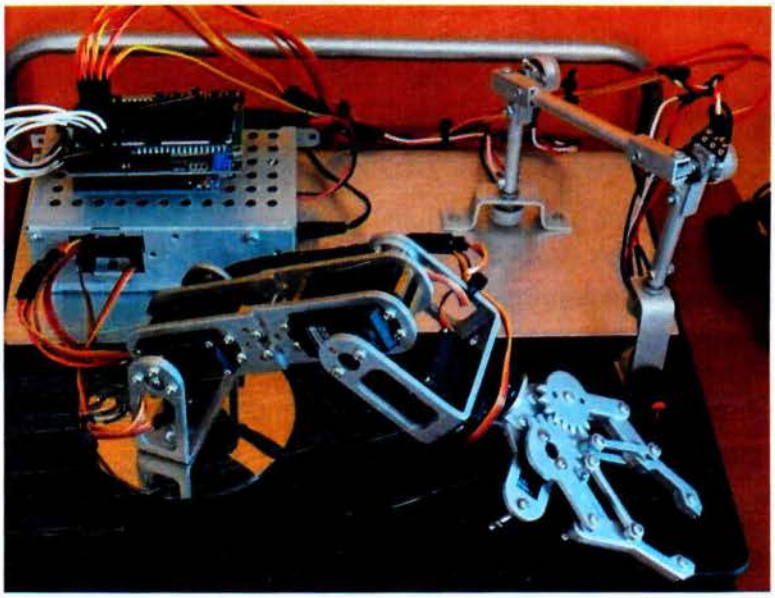
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ



ΧΕΙΡΙΣΜΟΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ ΜΕΣΩ ΣΥΣΤΗΜΑΤΟΣ ΑΦΕΝΤΗ-ΣΚΛΑΒΟΥ

ΠΑΝΑΓΟΠΟΥΛΟΣ ΝΙΚΟΛΑΟΣ

Α.Μ. :38685

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : ΠΑΠΟΥΤΣΙΔΑΚΗΣ ΜΙΧΑΛΗΣ

ΑΘΗΝΑ -2012

ΠΕΡΙΛΗΨΗ

Στη παρούσα εργασία προβάλλεται η μελέτη και ανάπτυξη μιας εκπαιδευτικής ρομποτικής πλατφόρμας , ικανής να χρησιμοποιηθεί για την κατανόηση βασικών θεμάτων που σχετίζονται με την ρομποτική, τον έλεγχο και την μηχατρονική .

Στα πλαίσια της εργασίας πραγματοποιήθηκε ο σχεδιασμός της τόσο σε επίπεδο υλικού(Hardware), όσο και σε επίπεδο λογισμικού(Software) .Στη συνέχεια πραγματοποιήθηκε η ολοκλήρωση της κατασκευής και δοκιμάστηκε η λειτουργικότητα της .Ο σχεδιασμός και η υλοποίηση πραγματοποιήθηκαν με στόχο την βέλτιστη σχέση κόστους-απόδοσης .

Η κατασκευή αυτή περιλαμβάνει ένα ρομποτικό βραχίονα 5 βαθμών ελευθερίας . Ο έλεγχος της κίνησης του βραχίονα πραγματοποιείται μέσω χειριστηρίου ιδίων βαθμών ελευθερία . Ο χειριστής έχει επίσης την δυνατότητα να προγραμματίσει τον ρομποτικό βραχίονα αποθηκεύοντας θέσεις στις οποίες τον έχει προσαρμόσει μέσω του χειριστηρίου. Αυτές τις θέσεις θα μπορεί να τις αναπαράγει προκειμένου να χρησιμοποιηθούν σε μια ακολουθία κινήσεων για αυτόματη και επαναλαμβανόμενη κίνηση, σημείο προς σημείο. Η επικοινωνία και ο προγραμματισμός επιτυγχάνονται μέσω ενός μικροελεγκτή. Σκοπός αυτής της εφαρμογής είναι η ανάπτυξη μιας φιλικής προς τον χρήστη διεπαφής , για τον χειρισμό του βραχίονα σε ελεγχόμενο περιβάλλον .

"CONTROL AND PROGRAMMING OF ROBOTIC ARM

VIA MASTER-SLAVE SYSTEM"

ABSTRACT

This paper presents the design and development of an educational robotic platform, usable for understanding key issues related to robotics, control and mechatronics.

This paper consists of the design of both hardware and software. Afterwards, completion of the construction and functionality test are performed. Designing and implementation are made aiming at an optimal cost –efficiency ratio.

This structure includes a robotic arm with 5 degrees of freedom. The arms motion control is performed via joystick with same degrees of freedom. The operator also has the ability to program the robotic arm by storing positions in which he has adapted the arm by use of the joystick . These positions can be recovered to be used in a sequence of movements for automatic and repetitive movement, point by point. Communication and programming are achieved through a microcontroller. The purpose of this application is to develop a user friendly interface to operate the arm in a controlled environment.

ΕΙΣΑΓΩΓΗ

Ο χειροκίνητος έλεγχος ενός ρομποτικού βραχίονα σε πραγματικό χρόνο είναι εξίσου σημαντικός και αναγκαίος με την αυτόνομη λειτουργία του. Ενδείκνυται σε εφαρμογές που ο χειριστής δεν μπορεί ή δεν πρέπει να πλησιάσει το περιβάλλον εργασίας, αλλά απαιτείτε η συνεχής επίβλεψη και παρέμβαση του. Υπάρχουν περιπτώσεις στις οποίες οι αυτοματοποιημένες λύσεις δεν είναι αξιόπιστης ή δεν μπορούν να προληφθούν πλήρως οι αντιξοότητες από τους αλγορίθμους ελέγχου του προγραμματιστή. Μερικά ενδεικτικά σενάρια μπορεί να είναι :

- Εργασίες σε τοξικά περιβάλλοντα
- Διαστημικές εφαρμογές (με δυνατότητα χειρισμού σε πραγματικό χρόνο)
- Εφαρμογές τηλεπαρουσίας (π.χ. Ανδροειδή διασώσεων)
- Ρομποτική χειρουργική

Σε αυτές τις περιπτώσεις είναι ιδιαίτερα αναγκαίο ο ρομποτικός μηχανισμός να ανταποκρίνεται με όσο το δυνατό μικρότερη απόκλιση θέσης και χρονική καθυστέρηση στις εντολές του χειριστή, καθώς σε αντίθετη περίπτωση μπορούν να προκύψουν ανεπιθύμητα ή και επικίνδυνα αποτελέσματα. Ιδανικά ένα σύστημα χειρισμού θα έχει την πρόσθετη ικανότητα να εντοπίζει τέτοιες ενέργειες με κατάλληλα αισθητήρια και να τις αποφεύγει.

Για να αποφθεχθεί η χρήση ενός σύνθετου και ακριβού ελεγκτή η παρούσα εργασία ασχολείται με την κατασκευή ενός μικρού εκπαιδευτικού βραχίονα . Έτσι ο ελεγκτής που απαιτείτε δεν χρειάζεται να εκτελεί σύνθετους ελέγχους και ο χειριστής μπορεί να ελέγξει τον βραχίονα εμπειρικά. Γι αυτό χρησιμοποιείται ένας μικροελεγκτής χαμηλού κόστους που μπορεί να εκτελεί εύκολα και γρήγορα τους κατάλληλους υπολογισμούς, έτσι ώστε να οδηγήσει ιδανικά το βραχίονα. Ο ελεγκτής μπορεί να εκτελέσει αρκετά σύνθετους υπολογισμούς καθώς προγραμματίζεται σε γλώσσα υψηλού επιπέδου.

Η εφαρμογή στην οποία θα χρησιμοποιηθεί ο ρομποτικός βραχίονας , επηρεάζει με την σειρά του και τα υλικά κατασκευής του. Στην εργασία αυτή σκοπός είναι η απλή μετακίνηση αντικειμένων στον χώρο εργασίας του ρομποτικού βραχίονα. Έτσι αφού ο ρομποτικός βραχίονας αποτελεί εκπαιδευτική εφαρμογή , για την κατασκευή του χρησιμοποιήθηκε ένα ελαφρύ και οικονομικό μέταλλο, ιδανικό για τα συμπαγή μέρη του βραχίονα. Τέλος, το μέγεθος του βραχίονα επιλέχθηκε κατάλληλα, ούτος ώστε να μπορεί να οδηγηθεί από κινητήρες που μπορούν να βρεθούν εύκολα και με χαμηλό κόστος στο εμπόριο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

1.1 Ιστορική αναδρομή

Αναζητώντας κανείς τις ρίζες της ρομποτικής θα οδηγηθεί αρκετά πίσω στην ιστορία της ανθρωπότητας. Πράγματι , η φιλοδοξία του ανθρώπου να δημιουργήσει μηχανές που θα του μοιάζουν τόσο στην μορφή όσο και στην λειτουργία πρωτοσυναντάται στην Ελληνική μυθολογία. Σύμφωνα με την τελευταία η κατασκευή του Ήφαιστου ¨Τάλος¨, μυθικός χάλκινος γίγαντας που προστάτευε την Κρήτη από τους εισβολείς, αποτελεί το πρώτο «αυτόματο» στην ανθρώπινη ιστορία.

Στην σύγχρονη εποχή, η εισαγωγή της έννοιας των ρομπότ έγινε το 1921 από τον Τσέχο θεατρικό συγγραφέα Karel Capek με το θεατρικό έργο "Rossum's Universal Robots". Στο τελευταίο ο συγγραφέας φαντάζεται ένα μηχανικό κατασκεύασμα, το οποίο και ονομάζει Robot από την Τσεχική λέξη robota για την καταναγκαστική εργασία. Το «αυτόματο» του Rossum στρέφεται τελικά εναντίον της ανθρωπότητας.

Λίγα χρόνια αργότερα, κατά την δεκαετία του 40', ο Ρώσος συγγραφέας επιστημονικής φαντασίας Issac Asimov συνέλαβε το robot ως ένα «αυτόματο» με εμφάνιση ανθρώπου αλλά απαλλαγμένο από συναισθήματα. Η συμπεριφορά του υπαγορευόταν από ένα «ποζιτρονικό μυαλό» προγραμματισμένο από τον άνθρωπο κατά τέτοιο τρόπο ώστε να ανταποκρίνεται σε συγκεκριμένες αρχές ηθικής συμπεριφοράς. Ο Όρος ρομποτική χρησιμοποιήθηκε από τον Asimov ως το σύμβολο της επιστήμης που είναι αφιερωμένη στην μελέτη των robot.

1.2 Οριοθέτηση της ρομποτικής

1.2.1 Ορισμός της ρομποτικής

Η ρομποτική είναι εκείνος ο κλάδος της επιστήμης του μηχανικού που ασχολείται με τη σύλληψη, το σχεδιασμό, την κατασκευή και τη λειτουργία robot. Τα robot είναι μηχανές ,η χρήση των οποίων αποσκοπεί στην αντικατάσταση του ανθρώπου στην εκτέλεση του έργου. Η αντικατάσταση αυτή αφορά τόσο στο φυσικό επίπεδο του έργου όσο και στο επίπεδο λήψης απόφασης.

1.2.2 Ορισμός του ρομπότ

Σύμφωνα με το Ινστιτούτο Ρομποτικής της Αμερικής, ρομπότ είναι ένας αυτοπρογραμματιζόμενος και προλειτουργικούς χωρικός μηχανισμός σχεδιασμένος να μετακινεί υλικά, αντικείμενα, εργαλεία ή ειδικευμένες συσκευές με κατάλληλες μεταβλητά προγραμματιζόμενες κινήσεις που στοχεύουν στην βελτίωση της απόδοσης μιας σειράς εργασιών.

