ΤΕΙ ΠΕΙΡΑΙΑ Τμ.ΑΥΤΟΜΑΤΙΣΜΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

΄΄ΚΑΤΑΣΚΕΥΗ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ ΜΕ ΑΥΤΟΜΑΤΗ ΚΑΙ ΧΕΙΡΟΚΙΝΗΤΗ ΛΕΙΤΟΥΡΓΙΑ΄΄

ΣΠΟΥΔΑΣΤΕΣ:

ΣΤΑΥΡΑΚΑΣ ΜΑΡΙΟΣ Α.Μ.33473

ΚΟΥΛΟΥΡΑ ΚΩΝΣΤΑΝΤΙΝΑ Α.Μ.36704

ΕΠΙΒΛΕΠΟΝ ΚΑΘΗΓΗΤΗΣ:

ΠΑΠΟΥΤΣΙΔΑΚΗΣ ΜΙΧΑΛΗΣ

ΑΙΓΑΛΕΩ,ΙΑΝΟΥΑΡΙΟΣ 2012

TEI OF PEIRAEUS Dep.AUTOMATION

FINAL YEAR PROJECT:

"CONSTRUCTION OF A ROBOTIC ARM USING AN AUTONOMOUS OR MANUAL OPERATION"

STUDENTS:

STAVRAKAS MARIOS 33473

KOULOURA KONSTANTINA 36704

SUPERVISOR:

PAPOUTSIDAKIS MICHAEL

EGALEO,JANUARY 2012

ΠΕΡΙΕΧΟΜΕΝΑ

Το **ρομπότ** είναι μια μηχανική συσκευή η οποία μπορεί να υποκαθιστά τον άνθρωπο σε διάφορες εργασίες. Πρόκειται για μία μηχανή που «αισθάνεται»,»σκέφτεται» και «επενεργεί»(sense,think,act),(βλέπε[5]).

Σύμφωνα όμως με το Robot Institute of America,ως ρομπότ μπορούμε να ορίσουμε ένα μηχανισμό σχεδιασμένο ώστε, μέσω προγραμματιζόμενων κινήσεων να μεταφέρει υλικά, τεμάχια, εργαλεία ή ειδικευμένες συσκευές, με σκοπό την επιτέλεση ποικιλίας εργασιών .Ένας τέτοιος μηχανισμός περιλαμβάνει συνήθως τις ακόλουθες συνιστώσες.

→Ένα μηχανολογικό υποσύστημα, το οποίο ενσωματώνει τη δυνατότητα του ρομπότ για εκτέλεση εργασιών Το υποσύστημα αυτόαποτελείται από μηχανισμούς που επιτρέπουν στο ρομπότ να κινείται, όπως αρθρώσεις, συστήματα μετάδοσης κίνησης, επενεργητές- 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 είναι υπεύθυνο εκτός των άλλων για την αποδοχή των εξωτερικών εντολών, την επεξεργασία τους ,τη μετάφρασή 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 μετρήσεως ,αισθητήρες, ηλεκτρονικά στοιχεία κλπ.

→Ένα σύστημα ελέγχου, το οποίο συνδυάζει κατάλληλα την αίσθηση με τη δράση, έτσι ώστε να λειτουργεί αποτελεσματικά και με επιθυμητό τρόπο το ρομπότ .Ο ελεγκτής του ρομπότ επιβλέπει και συντονίζει ολόκληρο το σύστημα, για τη σχεδίαση και υλοποίησή του δε απαιτείται ο συνδυασμός γνώσεων από πολλές γνωστικές περιοχές όπως είναι ο αυτόματος έλεγχος, η τεχνητή νοημοσύνη, η επιστήμη των υπολογιστών κλπ.

Τα ρομπότ μπορούν να χρησιμοποιηθούν ώστε να κάνουν εργασίες οι οποίες είτε είναι δύσκολες ή επικίνδυνες για να γίνουν απ' ευθείας από έναν άνθρωπο. Σε άλλες περιπτώσεις, χρησιμοποιούνται για να εκτελέσουν εργασίες ταχύτερα ή φθηνότερα απ' ότι ο άνθρωπος. Έτσι, μπορούν να χρησιμοποιηθούν στην αυτόματη παραγωγή μεγάλων ποσοτήτων κάποιου προϊόντος και με χαμηλότερο κόστος (για παράδειγμα, στις αλυσίδες παραγωγής).

