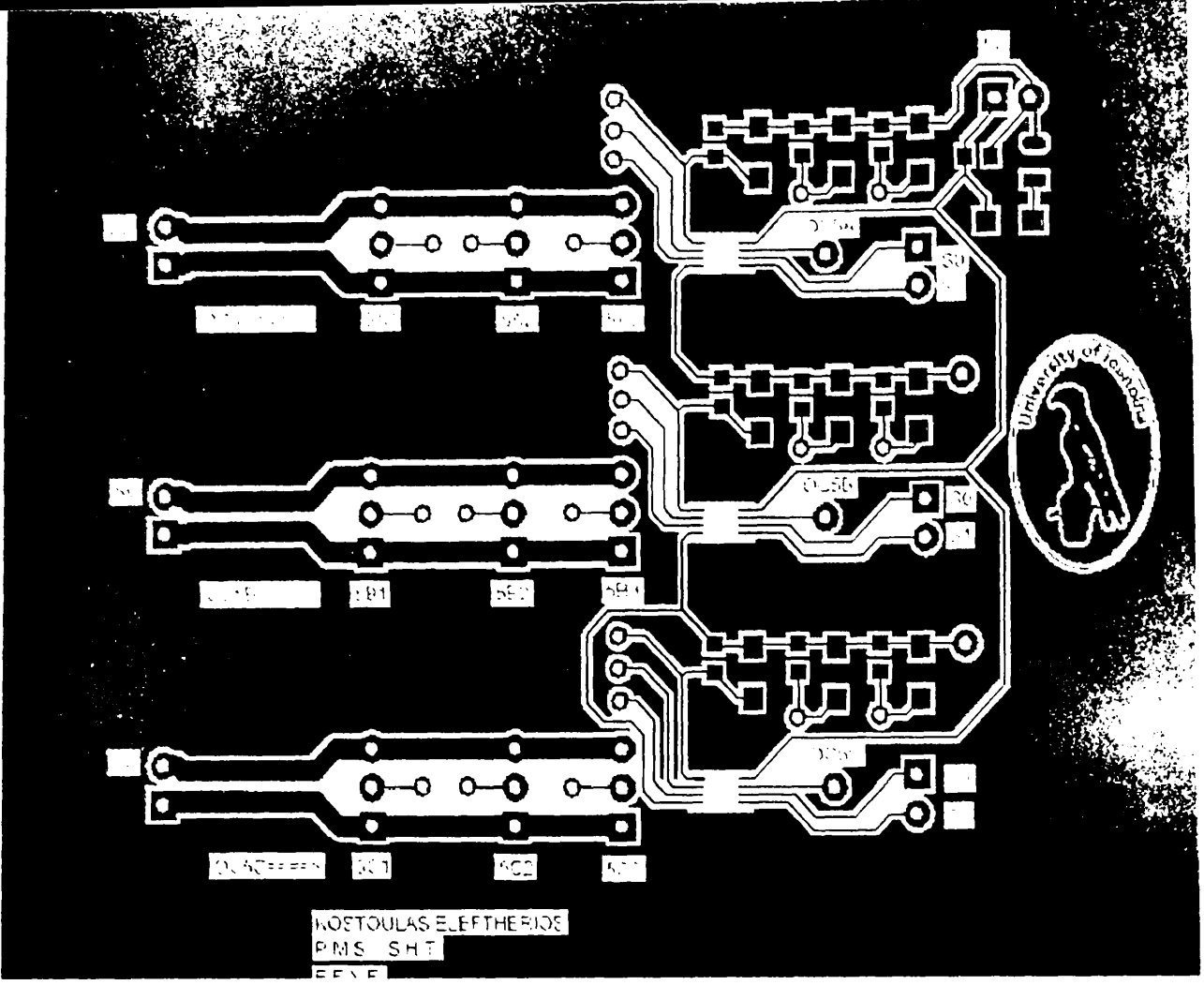
```
OCR5B = (uint16_t) var_B;   // Mesaia ar8rwsh mikrou daxtylou

OCR5C = (uint16_t) var_B;   // Mikrh ar8rwsh mikrou daxtylou

}
else
{
PORTC +=4 ;
}
}
```