Ένας τέτοιος μηχανισμός περιλαμβάνει συνήθως τις ακόλουθες συνιστώσες :

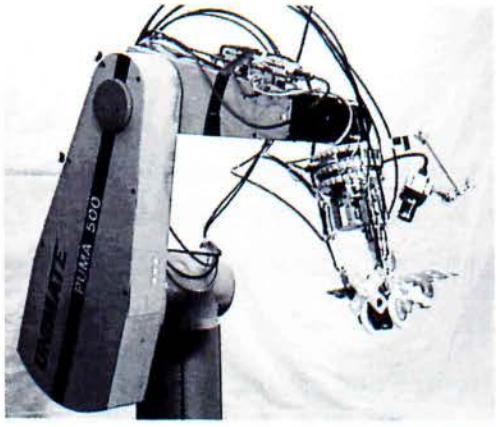
- Ένα μηχανολογικό υποσύστημα, το οποίο ενσωματώνει την δυνατότητα του ρομπότ για εκτέλεση έργου. Το υποσύστημα αυτό αποτελείται από μηχανισμούς που επιτρέπουν στο ρομπότ να κινείται όπως αρθρώσεις, συστήματα μετάδοσης κίνησης, επενέργησες-κινητήρες, οδηγούς κλπ.
- Ένα υποσύστημα αίσθησης, μέσω του οποίου το ρομπότ συγκεντρώνει πληροφορίες για την κατάσταση στην οποία βρίσκονται τόσο το ίδιο όσο και το περιβάλλον. Το υποσύστημα αυτό εκτός των άλλων είναι υπεύθυνο για την υποδοχή των εξωτερικών εντολών, την επεξεργασία τους, την μετάφραση τους σε ηλεκτρική ισχύ που θα δοθεί στους κινητήρες του ρομπότ, καθώς επίσης και για την παραγωγή σημάτων εξόδου που θα πληροφορούν για την κατάσταση του συστήματος. Στο υποσύστημα αίσθησης περιλαμβάνονται όργανα μετρήσεων, αισθητήρες, ηλεκτρονικά στοιχειά κλπ.
- Ένα σύστημα ελέγχου, το οποίο συνδυάζει κατάλληλα την αίσθηση με την δράση, έτσι ώστε το ρομπότ να λειτουργεί αποτελεσματικά και με τον επιθυμητό τρόπο. Ο ελεγκτής του ρομπότ επιβλέπει και συντονίζει ολόκληρο το σύστημα, για την σχεδίαση και υλοποίηση του, που απαιτείται ο συνδυασμός γνώσεων από πολλές γνωστικές περιοχές, όπως είναι ο αυτόματος έλεγχος, η τεχνητή νοημοσύνη, η επιστήμη των υπολογιστών κλπ.

1.3 Είδη ρομπότ

Κατά την πολυετή εξέλιξη της επιστήμης της ρομποτικής πρόεκυψαν διάφορα είδη ρομποτικών μηχανισμών, οι οποίοι διαφέρουν σημαντικά στην μορφή, αποτελούνται όμως από αντίστοιχα επιμέρους υποσυστήματα. Τα τελευταία είναι αυτά που αναφέρθηκαν παραπάνω.

Τα σπουδαιότερα είδη ρομπότ είναι τα παρακάτω :

- **Ρομπότ σταθερής βάσης :** τα ρομπότ αυτά αποτελούνται από διαδοχικά στερεά σώματα (σύνδεσμοι) που συνδέονται μέσω αρθρώσεων σχηματίζοντας μια κινηματική αλυσίδα. Η αλυσίδα αυτή έχει το ένα άκρο της (βάση) σταθερά συνδεδεμένο με κάποιο σημείο του περιβάλλοντος χώρου. Η μορφή αυτή ρομπότ είναι η παραδοσιακή μορφή ενός βιομηχανικού ρομποτικού βραχίονα, και περιλαμβάνει τον βραχίονα, το καρπό και το εργαλείο (Σχήμα 1).



Σχήμα 1 Ο βιομηχανικός ρομποτικός βραχίονας PUMA 500 της Unimation inc.

- **<u>Κινούμενα ρομπότ :</u>** ως κινητά ρομπότ χαρακτηρίζονται όλα εκείνα τα ρομπότ που έχουν την δυνατότητα να μετακινήσουν όλα τα σημεία του μηχανισμού τους. Η δυνατότητα αυτή προσφέρεται από ειδικά συστήματα προώθησης, τα οποία μπορεί να είναι είτε απλά (όπως τροχοί) είτε πολύπλοκα (όπως jet, προπέλες, μηχανικά πόδια). Τα κινούμενα ρομπότ διακρίνονται σε επιμέρους κατηγορίες ανάλογα με τον βαθμό αυτονομίας τους. Έτσι έχουμε :

    1. AGVs (Automatic guided vehicles)Περιορισμένης αυτονομίας κίνησης
    2. Αυτόνομα έντρομα ρομπότ
    3. Βαδίζοντα ρομπότ
    4. ROVs (Remotely operated vehicles) Περιορισμένης αυτονομίας κίνησης
    5. AUVs (Autonomous underwater vehicles)
    6. Εναέρια ρομπότ

Σχήμα 2 AGV σε βιομηχανικό περιβάλλον

1.4 Δομικά χαρακτηριστικά βιομηχανικών ρομποτικών βραχιόνων

1.4.1 Είδη αρθρώσεων

Όπως έχουμε ήδη σημειώσει ένας ρομποτικός βραχίονας αποτελείται από μια σειρά διαδοχικών στερεών σωμάτων που ονομάζονται σύνδεσμοι. Οι σύνδεσμοι συνδέονται ανά δυο μεταξύ τους μέσω αρθρώσεων σχηματίζοντας μια κινηματική αλυσίδα. Οι αρθρώσεις μπορεί να είναι :

Πρισματικές :Σχετική μεταφορική κίνηση μεταξύ δυο διαδοχικών συνδέσμων

Περιστροφικές :Υλοποιούν σχετική περιστροφική κίνηση μεταξύ δυο διαδοχικών συνδέσμων

Σφαιρικές :Υλοποιούν σφαιρική περιστροφική κίνηση μεταξύ δυο διαδοχικών συνδέσμων

1.4.2 Είδη κινητήρων

Για την κίνηση των αρθρώσεων των βραχιόνων είναι απαραίτητη η χρήση κινητήρων. Κάθε κατασκευαστής χρησιμοποιεί διαφορετικούς κινητήρες. Οι κινητήρες που χρησιμοποιούνται ευρέως είναι οι κινητήρες συνεχούς ρεύματος (dc motors), οι βηματικοί κινητήρες (stepper motors) και οι σερβοκινητήρες (servo motors). Κάθε είδος κινητήρων έχει πλεονεκτήματα και μειονεκτήματα, τα οποία καθορίζουν αν το είδος κινητήρων που θα χρησιμοποιηθεί για την κίνηση των αρθρώσεων ενός βραχίονα είναι κατάλληλο ή όχι. Όλοι οι κινητήρες που αναφέρθηκαν λειτουργούν με συνεχή τάση που διαφέρει από τον τύπο του κινητήρα. Στον Πίνακα 1.4.1 γίνεται σύγκριση των βασικότερων χαρακτηριστικών των τριών τύπων κινητήρων που αναφέρονται.

Πινάκας 1.4.1 Σύγκριση κινητήρων παρόμοιων διαστάσεων

Κινητήρες	Γωνία περιστροφής	Ροπή	Ενσωματωμένο κιβώτιο μετάδοσης	Έλεγχος θέσης του άξονα	Εξωτερικό κύκλωμα οδήγησης
Συνεχούς ρεύματος	>360°	Μεγάλη	Όχι	Όχι	Ναι
Βηματικός	>360°	Μικρή	Όχι	Απαιτείται αρχικοποίηση	Ναι
Σέρβο	≈180°	Μεγάλη	Ναι	Ναι	Όχι

1.4.3 Βαθμοί κινητικότητας και βαθμοί ελευθερίας

Είναι σημαντική η επισήμανση της διαφοράς που υπάρχει ανάμεσα στους βαθμούς κινητικότητας και βαθμούς ελευθερίας. Για ένα βραχίονα το πλήθος των βαθμών κινητικότητας είναι σταθερό και ίσο με το πλήθος των αρθρώσεων του. Από την άλλη πλευρά οι βαθμοί ελευθερίας είναι άμεσα συνδεδεμένοι με το συγκεκριμένο έργο που καλείται να φέρει εις πέρας ο βραχίονας. Για την γενική περίπτωση που θέλουμε να τοποθετήσουμε και να προσανατολίσουμε ένα αντικείμενο στον τρισδιάστατο χώρο απαιτούνται 6 βαθμοί ελευθερίας (3 για να τοποθετήσουν το αντικείμενο στο χώρο και 3 για να το προσανατολίσουν ως προς ένα σύστημα συντεταγμένων αναφοράς).

1.4.4 Χώρος εργασίας

Ως χώρος εργασίας ορίζεται ο τρισδιάστατος χώρος τον οποίο μπορεί να σαρώσει η άκρη του ρομποτικού μηχανισμού. Το μέγεθος και η γεωμετρική μορφή του χώρου αυτού εξαρτώνται από την κατασκευαστική δομή του ρομπότ, κάτι που θα γίνει φανερό και στην συνέχεια.

1.4.5 Ωφέλιμο φορτίο-Επαναληψιμότητα-Ακρίβεια

Από τα ποιο σημαντικά μεγέθη ενός βιομηχανικού βραχίονα είναι το ωφέλιμο φορτίο, η επαναληψιμότητα και η ακρίβεια. Ποιο συγκεκριμένα τα παραπάνω μεγέθη αναφέρονται στα εξής :

- **Ωφέλιμο φορτίο :** είναι το βάρος που μπορεί να μεταφέρει το άκρο του βραχίονα. Ως σημείο εφαρμογής του βάρους θεωρείται η φλάντζα του καρπού. Το προδιαγραφόμενο αυτό φορτίο δεν είναι σταθερό και εξαρτάται από την ταχύτητα με την οποία πρόκειται να κινηθεί ο καρπός.

- **Επαναλιψημότητα :** εκφράζει την δυνατότητα του βραχίονα να επιστρέψει στο ίδιο σημείο μετά από αρκετές επαναλήψεις και δίνεται ως εύρος στο οποίο ο βραχίονας θα τερματίσει την κίνηση. Η απόκλιση οφείλεται στο ότι το ρομπότ κατά την λειτουργία του είναι δυνατό να χάσει λίγο από την μέτρηση της θέσης με αποτέλεσμα να μην μπορεί να επιστρέψει στην συγκεκριμένη θέση μετά από ορισμένους κύκλους λειτουργίας. Δεδομένου ότι στις συνήθεις βιομηχανικές εφαρμογές οι επιθυμητές κινήσεις διδάσκονται στο ρομπότ, αντιλαμβάνεται κανείς την σπουδαιότητα της επαναληψιμότητας.

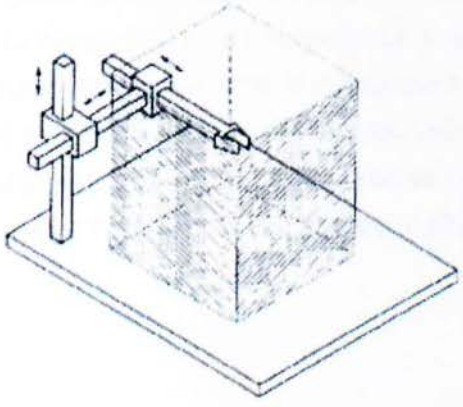
- **Ακρίβεια :** είναι η ικανότητα του ρομπότ να πηγαίνει ακριβώς στην θέση που του έχει δοθεί εντολή να πάει. Η ακρίβεια εξαρτάται κυρίως από την διακριτικότητα των εξαρτημάτων ελέγχου, την μηχανολογική σύνδεση των μελών του και το ελάχιστο επιτρεπόμενο σφάλμα που επιβάλει η ευστάθεια της λειτουργίας των σέρβο. Η ακρίβεια επηρεάζεται από το είδος και το μέγεθος του εκάστοτε φορτίου, σε αντίθεση με την επαναληψιμότητα, γι΄αυτό και ορισμένοι κατασκευαστές προδιαγράφουν μόνο το τελευταίο.

1.4.6 Ταξινόμηση βραχιόνων βάση της γεωμετρικής διαμόρφωσης τους

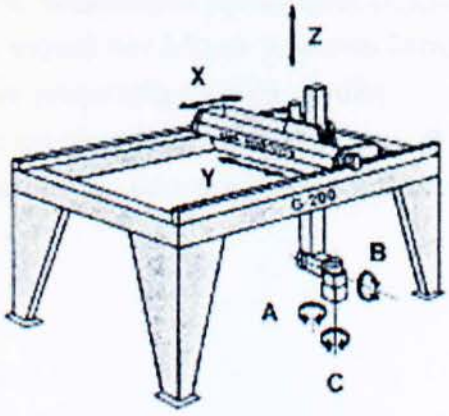
Ο τύπος και η διαδοχή των αρθρώσεων ενός βραχίονα επιτρέπει την ταξινόμηση των ρομπότ σε διαφορές κατηγορίες, οι οποίες αναφέρονται παρακάτω. Οι αρθρώσεις που μας απασχολούν στο σημείο αυτό είναι οι τρεις πρώτες του βραχίονα και κατά συνέπεια εξαιρούνται οι αρθρώσεις του καρπού. Θα έχουμε λοιπόν τα εξής :

- **Καρτεσιανοί βραχίονες :** η καρτεσιανή γεωμετρία υλοποιείται με τρεις διαδοχικές πρισματικές αρθρώσεις. Οι άξονες των αρθρώσεων αυτών είναι ανά δυο κάθετοι μεταξύ τους (Σχήμα 3). Η καρτεσιανή δομή παρέχει μεγάλη δυσκαμψία και σταθερή ακρίβεια σε ολόκληρο το χώρο εργασίας που είναι ένα παραλληλεπίπεδο. Βασικό μειονέκτημα είναι η μειωμένη επιδεξιότητα κίνησης, λόγο της πρισματικής φύσης των αρθρώσεων.

- **Βραχίονες Gantry :** οι βραχίονες Gantry είναι στην ουσία καρτεσιανοί, διαφέρουν όμως από τους τελευταίους στον τρόπο προσέγγισης του αντικείμενου ενδιαφέροντος (Σχήμα 4). Ειδικότερα ο βραχίονας Gantry προσεγγίζει το αντικείμενο από πάνω σε αντίθεση με ένα κλασσικό καρτεσιανό βραχίονα που προσεγγίζει το αντικείμενο από το πλάι. Άμεσες συνέπειες της διαφοροποίησης αυτής είναι η αύξηση του χώρου εργασίας, και της δυσκαμψίας καθώς επίσης και η δυνατότητα χειρισμού μεγάλων και βαριών αντικειμένων.



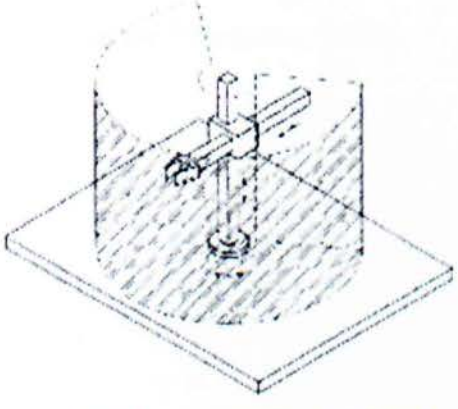
Σχήμα 3 καρτεσιανός βραχίονας



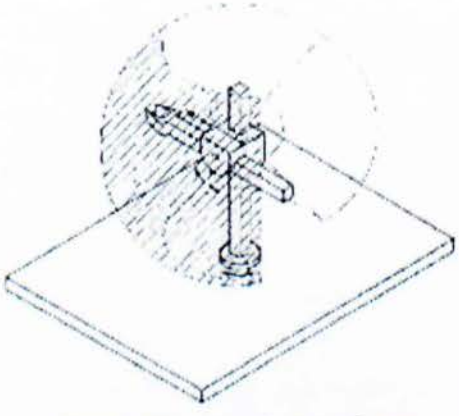
Σχήμα 4 Βραχίονας Gantry

- **Κυλινδρικοί βραχίονες :** στους κυλινδρικούς βραχίονες η πρώτη πρισματική άρθρωση της καρτεσιανής δομής έχει αντικατασταθεί από μια περιστροφική άρθρωση (Σχήμα 5). Ο χώρος εργασία στην περίπτωση αυτή είναι τμήμα κυλίνδρου. Σημαντικό μειονέκτημα της συγκεκριμένης γεωμετρίας είναι ότι ο βραχίονας εισέρχεται στον χώρο εργασίας και τον περιορίζει.

- **Σφαιρικοί βραχίονες :** στους βραχίονες αυτούς αντικαθίσταται πλέον και η δεύτερη πρισματική άρθρωση με περιστροφική (Σχήμα 6). Η μηχανολογική πολυπλοκότητα αυξάνεται, ενώ η δυσκαμψία μειώνεται. Επιπλέων η ακρίβεια του καρπού μειώνεται με την αύξηση της ακτινικής απόστασης. Ο χώρος εργασίας είναι τμήμα σφαίρας και περιέχει ένα μέρος της βάσης με άμεση συνέπεια την δυνατότητα χειρισμού των αντικειμένων που βρίσκονται στο έδαφος.



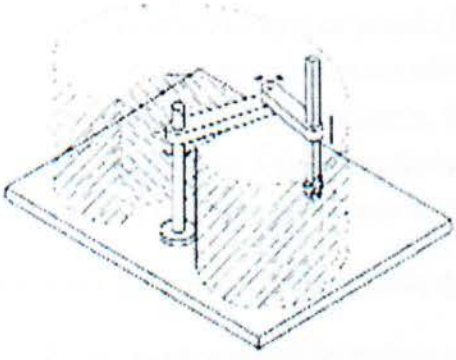
Σχήμα 5 Κυλινδρικός βραχίονας



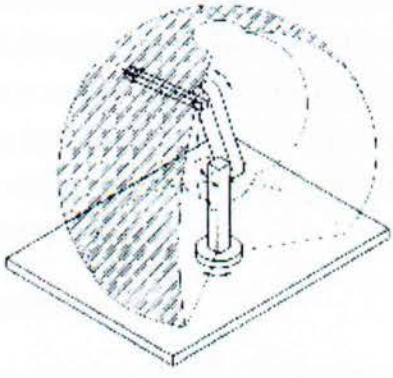
Σχήμα 6 Σφαιρικός βραχίονας

- **Βραχίονας SCARA :** η γεωμετρία SCARA είναι ειδική και περιλαμβάνει δύο περιστροφικές και μια πρισματική άρθρωση τοποθετημένες κατά τέτοιο τρόπο ώστε οι άξονες κίνησης να είναι παράλληλοι μεταξύ τους (Σχήμα 7). Το όνομα SCARA προέρχεται από τα αρχικά των λέξεων Selective Compliance Assembly Robot Arm. Η συγκεκριμένη γεωμετρία παρέχει μεγάλη δυσκαμψία σε κατακόρυφο επίπεδο και ελαστικότητα σε οριζόντιο. Η ακρίβεια τοποθέτησης του καρπού μειώνεται με την αύξηση της απόστασης του από τον άξονα της πρώτης άρθρωσης.

- **Ανθρωπομορφικοί βραχίονες :** η ανθρωπομορφική γεωμετρία επιτυγχάνεται με τρεις διαδοχικές περιστροφικές αρθρώσεις (Σχήμα 8). Ειδικότερα, ο άξονα περιστροφής της πρώτης άρθρωσης είναι κατακόρυφος και κάθετος στους άξονες περιστροφής των δύο επόμενων αρθρώσεων, οι οποίοι είναι παράλληλοι μεταξύ τους. Η συγκεκριμένη δομή παρέχει την μεγαλύτερη επιδεξιότητα από όλες τις προηγούμενες, καθώς όλες οι αρθρώσεις είναι περιστροφικές. Ωστόσο η ακρίβεια του καρπού δεν είναι σταθερή μέσα στον χώρο εργασίας που έχει την μορφή σφαίρας.



Σχήμα 7 Βραχίονας SCARA



Σχήμα 8 Ανθρωπομορφικός βραχίονας

ΚΕΦΑΛΑΙΟ 2

2.1 Αντικείμενο της εργασίας

Ο βασικός στόχος και αντικείμενο της εργασίας είναι η σχεδίαση και κατασκευή ενός ρομποτικού βραχίονα 5 βαθμών ελευθερίας, της διάταξης χειρισμού του, με εύκολο τρόπο, καθώς και μιας γραφικής διεπαφής που θα κάνει εύκολο τον προγραμματισμό του βραχίονα από τον χειριστή χωρίς την απαίτηση εξειδικευμένων γνώσεων προγραμματισμού και ρομποτικής από τον ίδιο. Η εφαρμογή μια κατασκευής σαν και αυτή σε μεγαλύτερη κλίμακα θα μπορούσε να παρέχει πιθανών την δυνατότητα σε μικρές βιοτεχνίες να αυξήσουν την παραγωγή τους εισάγοντας ευέλικτα αυτοματοποιημένα συστήματα για την εκτέλεση απλών διαδικασιών στην παραγωγική διαδικασία. Τα προγράμματα εκτέλεση αυτών των διαδικασιών θα μπορούν να δημιουργηθούν ακόμα και από υπαλλήλους (χειριστές) που δεν διαθέτουν υπόβαθρο γνώσεων προγραμματισμού.

2.2 Αναπτυξιακοί στόχοι και Σχεδιαστικές προδιαγραφές

1. Οι κινήσεις του βραχίονα θα πρέπει να μοιάζουν όσο το δυνατό περισσότερο με εκείνες του χειριστηρίου, αφού το χειριστήριο έχει τους ίδιους βαθμού ελευθερίας και ανάλογο μέγεθος.
2. Το κόστος του ελεγκτή να είναι χαμηλό.
3. Η ταχύτητα του ελεγκτή πρέπει να είναι υψηλή ώστε να ανταποκρίνεται άμεσα σε κάθε κίνηση του χειριστηρίου.
4. Το σφάλμα του συστήματος να είναι μικρό.
5. Η διεπαφή χρήστη-μηχανής θα πρέπει να είναι απλή.
6. Πρέπει να παρέχεται στον χειριστή η επιλογή επαναλαμβανόμενης εκτέλεσης των προγραμμάτων.

Η προσέγγιση των παραπάνω στόχων δίνεται στα επόμενα κεφάλαια.

2.3 Αναλυτική περιγραφή της κατασκευής

Η κατασκευή περιλαμβάνει το ρομποτικό βραχίονα 5 βαθμών ελευθερίας (5 DOF) οι οποίοι είναι , η άρθρωση της βάσης , δυο αρθρώσεις κλίσης ,η άρθρωση που περιστρέφει την αρπάγη και αυτή που την ανοίγει και την κλείνει . Στις αρθρώσεις αυτές χρησιμοποιούνται κινητήρες Servo για κίνηση 180 μοιρών σε κάθε επίπεδο . Οι κινητήρες λαμβάνουν την κατάλληλη τάση λειτουργίας από ένα κύκλωμα τροφοδοσία το οποίο χρησιμοποιείτε και για την λειτουργία του μικροελεγκτή. Τα σήματα ελέγχου που καθορίζουν την θέση του βραχίονα παρέχονται από τον μικροελεγκτή. Ο ελεγκτής είναι μία πλακέτα Arduino Uno R3, με ενσωματωμένο ένα μικροελεγκτή Atmel AVR (ATmega328) .

Πάνω στον ελεγκτή είναι τοποθετημένη μια LCD (Liquid crystal display) οθόνη 02X16 και πληκτρολόγιο 5 πλήκτρων, πάνω , κάτω , δεξιά , αριστερά και επιλογή , για την πλοήγηση στο μενού της οθόνης . Ακόμα υπάρχει το χειριστήριο (Joystick) το οποίο αποτελείται από πέντε ποτενσιόμετρα 1 kΩ , τα οποία χρησιμοποιούνται ως αρθρώσεις . Τα ποτενσιόμετρα τροφοδοτούνται από τον μικροελεγκτή με 5 Volt και επιστρέφουν σε αυτόν μια τιμή από 0 έως 5 volt .Ο ελεγκτής διαβάζει αυτές τις τιμές και τις χρησιμοποιεί κατάλληλα για τον χειρισμό του ρομποτικού βραχίονα. Ο χρήστης θα έχει την δυνατότητα να χειρίζεται τον βραχίονα σε πραγματικό χρόνο, όπως και να αποθηκεύει ενιαία προγράμματα στην μνήμη του μικροελεγκτή , ούτος ώστε να μπορεί να γίνει και αυτόματη εκτέλεση τους .Τα προγράμματα αυτά αντιγράφονται και στην μνήμη EEPROM του ελεγκτή έτσι ώστε να παραμένουν αποθηκευμένα και μετά την απενεργοποίηση ή επαναφορά του.

ΚΕΦΑΛΑΙΟ 3

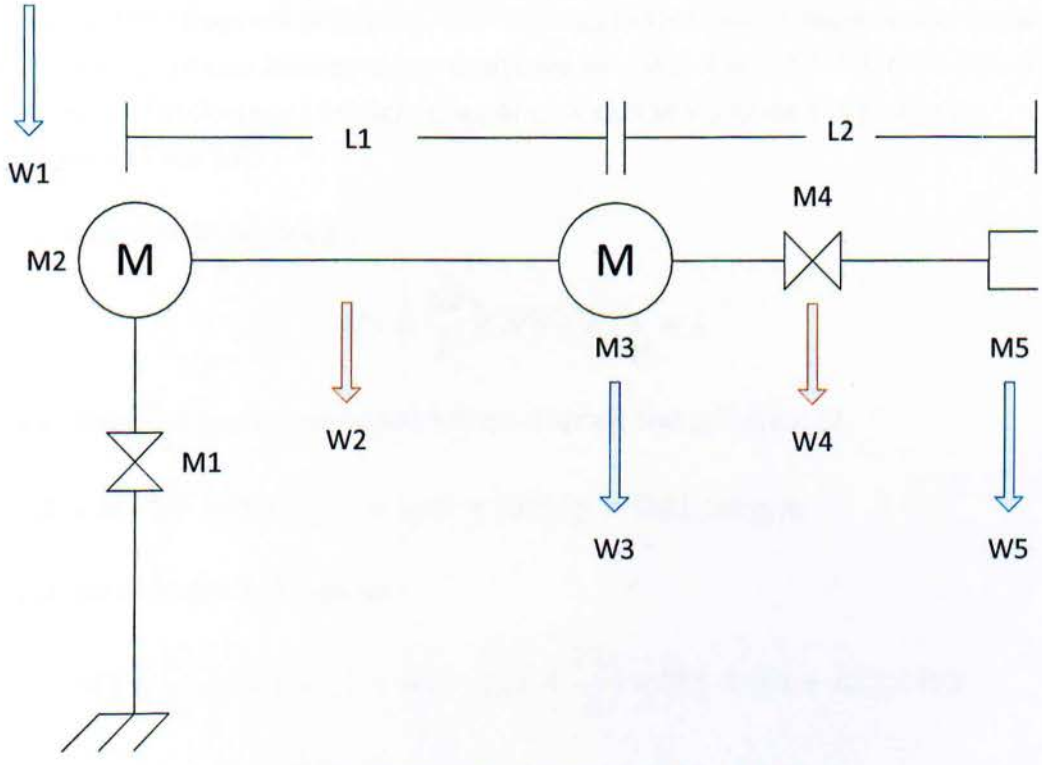
Το κεφάλαιο αυτό αφορά την σχεδίαση και κατασκευή του ρομποτικού βραχίονα, του χειριστηρίου, του κυκλώματος τροφοδοσίας, του ελεγκτή και της διεπαφής χρήστη-μηχανής.