Π6.3 Πλακέτα των παραγόμενων σημάτων από τους Demultiplexers

Σχηματικό των De-multiplexers

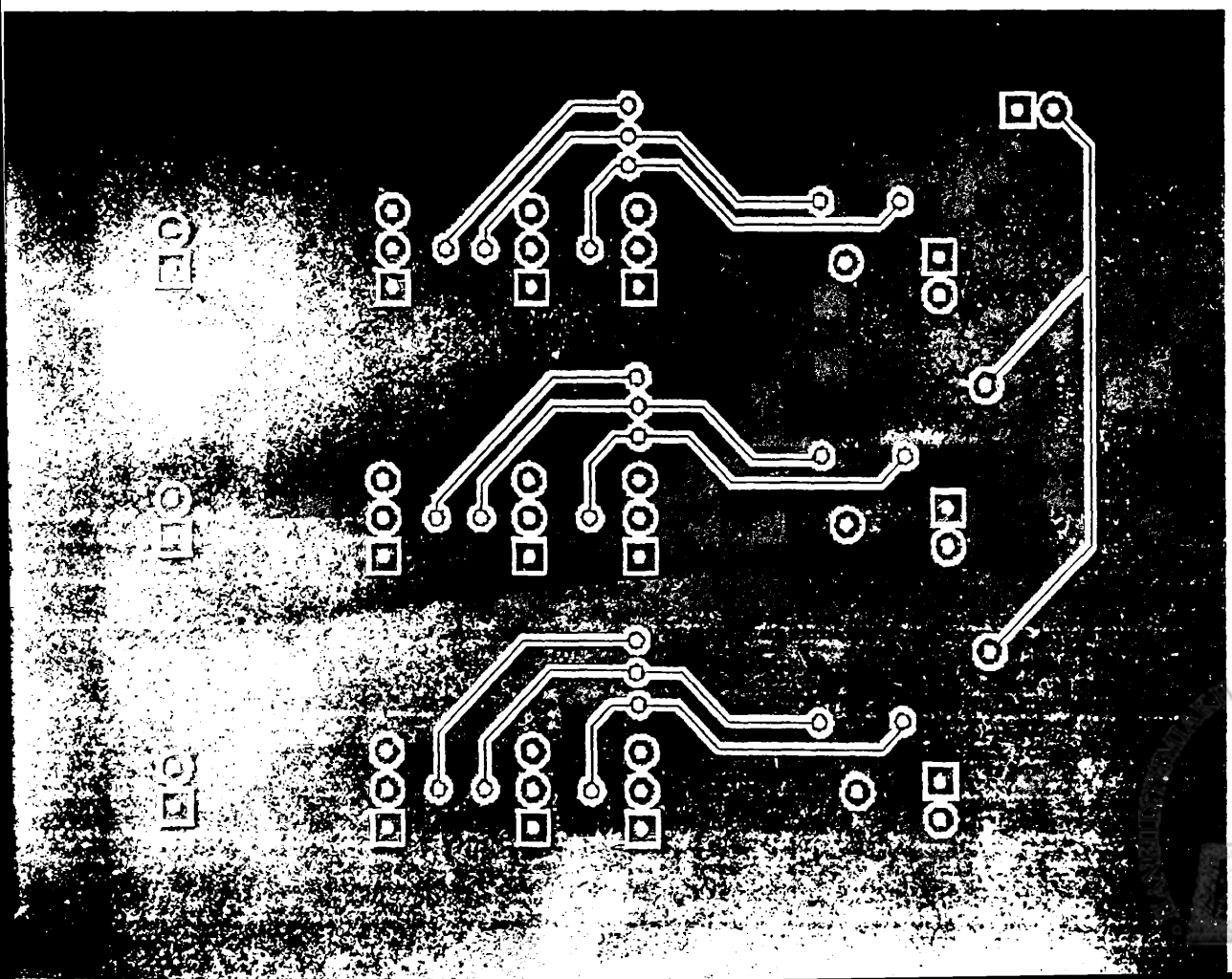


Εικόνα 136 : Σχηματικό πλακέτας DEMUX



Εικόνα 137: Πλακέτα DEMUX

Όψη



Π7 Λίστα Υλικών

Π7.1 Πλακέτα 1ου / 2ου Μικροελεγκτή

Footprint	Comment	LibRef	Designator	Description	Quantity
BUTTON_2PIN	button_2pin	button_2pin	but 1	Button_SMD pin-2	1
Capacitor	100nf	C0805	C1, C3, C4, C5, C6, C7, C10, C11, C12, C13, C14, C15, C16, C17, C18	C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, C0805, [NoValue]	15
Capacitor	10nf	C0805	C2	C0805	1
Capacitor	22pf	C0805	C8, C9	C0805	2
Crystal_16MHz	Crystal_16MHz	Crystal_16MHz	Crystal 1	Crystal_16MHz	1
Male_Header_3pin	OC3C	Male_Header_3pin	Head1	Header, 3-Pin	1
Male_Header_3pin	OC3B	Male_Header_3pin	Head2	Header, 3-Pin	1
Male_Header_3pin	OC3A	Male_Header_3pin	Head3	Header, 3-Pin	1
Male_Header_3pin	OC4C	Male_Header_3pin	Head4	Header, 3-Pin	1
Male_Header_3pin	OC4B	Male_Header_3pin	Head5	Header, 3-Pin	1
Male_Header_3pin	OC4A	Male_Header_3pin	Head6	Header, 3-Pin	1
Male_Header_3pin	OC1C	Male_Header_3pin	Head7	Header, 3-Pin	1
Male_Header_3pin	OC1B	Male_Header_3pin	Head8	Header, 3-Pin	1
Male_Header_3pin	OC1A	Male_Header_3pin	Head9	Header, 3-Pin	1
Male_Header_3pin	OC5C	Male_Header_3pin	Head10	Header, 3-Pin	1
Male_Header_3pin	OC5B	Male_Header_3pin	Head11	Header, 3-Pin	1
Male_Header_3pin	OC5A	Male_Header_3pin	Head12	Header, 3-Pin	1
Male_Header_2pin	Daxtylo2_6V	Male_Header_2pin	header1	Header, 2-Pin	1
Male_Header_2pin	Daxtylo3_6V	Male_Header_2pin	header2	Header, 2-Pin	1
Male_Header_2pin	Daxtylo1_6V	Male_Header_2pin	header3	Header, 2-Pin	1
Male_Header_2pin	Daxtylo4_6V	Male_Header_2pin	header4	Header, 2-Pin	1