3.1 Ανάλυση ρομποτικού βραχίονα

3.1.1 Μηχανική ανάλυση

Για την σχεδίαση του ρομποτικού βραχίονα είναι απαραίτητη η σχεδίαση του απλοποιημένου μηχανικού σχεδίου του βραχίονα για τον υπολογισμό της ροπής των κινητήρων των αρθρώσεων.

Το απλοποιημένο μηχανικό σχέδιο του βραχίονα είναι το ακόλουθο :



Σχήμα 9 Απλοποιημένο μηχανικό σχέδιο βραχίονα

Σε αυτό βλέπουμε την βάση του βραχίονα ,τοποθετημένη σε στερεό έδαφος.

Ο κινητήρας Μ1 είναι για την οριζόντια περιστροφική κίνηση του βραχίονα.

Οι κινητήρες Μ2 και Μ3 είναι για την κάθετη περιστροφική κίνηση του βραχίονα.

Ο κινητήρας Μ4 είναι για την στροφική κίνηση της άρθρωσης που περιστρέφει την αρπάγη και ο κινητήρας Μ5 για το κλείσιμο της αρπαγής.

Πάνω στο σχέδιο βλέπουμε τις μεταβλητές που έχουν σημασία στον υπολογισμό της ροπής κάθε κινητήρα. Αυτές είναι :

- ¨Wx¨ που είναι το βάρος των κινητήρων ή των συμπαγών μερών μεταξύ δυο αρθρώσεων.
- ¨Lx¨ που είναι το μήκος των στερεών σωμάτων του βραχίονα.

Είναι γνωστό ότι το γινόμενο της απόστασης με το βάρος μας δίνει ροπή. Επομένως χρησιμοποιώντας τις παραπάνω μεταβλητές μπορούμε να κάνουμε μια εκτίμηση της ροπής των κινητήρων για την σωστή επιλογή τους. Οι κινητήρες που αντιμετωπίζουν το μεγαλύτερο φορτίο είναι αυτοί της κάθετης περιστροφικής μετατόπισης του βραχιονα(M2,M3).

Υπολογίζοντας βάρος αντικειμένου W5=50gr και μετρώντας τα βάρη των κινητήρων και των αλουμινένιων σωμάτων του βραχίονα W1=W3=48gr, W2=63gr και W4=80gr και τα μήκη L1=10cm και L2=14cm μπορούμε να υπολογίσουμε την ροπή των κινητήρων M2 και M3.

Για τον κινητήρα M3 έχουμε :

$$M3 = \frac{L2}{2} \times W4 + L2 \times W5$$

Θεωρώντας $\frac{L2}{2}$ το κατά προσέγγιση κέντρο βάρους του σώματος L2.

Άρα : $M3 = \frac{0.14m}{2} \times 0.08kg + 0.14m \times 0.05kg = 0.0126kg.m$

Και για τον κινητήρα M2 έχουμε :

$$M2 = \frac{L1}{2} \times W2 + L1 \times W3 + \left(L1 + \frac{L2}{2}\right) \times W4 + (L1 + L2) \times W5$$

Άρα :

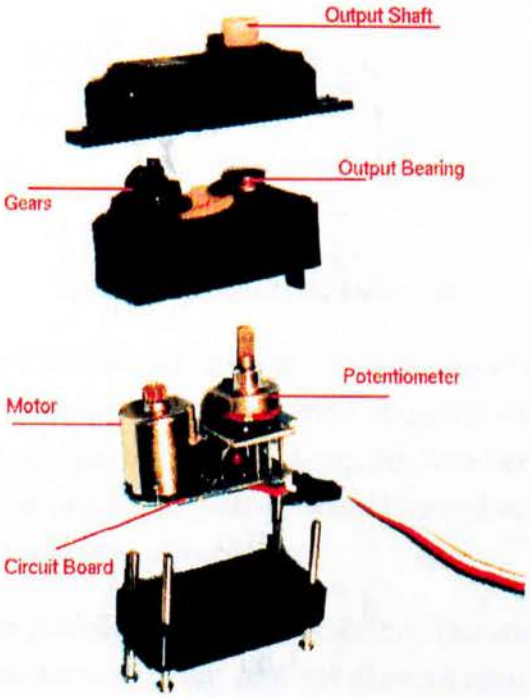
$$M2 = \frac{0.1m}{2} \times 0.063kg + 0.1m \times 0.048kg + \left(0.1m + \frac{0.14m}{2}\right) \times 0.08kg +$$

$$+(0.1m + 0.14m) \times 0.05kg = 0.03355kg.m$$

$$M2 = 3{,}355kg.cm \text{ και } M3 = 1{,}26kg.cm$$

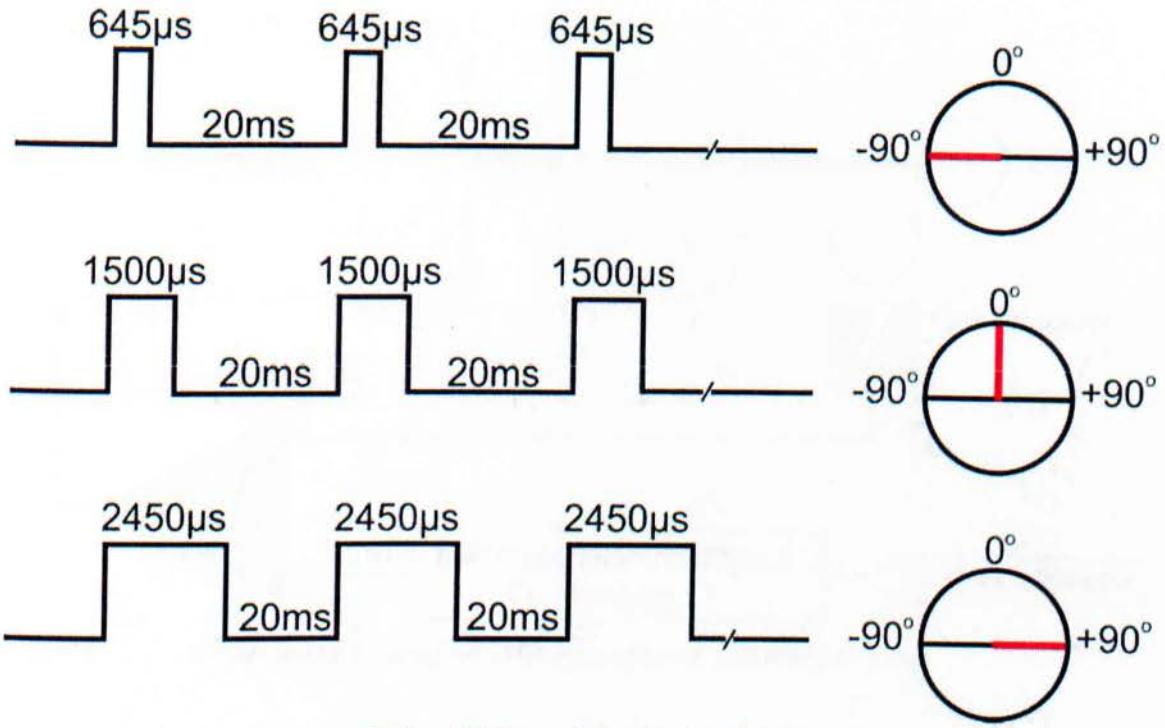
3.1.2 Κινητήρες Servo

Οι κινητήρες Servo είναι μικρές συσκευές που αποτελούνται από ένα ηλεκτροκινητήρα συνεχούς ρεύματος , ένα ηλεκτρονικό κύκλωμα που ελέγχει τη θέση του τελικού άξονα κίνησης και ένα κιβώτιο υποβιβασμού της σχέσης μετάδοσης του κινητήρα .



Σχήμα 10 Εσωτερική μορφή Κινητήρα Servo

Ο άξονας μπορεί να μετακινηθεί σε διάφορες θέσεις αν αποσταλεί στον Servo ένα κωδικοποιημένο σήμα. Πιο συγκεκριμένα, ο ελεγκτής διαμορφώνει και μεταδίδει στο Servo ηλεκτρικούς παλμούς, ανάλογα με τη θέση στην οποία πρέπει να περιστραφεί ο άξονας του. Οι ηλεκτρικοί παλμοί λαμβάνονται και αποκωδικοποιούνται από το Servo, με τη βοήθεια του κυκλώματος ελέγχου που περιλαμβάνεται σε αυτό.

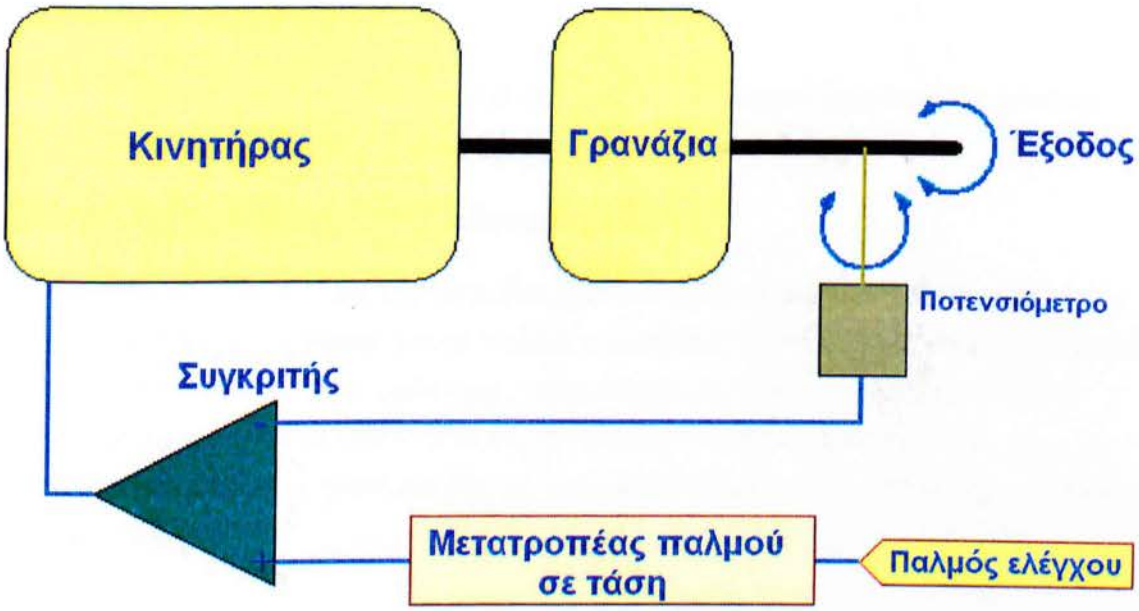


Σχήμα 11 Οριακές τιμές κινητήρα

Στη συνέχεια, μετά την αποκωδικοποίηση του ηλεκτρικού σήματος των παλμών, το κύκλωμα ελέγχου του σέρβο οδηγεί τον κινητήρα του στην κατάλληλη θέση (Σχήμα 12) . Όσο υπάρχει αυτό το σήμα στην γραμμή εισόδου του Servo τόσο αυτός θα διατηρεί τον άξονα του σε μια συγκεκριμένη θέση. Όταν αλλάζει το σήμα προκαλεί στον Servo να μεταβάλει την γωνία του άξονα.

Στη μέθοδο ελέγχου ανοιχτού βρόχου που χρησιμοποιείται στους κινητήρες Servo, υπάρχει επικοινωνία μιας κατεύθυνσης, από τον ελεγκτή προς το σύστημα που πρόκειται να ελεγχθεί. Δεν υπάρχει ανατροφοδότηση πληροφορίας από το σύστημα προς τον ελεγκτή. Για το λόγο αυτό η μέθοδος ανοιχτού βρόχου δεν χρησιμοποιεί αισθητήρες όπως η μέθοδος κλειστού βρόχου . Παρ' όλα αυτά, διαβάζοντας την τάση από τον κεντρικό ακροδέκτη του ποτενσιόμετρου που είναι ενσωματωμένο στον κινητήρα , μπορεί να χρησιμοποιηθεί και σαν συσκευή εισόδου , παρέχοντας ένα αισθητήριο θέσης. Αυτό μπορεί να φανεί πολύ χρήσιμο σε εφαρμογές όπως ανίχνευση συγκρούσεων σε ανενεργούς ρομποτικούς βραχίονες , μέτρηση ροπής του κινητήρα , απτικό έλεγχο με ανάδραση κλπ.

Σε πρακτικές εφαρμογές οι Servo χρησιμοποιούνται σε τηλεχειριζόμενα αεροπλάνα, αυτοκίνητα και στην ρομποτική. Οι κινητήρες αυτοί έχουν ένα εξαιρετικά μικρό μέγεθος αλλά είναι αρκετά ισχυροί για το μέγεθος τους.



Σχήμα 12 Εσωτερικό κύκλωμα οδήγησης κινητήρα Servo

Οι κινητήρες που χρησιμοποιήθηκαν σε αυτόν το ρομποτικό βραχίονα (και βάση του υπολογισμού της ροπής των κινητήρων) έχουν τα εξής χαρακτηριστικά :

Περιστροφή 180 μοίρες
Τάση λειτουργίας : 4.8-6 Volt DC

Οι κινητήρες περιστροφής της βάσης , περιστροφής και κλεισίματος της αρπάγης :

Ροπή εξόδου: 2.8-3.2 kg/cm
Ταχύτητα λειτουργίας: 0.18-0.20sec/60^{O}

Οι κινητήρες κάθετης περιστροφής του σώματος του βραχίονα :

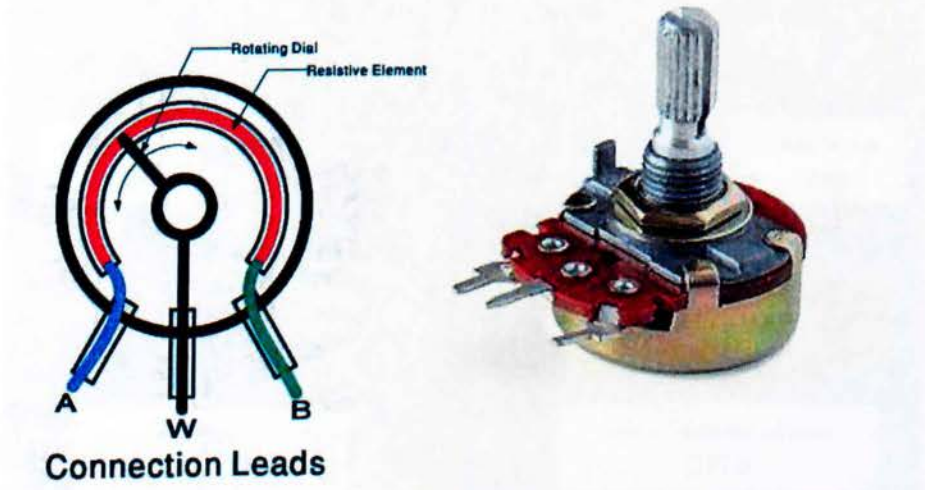
Ροπή εξόδου: 12kg/cm
Ταχύτητα λειτουργίας: 0.22 sec/60^{O}

3.2 Μηχανισμός χειρισμού

Το χειριστήριο είναι μια αλουμινένια κατασκευή ανάλογου μεγέθους με αυτό του βραχίονα και οι αρθρώσεις του δομούνται από ποτενσιόμετρα.

3.2.1 Αισθητήρια θέσης – ποτενσιόμετρα

Ένα ποτενσιόμετρο (Σχήμα 13) είναι μια χειροκίνητα ρυθμιζόμενη ηλεκτρική αντίσταση που χρησιμοποιείται σε πολλές ηλεκτρικές συσκευές για να καθορίζει τα επίπεδα της εξόδου. Για παράδειγμα , στον έλεγχο έντασης σε εξοπλισμό ήχου. Παρ' όλα αυτά , χρησιμοποιούμενο σε κατάλληλο μηχανισμό, μπορεί να λειτουργήσει και ως αισθητήριο θέσης, για παράδειγμα, σε ένα χειριστήριο ή ένα ποντίκι υπολογιστή.

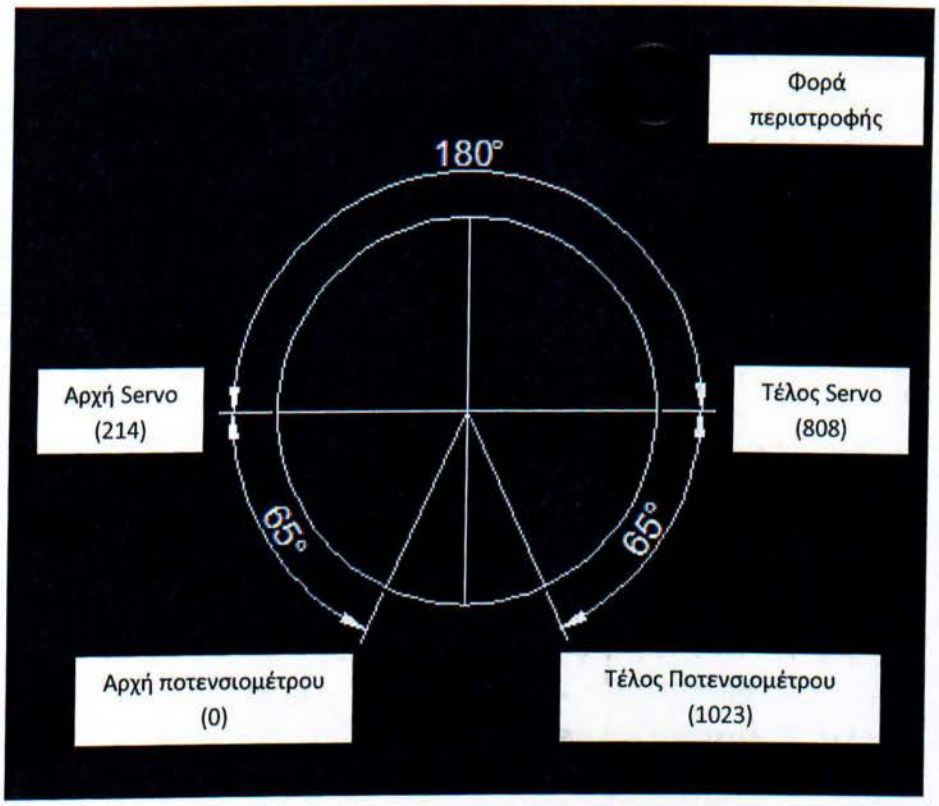


Σχήμα 13 Ποτενσιόμετρο

Στην κατασκευή αυτή χρησιμοποιούνται 5 γραμμικά ποτενσιόμετρα 1 kΩ το καθένα, ως αισθητήρια θέση για κάθε άρθρωση του χειριστηρίου . Στη προκειμένη περίπτωση , η τιμή της αντίστασης που προβάλουν τα ποτενσιόμετρα δεν έχει ιδιαίτερη σημασία .Ο μικροελεγκτής δέχεται την τιμή (0-5 volt) που επιστρέφει το ποτενσιόμετρο στην αναλογική θύρα και την μετατρέπει κατά αναλογία σε ακέραιους δεκαδικούς αριθμούς (0-1023)για την περεταίρω επεξεργασία .

3.2.2 Υπολογισμός μοιρών ποτενσιόμετρων

Όπως προαναφέρθηκε η τάση το ποτενσιόμετρου μετατρέπεται σε ακέραιους δεκαδικούς αριθμούς από 0 έως 1023. Τα ποτενσιόμετρα όμως έχουν κατά προσέγγιση 310° γωνία περιστροφής ενώ οι κινητήρες 180°. Συνεπώς πρέπει να γίνει υπολογισμός του εύρους των αναλογικών τιμών του ελεγκτή έτσι ώστε να ταυτίζεται η κίνηση όσο το δυνατό περισσότερο με αυτή του αντίστοιχου κινητήρα. Όπως φαίνεται από το Σχήμα 14, σε καρτεσιανό επίπεδο, η αρχή του ποτενσιομέτρου ξεκινά από τις 245° και καταλήγει δεξιόστροφα στις -65° ή αλλιώς στις 295°.



Σχήμα 14 Κίνηση ποτενσιόμετρου σε καρτεσιανό επίπεδο

Αφού έχουμε 1024 μονάδες στις 310 μοίρες, τότε έχουμε :

$$\frac{1024}{310} = 3{,}3\ \mu o \nu a \delta \varepsilon \varsigma\ a \nu a\ \mu o \iota \rho a$$

Άρα η τιμή του ποτενσιόμετρου στο σημείο εκκίνησης του κινητήρα Servo είναι :

$$65 \times 3{,}3 = 214{,}5$$

Ενώ στο σημείο τερματισμού είναι :

$$(65 + 180) \times 3{,}3 = 808{,}5$$

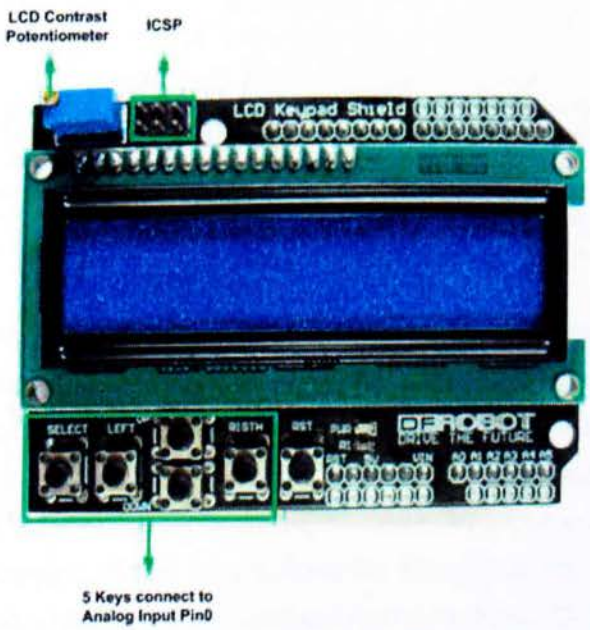
Επομένως, επειδή δεν μπορούν να χρησιμοποιηθούν δεκαδικά ψηφία, το χρήσιμο εύρος του ποτενσιόμετρου είναι 214-808.

3.3 Διεπαφή χρήστη-μηχανής

3.3.1 Πληκτρολόγιο και LCD οθόνη

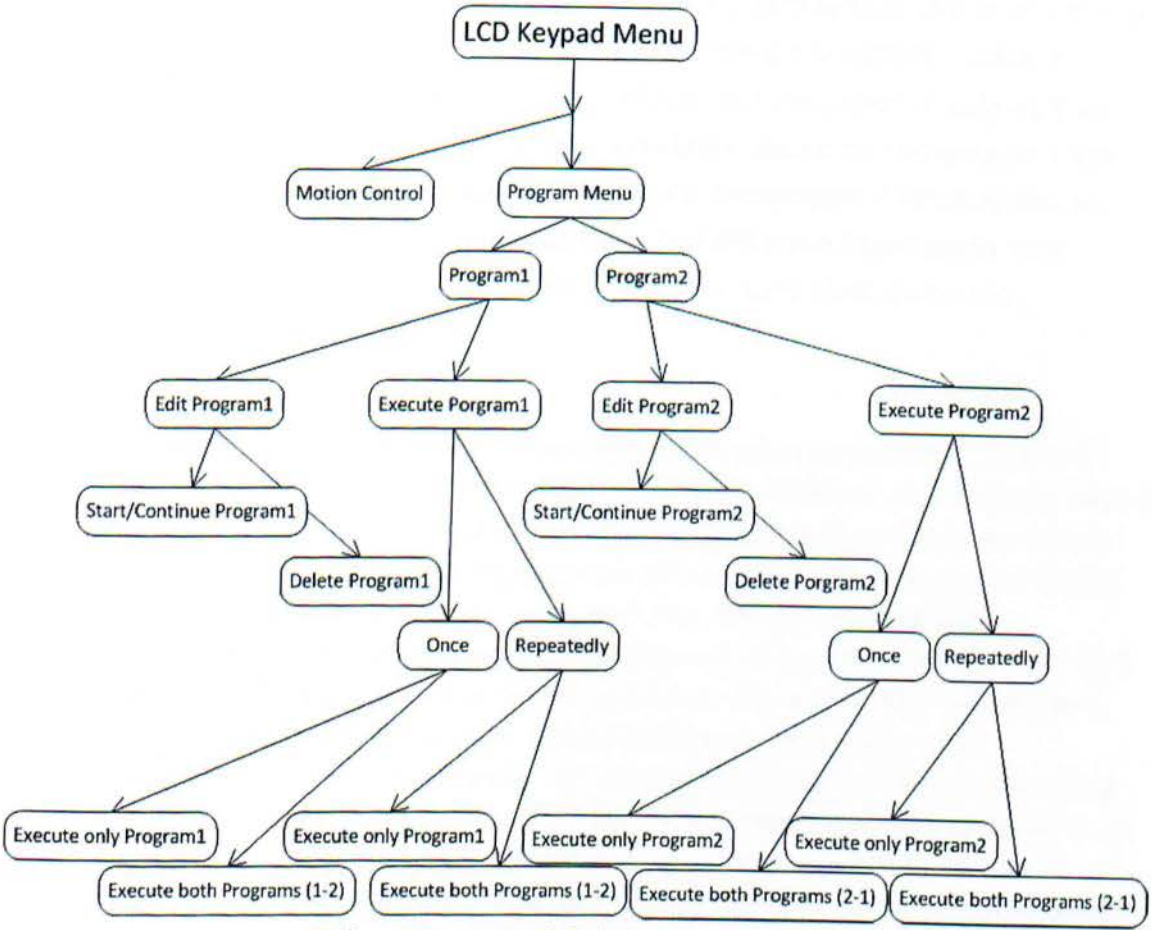
Μια οθόνη LCD είναι μια χαμηλού κόστους οθόνη. Είναι εύκολη η διασύνδεση της με έναν μικροελεγκτή εξαιτίας ενός ενσωματωμένου ελεγκτή στο πίσω μέρος της πλακέτας της οθόνης. Αυτός ο ελεγκτής είναι κοινός σε πολλές οθόνες (HD 44780), το οποίο σημαίνει ότι πολλοί μικροελεγκτές (συμπεριλαμβανομένου του Arduino) έχουν βιβλιοθήκες που κάνουν την εμφάνιση μηνύματα εύκολη.