HDR1X2	Vcc -5V Gnd	Header 2	HS1	Header, 2-Pin	1
inductor_10uH	Inductor_10uH	Inductor_10uH	inductor 1	L0805	1
HDR2X5	Header 5X2	Header 5X2	Jtag Connector	Header, 5-Pin, Dual row	1
led	led	led	led1		1
HDR1X8	Header 8	Header 8	Port A, Port C, Port J	Header, 8-Pin	3
HDR1X8	ADC7-0	Header 8	Port F	Header, 8-Pin	1
HDR1X8	ADC8-11	Header 8	Port K	Header, 8-Pin	1
Resistor	10k	R1206	R1	R1206	1
Resistor	300	R1206	R2	R1206	1
HDR2X5_CEN	ASP	Header 5X2A	SPI Conncetor	Header, 5-Pin, Dual row	1
100A_N	ATmega2560-16AU	ATmega2560-16AU	U1	8-bit AVR Microcontroller, 4.5-5.5V, 256KB Flash, 4KB EEPROM, 8KB SRAM, 86 GPIO pins, 100-pin TQFP, Industrial Grade (-40°C to 85°C), Pb-Free	1
					49

Π7.2 Πλακέτα Παραγωγής Αναλογικών Σημάτων

Footprint	Comment	LibRef	Designator	Description	Quantity
Capacitor	C0805	C0805	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21		21
Male_Header_2pin	5v	Male_Header_2pin	header1		1
PIN1	Socket	Socket	J1, J2, J3, J4, J5, J6, J7, J8, J9, J10, J11, J12, J13, J14, J15, J16, J17, J18, J19, J20	Socket	20
led	led	led	led1		1
HDR1X2	Header 2	Header 2	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20	Header, 2-Pin	20
Resistor	2.2KΩ	R1206	R1, R8, R14, R21	R1206	4
Resistor	2.3KΩ	R1206	R2, R19	R1206	2
Resistor	2KΩ	R1206	R3	R1206	1
Resistor	3.3KΩ	R1206	R4, R9	R1206	2
Resistor	1.7KΩ	R1206	R5, R7, R17	R1206	3
Resistor	300Ω	R1206	R6	R1206	1
Resistor	2.4KΩ	R1206	R10	R1206	1
Resistor	3.2KΩ	R1206	R11	R1206	1
Resistor	90KΩ	R1206	R12	R1206	1
Resistor	101KΩ	R1206	R13	R1206	1
Resistor	3.1KΩ	R1206	R15	R1206	1
Resistor	120KΩ	R1206	R16	R1206	1
Resistor	2KΩ	R1206	R18	R1206	1
Resistor	2.5KΩ	R1206	R20	R1206	1
SO8_N	LM358AD	LM358AD	U1, U2, U3, U4, U5, U6, U7, U8, U9, U10	Low-Power Dual Operational Amplifier	10
					94