Για εξοικονόμηση θυρών και μεγαλύτερη ευκολία στην διαχείριση του μενού , σε αυτή την κατασκευή χρησιμοποιήθηκε ένα LCD Keypad Shield (Σχήμα 15) που τοποθετείτε κατευθείαν πάνω στον μικροελεγκτή και περιλαμβάνει την οθόνη LCD 2x16, 5 στιγμιαία πλήκτρα καθώς και ένα ποτενσιόμετρο για την ρύθμιση της αντίθεσης και πίσω φωτισμού της οθόνης. Ο μικροελεγκτής χρησιμοποιεί τις ψηφιακές θύρες 4-10 για την επικοινωνία με την οθόνη ,ενώ για το πληκτρολόγιο χρησιμοποιεί μόλις μια αναλογική θύρα (θύρα 0). Αυτό επιτυγχάνεται διαβάζοντας την βασική τιμή μέσω ενός διαιρέτη τάσης 5 σταδίων .



Σχήμα 15 Εμπρόσθια εικόνα του LCD keypad shield

3.3.2 Μενού επιλογών



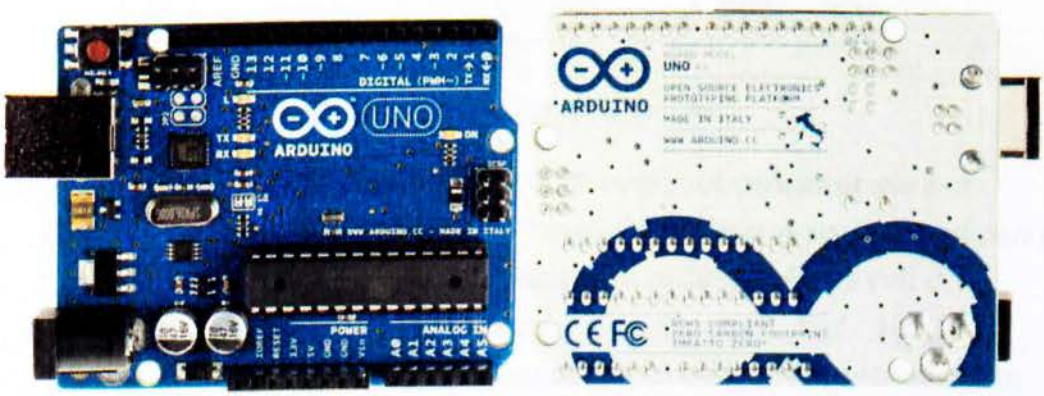
Σχήμα 16 Ιεραρχικό διάγραμμα του μενού

Όπως αναφέρθηκε σε προηγούμενη ενότητα , χρησιμοποιούνται 5 πλήκτρα για την διαχείριση του μενού. Τα πλήκτρα Up και Down μετατοπίζουν το κέρσορα της οθόνης στην αριστερή πλευρά , πάνω και κάτω ώστε να είναι εμφανής η επιλογή του χρήστη. Το πλήκτρο Right προωθεί τον χρήστη στις επόμενες επιλογές , ενώ το Left επιστρέφει στην προηγούμενη ομάδα επιλογών. Το πλήκτρο Select έχει συγκεκριμένες λειτουργίες , όπως να αποθηκεύει θέσεις και να σταματάει την εκτέλεση ενός προγράμματος ή του χειρισμού του βραχίονα. Οι πρώτες 2 επιλογές του μενού είναι το Motion control , που προωθεί τον χρήστης στην λειτουργία του βραχίονα μέσω του χειριστηρίου 5 βαθμών ελευθερίας και το Program menu που δίνει πρόσβαση στα 2 προγράμματα της εφαρμογής. Σε κάθε πρόγραμμα ο χειριστής έχει την δυνατότητα να το επεξεργαστεί ,μέσω της επιλογής Edit Program είτε διαγράφοντας το υπάρχον και ξεκινώντας νέο πρόγραμμα είτε συνεχίζοντας το υπάρχον πρόγραμμα από την τελευταία θέση που είναι αποθηκευμένη στην μνήμη του ελεγκτή με μέγιστο αριθμό θέσεων τις 25.

Επίσης , από την επιλογή Execute Program , ο χρήστης έχει την επιλογή να εκτελέσει το πρόγραμμα είτε επαναλαμβανόμενα μέχρι να γίνει αίτημα τερματισμού του προγράμματος από τον χρήστη είτε εκτελώντας ένα ενιαίο κύκλο, όπου το πρόγραμμα τερματίζει αυτόματα μετά το τέλος του κύκλου. Τέλος ο χρήστης έχει την δυνατότητα να επιλέξει εκτέλεση του ενός προγράμματος ή και των 2 σε αλληλουχία. Από το πρόγραμμα 1 εκτελείτε πρώτα το πρόγραμμα 1 και στην συνέχεια το δεύτερο πρόγραμμα ενώ από το πρόγραμμα 2 εκτελούνται με αντίθετη σειρά. Με αυτό τον συνδυασμό είναι δυνατή και η δημιουργία ενός ενιαίου προγράμματος, έως και 50 θέσεων , εφόσον αυτό είναι αναγκαίο .

3.4 Ο Μικροελεγκτης Arduino

Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring. Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider. Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz .Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με ένα *bootloader*, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής. Οι πλακέτες Arduino προγραμματίζονται μέσω USB, εφαρμόζοντας ένα τσίπ προσαρμογέα USB-to-serial όπως το FTDI FT232.



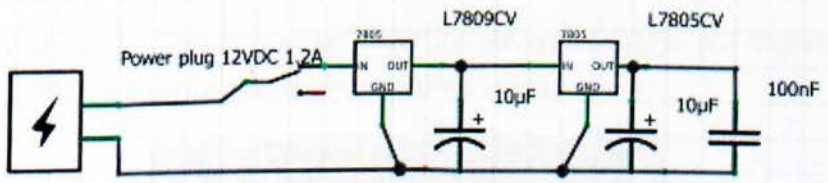
Σχήμα 17 Εμπρόσθια και πίσω εικόνα του Arduino Uno Rev3

Η έκδοση της πλατφόρμας που χρησιμοποιείται σε αυτή την εφαρμογή είναι η Arduino Uno Rev3 με μικροελεγκτή ATmega328 .

Η πλατφόρμα απαιτεί 7-12 volt τάση τροφοδοσία και λειτουργεί στα 5 volt. Παρέχει 14 ψηφιακές θύρες εισόδου/εξόδου , εκ των οποίων οι 6 μπορούν να χρησιμοποιηθούν και για διαμόρφωση πλάτους παλμού (PWM) ,καθώς και 6 αναλογικές εισόδους .Επίσης ο μικροελεγκτής έχει ενσωματωμένη Flash memory 32KB , SRAM 2KB και EEPROM 1KB.

3.5 Κύκλωμα τροφοδοσίας

Σε αυτή την κατασκευή χρειάζεται μια πυγή τάσης 7-12Volt για την τροφοδοσία του μικροελεγκτή , μια πηγή 5Volt για τα ποτενσιόμετρα ,τα πλήκτρα και την LCD οθόνη και 4.8-6 Volt για την λειτουργιά των κινητήρων. Η πλατφόρμα του Arduino έχει το προνόμιο να παρέχει πηγή τάσης 5 Volt από την οποία μπορεί να γίνει η τροφοδοσία συσκευών οι οποίες δεν έχουν υψηλές απαιτήσεις για την ένταση του ρεύματος, όπως είναι το πληκτρολόγιο , τα ποτενσιόμετρα και η LCD οθόνη. Ένας μικρός κινητήρας Servo όπως αυτοί που χρησιμοποιούμε σε αυτή την εφαρμογή θα μπορούσε να τροφοδοτηθεί από τον μικροελεγκτή εφόσον δεν έχει φορτίο, στην προκειμένη περίπτωση όμως η τροφοδοσία 5 κινητήρων με πιθανό φορτίο μπορεί να προκαλέσει ανεπιθύμητες επαναφορές (Reset) του ελεγκτή , χαμηλή τάση στην έξοδο και αύξηση της θερμοκρασίας που πιθανό να καταστρέψει την πλακέτα.
Γι αυτό το λόγο για την τροφοδότηση του βραχίονα είναι απαραίτητο ένα εξωτερικό κύκλωμα τροφοδοσίας.



Σχήμα 18 Κύκλωμα τροφοδοσίας

Χρησιμοποιώντας ένα τροφοδοτικό 12Volt 1.2A συνεχούς ρεύματος και ένα ρυθμιστή τάσης L7809CV παρέχονται 9 Volt στον μικροελεγκτή και στη συνέχεια με ένα L7805CV και πυκνωτές εξομάλυνσης παρέχεται σταθερή τάση 5Volt στον βραχίονα. Ακόμα χρησιμοποιούνται πυκνωτές, ένας ηλεκτρολυτικός 10μF στην είσοδο και στην έξοδο του ολοκληρωμένου (σε παράλληλη συνδεσμολογία) και ένας πολυεστερικός 0.1μF στην έξοδο του, ώστε να φιλτράρεται τυχόν θόρυβος ή υψηλής συχνότητας AC σήματα που μπορεί να είναι στην DC γραμμή τάσης ,και είναι επιλεγμένοι σύμφωνα με τις προδιαγραφές του ολοκληρωμένου .

ΚΕΦΑΛΑΙΟ 4

4.1 Περιβάλλον προγραμματισμού

Το Arduino IDE (Integrated Development Environment) είναι μια cross-platform εφαρμογή γραμμένη σε Java, και προέρχεται από τον IDE για τη γλώσσα προγραμματισμού Processing. Έχει σχεδιαστεί για να εισαγάγει τον προγραμματισμό στους καλλιτέχνες και τους νέους φορείς που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστής και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα.

Το Arduino IDE έρχεται με μια βιβλιοθήκη γραμμένη σε C / C + + που ονομάζεται "Wiring", η οποία κάνει πολλές κοινές λειτουργίες εισόδου / εξόδου πολύ πιο εύκολες. Τα προγράμματα του Arduino είναι γραμμένα σε C / C + + και οι χρήστες πρέπει μόνο να ορίσουν δύο λειτουργίες για να γίνει ένα πρόγραμμα εκτελέσιμο :

setup()-Είναι μια συνάρτηση που εκτελείτε μια φορά μόνο κατά την έναρξη του προγράμματος και μπορεί να χρησιμοποιηθεί για προετοιμασία ρυθμίσεων όπως για παράδειγμα ο ορισμός της ταχύτητας μετάδοσης δεδομένων της σειριακής θύρας, η αντιστοίχηση συσκευών σε συγκεκριμένες θύρες και η αρχικοποίηση μεταβλητών.

loop()-Είναι μια συνάρτηση που καλείται επανειλημμένα μέχρι την απενεργοποίηση της συσκευής. Μέσα σε αυτή δομείται το κύριο πρόγραμμα.

Σχήμα 19 Arduino IDE

Το Arduino IDE χρησιμοποιεί το GNU toolchain και AVR libc για την κατάρτιση των προγραμμάτων, και χρησιμοποιεί avrdude για να φορτώνει τα προγράμματα στην πλακέτα.
Καθώς η πλατφόρμα Arduino χρησιμοποιεί μικροελεγκτές της Atmel, περιβάλλοντα ανάπτυξης όπως το AVR Studio ή το νεότερο Studio Atmel, μπορούν να χρησιμοποιηθούν για την ανάπτυξη λογισμικού για το Arduino. Για εκπαιδευτικούς σκοπούς υπάρχει και τρίτος γραφικό περιβάλλον ανάπτυξης που ονομάζεται Minibloq και διατίθεται υπό μια διαφορετική open source άδεια.

4.2 Πρόγραμμα ελέγχου

Στο πρόγραμμα του μικροελεγκτή χρησιμοποιούνται 3 βιβλιοθήκες (LiquidCrystal.h, Servo.h, EEPROM.h) για την διαχείριση της οθόνης, των κινητήρων του βραχίονα και την χρήση των θέσεων μνήμης της EEPROM.

Στην έναρξη εισάγονται οι βιβλιοθήκες αυτές, ορίζονται διάφορες μεταβλητές και συσκευές και δημιουργούνται 10 πίνακες 25 θέσεων για την αποθήκευση των προγραμμάτων που θα δημιουργούνται από τον χρήστη.

Στη συνέχεια εκτελείται η συνάρτηση setup() στην οποία γίνεται η αντιστοίχηση των κινητήρων Servo του βραχίονα με τις ψηφιακές θύρες, ορίζονται οι αρχικές τιμές διαφόρων μεταβλητών του προγράμματος και δίνεται μια αρχική θέση στον βραχίονα (Home position). Επίσης, επαναφέρονται στο τρέχον πρόγραμμα, τα προγράμματα που είχαν αποθηκευτεί από τον χειριστή στην μνήμη EEPROM και τυπώνεται ένα μήνυμα στην LCD οθόνη που σηματοδοτεί την έναρξη του κύριου προγράμματος.

Το κύριο πρόγραμμα, λόγο όγκου έχει διασπαστεί σε μικρές υπορουτίνες, ώστε να είναι ευδιάκριτο για ποιες λειτουργίες χρησιμοποιούνται. Επομένως μέσα στην συνάρτηση loop() καλείται μόνο μια υπορουτίνα.

Οι ρουτίνες αυτές χωρίζονται σε τρεις κύριες κατηγορίες. Η πρώτη κατηγορία έχει δυο υπορουτίνες (menuup(), menudown()) που χρησιμοποιούνται για την μετακίνηση του κέρσορα της οθόνης πάνω και κάτω προκειμένου να μπορεί ο χρήστης να βλέπει σε πια επιλογή βρίσκεται και μια (read_LCD_buttons()) για την αναγνώριση των επιλογών του χειριστή στο πληκτρολόγιο. Η δεύτερη κατηγορία περιλαμβάνει μια ρουτίνα (motioncontrol()) για τον χειρισμό του βραχίονα , μια για την δημιουργία των προγραμμάτων (mcprogramming()), μια για την διαγραφή τους (erase()) και δυο για την εκτέλεση των προγραμμάτων (executeonce(), executeloop()). Τέλος η τρίτη κατηγορία περιλαμβάνει εννέα υπορουτίνες (menuscreen#(), όπου # αριθμός από 1 έως 9) που καλούνται ιεραρχικά η κάθε μια από την προηγούμενη, για την δόμηση του μενού και την εμφάνιση των μηνυμάτων σε κάθε επίπεδο του μενού επιλογών. Στην συνέχεια παρατίθεται το πρόγραμμα.