Π7.3 Πλακέτες Τροφοδοσίας

Η κάθε μία από τις τρείς πλακέτες τροφοδοσίας έχει τα εξής χαρακτηριστικά

Footprint	Comment	LibRef	Designator	Description	Quantity
Capacitor_330uF	Capacitor_330uF	Capacitor_330uF	1, 2	SMD_cap	2
C1210	10uf	Cap Semi	C1, C2	SMD Capacitor	2
EUL7	SMD package	PTN78020w	PS1, PS2	Switching Regulator	2
14-1210	110Ω	Res3	R1, R2	R1206	2
14-1210	13KΩ	Res3	R3, R4	R1206	2
TERMINAL_CONNECTOR_2p	12V input	Terminal_Connector_2pin	TC1, TC2	Terminal_Connector_2pin	2
TERMINAL_CONNECTOR_2p	6 Volt Output	Terminal_Connector_2pin	TC3, TC4	Header, pin-2	2
					14

Π7.4 Αισθητήρες – Κινητήρες

Υλικά	*Κωδικός*	*Ποσότητα*	*Περιγραφή*	*Κατασκ/τής*
Motors	Turnigy TSS-10MG digital Micro	19	Mini Servos 2.2 kg*cm	Turnigy
	TGY-1440A	4	micro servo 0.8 kg*cm	Tower pro
Sensors	Bend Sensor® 1 inch	14	Polyester Over Laminate	Flexpoint sensor systems
	Flex Sensor 2.2 inches	4	(FS-L-0055-253-ST)	Spectra symbol

Π7.5 Μεταλλικά μέρη

Λωρίδες Αλουμινίου πάχους 0.8mm

Μήκη: 62.2mm, 69.2mm, 64mm, 68mm, 76mm, 83mm

Σύνολο κομματιών : 28

Μεταλλικό κουτί κατασκευών Διαστάσεων: 27x18x12

Π8 Βιβλιογραφία – Αναφορές

[1] http://www.polymtl.ca/labrobot/pdf/MIROC03.pdf

[2] http://biorobotics.harvard.edu/robotic_hand_optimization.html

[3] Μικροεπεξεργαστές ,θεωρία και εφαρμογές
Gilmore , Εκδόσεις Τζιόλα

[4] Προγραμματίζοντας τον μικροελεγκτή 8051
Myke Predko, Εκδόσεις Τζιόλα

[5] http://en.wikipedia.org/wiki/Instruction_pipeline

[6] Embedded C Programming and the ATMEL AVR
Barnett, Cox and O'Cull, Εκδόσεις Thomson

[7] http://www.atmel.com/tools/avrispmkii.aspx

[8] http://www.atmel.com/images/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[9] http://www.sensorwiki.org/doku.php/sensors/flexion

[10] http://pdf.datasheetcatalog.com/datasheet/stmicroelectronics/2163.pdf

[11] http://www.electrical4u.com/servo-motor-servo-mechanism-theory-and-working-principle/

[12] http://www.hobbyking.com/hobbyking/store/__30034__Turnigy_1440A_Servo_4_4g_0_8kg_10sec_V2_EU_warehouse_.html?strSearch=TGY-144

[13] http://www.hobbyking.com/hobbyking/store/__25454__Turnigy_TSS_10HM_Digital_Micro_Servo_2_2kg_10g_0_12.html?strSearch=TSS-10

[14] http://www.ti.com/product/ptn78020w

[15] http://www.ti.com/product/ts5a3359

[16] http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=68302&start=all&postdays=0&postorder=asc

[17] http://www.atmel.com/Images/Atmel-8456-8-and-32-bit-AVR-Microcontrollers-AVR127-Understanding-ADC-Parameters_Application-Note.pdf