Πρόγραμμα Ελεγκτή :

```
#include <LiquidCrystal.h>
#include <Servo.h>
#include <EEPROM.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

int j=0;//counter for LCD cursor
int x=0;//matrix position holder for positiontable1
int y=0;//matrix position holder for positiontable2
int z=0;//matrix position holder for positiontable3
int c=0;//matrix position holder for positiontable4
int v=0;//matrix position holder for positiontable5

Servo baseservo,tilt1servo,tilt2servo;
Servo wristservo,gripperservo;

boolean flagmotion,flagmenu2,flagmenu3;
boolean flagmenu4,flagmenu5,flagmenu6;
boolean flagmenu7,flagmenu8,flagmenu9;
boolean flagexitplay,pro1,pro2;
boolean flagyes,flagno;
int lcd_key     = 0;
int adc_key_in  = 0;
int potpin1 = 1;
int potpin2 = 2;
int potpin3 = 3;
int potpin4 = 4;
int potpin5 = 5;
int val1,val2,val3,val4,val5;

// define some values used by the panel and buttons
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
```

```
#define MAX_POSITIONS 25  // Maximum number of positions that can be stored
// Tables to hold each position of each servo motor for program1
int positiontable1_1[MAX_POSITIONS];
int positiontable1_2[MAX_POSITIONS];
int positiontable1_3[MAX_POSITIONS];
int positiontable1_4[MAX_POSITIONS];
int positiontable1_5[MAX_POSITIONS];
// Tables to hold each position of each servo motor for program2
int positiontable2_1[MAX_POSITIONS];
int positiontable2_2[MAX_POSITIONS];
int positiontable2_3[MAX_POSITIONS];
int positiontable2_4[MAX_POSITIONS];
int positiontable2_5[MAX_POSITIONS];

int positioncount;// Number of positions recorded in program1
int positioncount2;// Number of positions recorded in program2
int k; //for saving position in program1
int k2; //for saving position in program2

void setup() {
  baseservo.attach(2);
  tilt1servo.attach(3);
  tilt2servo.attach(11);
  wristservo.attach(12);
  gripperservo.attach(13);
  flagmotion=false;
  flagmenu2=false;
  flagmenu3=false;
  flagmenu4=false;
  flagmenu5=false;
  flagmenu6=false;
  flagmenu7=false;
  flagmenu8=false;
  flagmenu9=false;
  flagexitplay=false;
  flagno=false;
  flagyes=false;
  pro1=false;
  pro2=false;
```

```
  //Home position
  baseservo.write(89);
  tilt1servo.write(70);
  tilt2servo.write(50);
  wristservo.write(89);
  gripperservo.write(44);
  //Reading from the EEPROM memory the existing programs
  for(int i=0;i<25;i++){
   positiontable1_1[i]=EEPROM.read(i);
   positiontable1_2[i]=EEPROM.read(i+25);
   positiontable1_3[i]=EEPROM.read(i+50);
   positiontable1_4[i]=EEPROM.read(i+75);
   positiontable1_5[i]=EEPROM.read(i+100);
   positiontable2_1[i]=EEPROM.read(i+125);
   positiontable2_2[i]=EEPROM.read(i+150);
   positiontable2_3[i]=EEPROM.read(i+175);
   positiontable2_4[i]=EEPROM.read(i+200);
   positiontable2_5[i]=EEPROM.read(i+225);
  }
  positioncount=EEPROM.read(500);
  positioncount2=EEPROM.read(501);
  if(positioncount==99){
    k=0;
  }
  else{
    k=positioncount+1;
  }
  if(positioncount2==99){
    k2=0;
  }
  else{
    k2=positioncount2+1;
  }
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd.setCursor(1,0);
  // Print a message to the LCD.
  lcd.print("TEI of PIRAEUS");
  delay(5000);
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("MENU");
  delay(4000);
```

```
}

//--------------------------------------------------Subroutines for handling buttons and LCD screen .

// This subroutine read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);
 delay(250);//debounce time
 // read the value from the sensor
 // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
 // we add approx 50 to those values and check to see if we are close
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 195)  return btnUP;
 if (adc_key_in < 380)  return btnDOWN;
 if (adc_key_in < 555)  return btnLEFT;
 if (adc_key_in < 790)  return btnSELECT;
}

// These subroutines moves the screen cursor up and down .
void menuup()
{
 if(j==0)
 {
  lcd.noCursor();
  delay(200);
  lcd.cursor();
 }
 if(j==1)
 {
  lcd.noCursor();
  lcd.setCursor(0,0);
  lcd.cursor();
  j=0;
 }
}
```

```
void menudown()
{
 if(j==0)
 {
  lcd.noCursor();
  lcd.setCursor(0,1);
  lcd.cursor();
  j=1;
 }
 if(j==1)
 {
  lcd.noCursor();
  delay(200);
  lcd.cursor();
 }
}
//-Subroutines for the Motion control,programming,erasing and execution of the programs .

//This subroutine is for controling the arm .
void motioncontrol()
{
 while(flagmotion==true)
 {
  val1 = analogRead(potpin1);
  val2 = analogRead(potpin2);
  val3 = analogRead(potpin3);
  val4 = analogRead(potpin4);
  val5 = analogRead(potpin5);
  if(val1<809 && val1>213)  val1 = map(val1, 214, 808, 1, 178);
  else if(val1<214)  val1=1;
  else val1=178;
  if(val2<809 && val2>213)  val2 = map(val2, 214, 808, 1, 178);
  else if(val2<214)  val2=1;
  else val2=178;
  if(val3<809 && val3>213)  val3 = map(val3, 214, 808, 1, 178);
  else if(val3<214)  val3=1;
  else val3=178;
  if(val4<809 && val4>213)  val4 = map(val4, 214, 808, 1, 178);
  else if(val4<214)  val4=1;
  else val4=178;
  if(val5<809 && val5>213)  val5 = map(val5, 214, 808, 1, 178);
  else if(val5<214)  val5=1;
```

```
    else val5=178;

    baseservo.write(val1);
    tilt1servo.write(val2);
    tilt2servo.write(val3);
    wristservo.write(val4);
    gripperservo.write(val5);
    delay(10);
    adc_key_in = analogRead(0);
    if (adc_key_in<790 && adc_key_in>741)// read button LEFT
    {
     flagmotion=false;
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Motion Control");
     lcd.setCursor(1,1);
     lcd.print("Program menu");
     lcd.setCursor(0,0);
     j=0;
     lcd.cursor();
    }
  }
}
//This subroutine is for programming .
void mcprogramming()
{
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Save positions");
 while(flagmotion==true)
 {
  val1 = analogRead(potpin1);
  val2 = analogRead(potpin2);
  val3 = analogRead(potpin3);
  val4 = analogRead(potpin4);
  val5 = analogRead(potpin5);
```

```
 if(val1<809 && val1>213)  val1 = map(val1, 214, 808, 1, 178);
 else if(val1<214)  val1=1;
 else val1=178;
 if(val2<809 && val2>213)  val2 = map(val2, 214, 808, 1, 178);
 else if(val2<214)  val2=1;
 else val2=178;
 if(val3<809 && val3>213)  val3 = map(val3, 214, 808, 1, 178);
 else if(val3<214)  val3=1;
 else val3=178;
 if(val4<809 && val4>213)  val4 = map(val4, 214, 808, 1, 178);
 else if(val4<214)  val4=1;
 else val4=178;
 if(val5<809 && val5>213)  val5 = map(val5, 214, 808, 1, 178);
 else if(val5<214)  val5=1;
 else val5=178;
 baseservo.write(val1);
 tilt1servo.write(val2);
 tilt2servo.write(val3);
 wristservo.write(val4);
 gripperservo.write(val5);
 adc_key_in = analogRead(0);
 delay(10);
 if (adc_key_in<790 && adc_key_in>741)// read the button SELECT
 {
  if (flagmenu4==true)
  {
   if(k<MAX_POSITIONS)
   {
    positiontable1_1[k]=val1;
    positiontable1_2[k]=val2;
    positiontable1_3[k]=val3;
    positiontable1_4[k]=val4;
    positiontable1_5[k]=val5;
    positioncount=k;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Saving Position:");
    lcd.setCursor(0,1);
    lcd.print(positioncount+1);
    delay(500);
    lcd.clear();
    lcd.setCursor(0,0);
    j=0;
```

```
    k=k+1;
   }
   else
   { //Show that program 1 is complete
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Program1");
    lcd.setCursor(1,1);
    lcd.print("Completed");
    delay(2000);
    lcd.clear();
   }
  }
  else if(flagmenu7==true)
  {
   if(k2<MAX_POSITIONS)
   {
    positiontable2_1[k2]=val1;
    positiontable2_2[k2]=val2;
    positiontable2_3[k2]=val3;
    positiontable2_4[k2]=val4;
    positiontable2_5[k2]=val5;
    positioncount2=k2;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Saving Positiion:");
    lcd.setCursor(0,1);
    lcd.print(positioncount2+1);
    delay(500);
    lcd.clear();
    lcd.setCursor(0,0);
    j=0;
    k2=k2+1;
   }
   else
   { //Show that Program 2 is complete
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Program2");
    lcd.setCursor(1,1);
    lcd.print("Completed");
    delay(2000);
    lcd.clear();
   }
  }
```

```
      }

      adc_key_in = analogRead(0);
      if(adc_key_in<555 && adc_key_in>504)//read button LEFT
      {
       //Rewrite programs in EEPROM memory
       for(int i=0;i<25;i++){
        EEPROM.write(i, positiontable1_1[i]);
        EEPROM.write(i+25, positiontable1_2[i]);
        EEPROM.write(i+50, positiontable1_3[i]);
        EEPROM.write(i+75, positiontable1_4[i]);
        EEPROM.write(i+100, positiontable1_5[i]);
        EEPROM.write(i+125, positiontable2_1[i]);
        EEPROM.write(i+150, positiontable2_2[i]);
        EEPROM.write(i+175, positiontable2_3[i]);
        EEPROM.write(i+200, positiontable2_4[i]);
        EEPROM.write(i+225, positiontable2_5[i]);
       }
       EEPROM.write(500, positioncount);
       EEPROM.write(501, positioncount2);

       flagmotion=false;
       lcd.clear();
       lcd.setCursor(1,0);
       lcd.print("St/cont. Program");
       lcd.setCursor(1,1);
       lcd.print("Erase Program");
       lcd.setCursor(0,0);
       j=0;
       lcd.cursor();
     }
    }
  }
void erase()
{
 if (flagmenu4==true){
  k=0;//repositioning k to 0 position and erasing position 0 only.
  positioncount=99;
  // Erase the program1 from EEPROM memory
  for(int i=0;i<125;i++){
   EEPROM.write(i,0);
  }
  EEPROM.write(500,99);
 }
```

```
   else if(flagmenu7==true){
    k2=0;//repositioning k2 to 0 position and erasing position 0 only.
    positioncount2=99;
    // Erase the program2 from EEPROM memory
    for( int i=125;i<250;i++){
     EEPROM.write(i,0);
    }
    EEPROM.write(501,99);
   }
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("Program Erased");
   delay(2000);
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("St/Cont. Program");
   lcd.setCursor(1,1);
   lcd.print("Erase Program");
   lcd.setCursor(0,0);
   j=0;
   lcd.cursor();
}
//This subroutine is for single execution of a program
//or both programs in series .
void executeonce(){
 if (pro1==true){
  for(int i=0;i<=positioncount;i++)
  {
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("Position:");
   lcd.print(i+1);
   x = positiontable1_1[i];
   y = positiontable1_2[i];
   z = positiontable1_3[i];
   c = positiontable1_4[i];
   v = positiontable1_5[i];
   baseservo.write(x);
   tilt1servo.write(y);
   tilt2servo.write(z);
   wristservo.write(c);
   gripperservo.write(v);
   delay(1500);
```

```
    }
   }
   if (pro2==true){
    for(int i=0;i<=positioncount2;i++)
    {
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Position:");
     lcd.print(i+1);
     x = positiontable2_1[i];
     y = positiontable2_2[i];
     z = positiontable2_3[i];
     c = positiontable2_4[i];
     v = positiontable2_5[i];
     baseservo.write(x);
     tilt1servo.write(y);
     tilt2servo.write(z);
     wristservo.write(c);
     gripperservo.write(v);
     delay(1500);
    }
   }
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("Once");
   lcd.setCursor(1,1);
   lcd.print("Repeatedly");
   lcd.setCursor(0,0);
   j=0;
   lcd.cursor();
}
//This subroutine is for executing a program repeatedly ,
//or both program in series, 2 after 1 or reverse.
void executeloop(){
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Select to exit");
 lcd.setCursor(1,1);
 lcd.print("Position:");
 while(flagexitplay==true){
```

```
    if (pro1==true){
     for(int i=0;i<=positioncount;i++)
     {
      adc_key_in = analogRead(0);
      if (adc_key_in<790 && adc_key_in>741){ // read the button SELECT
       flagexitplay=false;
       lcd.setCursor(15,1);
       lcd.print("E");
      }
      lcd.setCursor(12,1);
      lcd.print(i+1);
      x = positiontable1_1[i];
      y = positiontable1_2[i];
      z = positiontable1_3[i];
      c = positiontable1_4[i];
      v = positiontable1_5[i];
      baseservo.write(x);
      tilt1servo.write(y);
      tilt2servo.write(z);
      wristservo.write(c);
      gripperservo.write(v);
      delay(1500);
      adc_key_in = analogRead(0);
      if (adc_key_in<790 && adc_key_in>741){ // read the button Select
       flagexitplay=false;
       lcd.setCursor(15,1);
       lcd.print("E");
      }
    }
  }
  //This condition is for reversing the sequence of the programs
  if (flagmenu8==true && flagyes==true && j==1){
   pro1=true;
  }
```

```
      if(pro2==true){
       for(int i=0;i<=positioncount2;i++)
       {
        adc_key_in = analogRead(0);
        if (adc_key_in<790 && adc_key_in>741){ // read the button Select
         flagexitplay=false;
         lcd.setCursor(15,1);
         lcd.print("E");
        }
        lcd.setCursor(12,1);
        lcd.print(i+1);
        x = positiontable2_1[i];
        y = positiontable2_2[i];
        z = positiontable2_3[i];
        c = positiontable2_4[i];
        v = positiontable2_5[i];
        baseservo.write(x);
        tilt1servo.write(y);
        tilt2servo.write(z);
        wristservo.write(c);
        gripperservo.write(v);
        delay(1500);
        adc_key_in = analogRead(0);
        if (adc_key_in<790 && adc_key_in>741){ // read the button SELECT
         flagexitplay=false;
         lcd.setCursor(15,1);
         lcd.print("E");
        }
       }
      }
     }
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Once");
     lcd.setCursor(1,1);
     lcd.print("Repeatedly");
     lcd.setCursor(0,0);
     j=0;
     lcd.cursor();
}
```

```
//-------------------------------------------------Subroutines for handling the Menu.
void menuscreen1()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Motion Control");
 lcd.setCursor(1,1);
 lcd.print("Program menu");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
 while(flagmenu2==false){
  lcd_key = read_LCD_buttons();  // read the buttons
  switch (lcd_key)               // depending on which button was pushed, we perform an action
  {
  case btnRIGHT:
   {
    if (j==0){
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Press Select");
     lcd.setCursor(1,1);
     lcd.print("to exit");
     flagmotion=true;
     motioncontrol();  //Motion control subroutine
    }
    if(j==1) {
     flagmenu2=true;
     menuscreen2();
    }
    break;
  }
 case btnLEFT:
  {
   //Home position2
   baseservo.write(89);
   tilt1servo.write(70);
   tilt2servo.write(50);
   wristservo.write(89);
   gripperservo.write(44);
   break;
 }
```

```
      case btnUP:
       {
        menuup();
        break;
       }
      case btnDOWN:
       {
        menudown();
        break;
       }
     case btnSELECT:
      {
       lcd.noDisplay();
       delay(500);
       lcd.display();
       delay(500);
       break;
      }
    }
  }
}
void menuscreen2()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Program1 ->");
 lcd.setCursor(1,1);
 lcd.print("Program2 ->");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
 while (flagmenu2==true){
  lcd_key = read_LCD_buttons();  // read the buttons
  switch (lcd_key)               // depending on which button was pushed, we perform an action
  {
  case btnRIGHT:
   {
    if (j==0){
     flagmenu3=true;
     menuscreen3();
    }
```

```
          if(j==1) {
           flagmenu6=true;
           menuscreen6();
          }
          break;
         }
       case btnLEFT:
        {
         flagmenu2=false;
         lcd.clear();
         lcd.setCursor(1,0);
         lcd.print("Motion Control");
         lcd.setCursor(1,1);
         lcd.print("Program menu");
         lcd.setCursor(0,0);
         j=0;
         lcd.cursor();
         break;
        }
      case btnUP:
       menuup();
       break;
      case btnDOWN:
       menudown();
       break;
      case btnSELECT:
       {
        lcd.noDisplay();
        delay(500);
        lcd.display();
        delay(500);
        break;
       }
     }
   }
}
void menuscreen3()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Edit Program");
 lcd.setCursor(1,1);
 lcd.print("Execute Program");
```

```
lcd.setCursor(0,0);

j=0;
lcd.cursor();
while(flagmenu3==true){
 lcd_key = read_LCD_buttons();  // read the buttons
 switch (lcd_key)               // depending on which button was pushed, we perform an action
 {
 case btnRIGHT:
  {
   if (j==0){
    flagmenu4=true;
    menuscreen4();
   }
   if(j==1) {
    if (positioncount<25){
     flagmenu5=true;
     menuscreen5();
    }
    else
    {
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Program1 was");
     lcd.setCursor(1,1);
     lcd.print("erased");
     lcd.setCursor(0,0);
     delay(1000);
     lcd.clear();
     lcd.setCursor(1,0);
     lcd.print("Edit Program");
     lcd.setCursor(1,1);
     lcd.print("Execute Program");
     lcd.setCursor(0,0);
     j=0;
     lcd.cursor();
   }
  }
 break;
}
```

```
      case btnLEFT:
       {
        flagmenu3=false;
        lcd.clear();
        lcd.setCursor(1,0);
        lcd.print("Program1 ->");
        lcd.setCursor(1,1);
        lcd.print("Program2 ->");
        lcd.setCursor(0,0);
        j=0;
        lcd.cursor();
        break;
       }
      case btnUP:
       menuup();
       break;
      case btnDOWN:
       menudown();
       break;
      case btnSELECT:
       {
        lcd.noDisplay();
        delay(500);
        lcd.display();
        delay(500);
        break;
       }
    }
  }
}
void menuscreen4()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("St/Cont. Program");
 lcd.setCursor(1,1);
 lcd.print("Erase Program");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
```

```
while(flagmenu4==true){
 lcd_key = read_LCD_buttons();  // read the buttons
 switch (lcd_key)               // depending on which button was pushed, we perform an action
 {
 case btnRIGHT:
  {
    if (j==0){
     flagmotion=true;
     mcprogramming();
    }
    if(j==1) erase();
    break;
  }
 case btnLEFT:
  {
    flagmenu4=false;
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Edit Program");
    lcd.setCursor(1,1);
    lcd.print("Execute Program");
    lcd.setCursor(0,0);
    j=0;
    lcd.cursor();
    break;
  }
 case btnUP:
  menuup();
  break;
 case btnDOWN:
  menudown();
  break;
 case btnSELECT:
  {
   lcd.noDisplay();
   delay(500);
   lcd.display();
   delay(500);
   break;
  }
 }
}
}
```

```
void menuscreen5()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Once");
 lcd.setCursor(1,1);
 lcd.print("Repeatedly");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
 while(flagmenu5==true){
  lcd_key = read_LCD_buttons();  // read the buttons
  switch (lcd_key)               // depending on which button was pushed, we perform an action
  {
  case btnRIGHT:
   {
    flagmenu9=true;
    menuscreen9();
    break;
   }
  case btnLEFT:
   {
    flagmenu5=false;
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("Edit Program");
    lcd.setCursor(1,1);
    lcd.print("Execute Program");
    lcd.setCursor(0,0);
    j=0;
    lcd.cursor();
    break;
   }
 case btnUP:
  menuup();
  break;
 case btnDOWN:
  menudown();
  break;
```

```
    case btnSELECT:
     {
      lcd.noDisplay();
      delay(500);
      lcd.display();
      delay(500);
      break;
     }
    }
  }
}
void menuscreen6()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("Edit Program");
 lcd.setCursor(1,1);
 lcd.print("Execute Program");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
 while(flagmenu6==true){
  lcd_key = read_LCD_buttons();  // read the buttons
  switch (lcd_key)               // depending on which button was pushed, we perform an action
  {
  case btnRIGHT:
   {
    if (j==0){
     flagmenu7=true;
     menuscreen7();
    }
    if(j==1) {
     if (positioncount2<25){
      flagmenu8=true;
      menuscreen8();
     }
```

```
        else{
         lcd.clear();
         lcd.setCursor(1,0);
         lcd.print("Program2 was");
         lcd.setCursor(1,1);
         lcd.print("erased");
         lcd.setCursor(0,0);
         delay(1000);
         lcd.clear();
         lcd.setCursor(1,0);
         lcd.print("Edit Program");
         lcd.setCursor(1,1);
         lcd.print("Execute Program");
         lcd.setCursor(0,0);
         j=0;
         lcd.cursor();
        }
       }
       break;
     }
    case btnLEFT:
     {
       flagmenu6=false;
       lcd.clear();
       lcd.setCursor(1,0);
       lcd.print("Program1 ->");
       lcd.setCursor(1,1);
       lcd.print("Program2 ->");
       lcd.setCursor(0,0);
       j=0;
       lcd.cursor();
       break;
     }
   case btnUP:
    menuup();
    break;
   case btnDOWN:
    menudown();
    break;
```

```
    case btnSELECT:
     {
      lcd.noDisplay();
      delay(500);
      lcd.display();
      delay(500);
      break;
     }
    }
  }
}
void menuscreen7()
{
 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("St/Cont. Program");
 lcd.setCursor(1,1);
 lcd.print("Erase Program");
 lcd.setCursor(0,0);
 j=0;
 lcd.cursor();
 while(flagmenu7==true){
  lcd_key = read_LCD_buttons();  // read the buttons
  switch (lcd_key)               // depending on which button was pushed, we perform an action
  {
  case btnRIGHT:
   {
    if (j==0){
     flagmotion=true;
     mcprogramming();
    }
    if(j==1) erase();
    break;
   }
```