Η λέξη ρομπότ προέρχεται από το σλαβικό robota που σημαίνει εργασία. Καθιερώθηκε ως όρος με την σημερινή του έννοια το 1920 από τον Τσέχο θεατρικό συγγραφέα Karel Čapek στο έργο του "R.U.R." (Rossum's Universal Robots), όπου σατιρίζει την εξάρτηση της κοινωνίας από τους μηχανικούς εργάτες (ρομπότ) της τεχνολογικής

εξέλιξης και που τελικά εξοντώνουν τους δημιουργούς τους. Σε πολλές σύγχρονες σλαβικές γλώσσες (πχ την πολωνική) χρησιμοποιείται σαν έκφραση της καθημερινότητας με την έννοια της σκληρής δουλειάς (αντίστοιχο του χαμαλίκι).

Πράγματι ,η φιλοδοξία του ανθρώπου να δημιουργήσει μηχανές που θα του μοιάζουν τόσο στη μορφή όσο και τη λειτουργία, πρωτοσυναντάται στην ελληνική μυθολογία. Ο Προμηθέας έπλασε την ανθρωπότητα από πηλό. Επιπλέον, από τα πρώτα ρομπότ που αναφέρονται στη λογοτεχνία είναι ο Τάλως, ο μυθικός χάλκινος γίγαντας που προστάτευε την Κρήτη από τους εισβολείς, αποτελώντας το πρώτο <<αυτόματο>> στην ανθρώπινη ιστορία καθώς και οι 20 τρίποδες λέβητες του Ηφαίστου θεωρούμενοι "*θαύμα ιδέσθαι*".

Με την ανάπτυξη και μελέτη των ρομπότ ασχολείται η http://el.wikipedia.org/w/index.php?title=%CE%A1%CE%BF%CE%BC%CF%80%CE%BF%CF%84%CE%B9%CE%BA%CE%AE&action=edit&redlink=1**ρομποτική**, επιστήμη που αποτελεί συνδυασμό πολλών κλάδων άλλων επιστημών, κυρίως δε της πληροφορικής, της ηλεκτρονικής και της μηχανολογίας.

Στην επιστημονική φαντασία συνήθως συναντούνται ρομπότ τα οποία έχουν τη μορφή ανθρώπου. Αυτά τα ρομπότ καλούνται ανδροειδή. Τα σημερινά ρομπότ δεν είναι ανδροειδή (androids) που κατασκευάστηκαν για να υποδυθούν ανθρώπινα όντα.

Λίγα χρόνια αργότερα κατά τη δεκαετία του '40,ο ρώσος συγγραφέας επιστημονικής φαντασίας Isaac Asimov συνέλαβε το robot ως ένα <<αυτόματο>> ,με εμφάνιση ανθρώπου, αλλά απαλλαγμένο από συναισθήματα. Η συμπεριφορά του υπαγορευόταν από ένα <<ποζιτρονικό μυαλό>> προγραμματισμένο από έναν 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 αφιερωμένο στη μελέτη των ρομπότ και διέπονται από τους τρείς παρακάτω βασικούς νόμους,(βλέπε[4]).

1.Ένα ρομπότ δεν μπορεί να τραυματίσει ή μέσω της αδράνειάς του να βλάψει ένα ανθρώπινο πλάσμα.

2. Ένα ρομπότ πρέπει να υπακούει σε εντολές που δίνονται από άνθρωπο, εκτός και αν αυτό έρχεται σε αντίθεση με τον πρώτο νόμο.

3. Ένα ρομπότ πρέπει να προστατεύει την ίδια του την ύπαρξη, εκτός και αν αυτό έρχεται σε αντίθεση με τον πρώτο ή δεύτερο νόμο.

Είναι σημαντική η ανάπτυξη ρομπότ που να έχουν τα αναγκαία χαρακτηριστικά ώστε να είναι φιλικά και ωφέλιμα προς τον άνθρωπο. Τα στοιχεία αυτά ονομάζονται στοιχεία κοινωνικής νοημοσύνης.

Ρομπότ: Η εξέλιξη (;) των μηχανών

Μπορεί το τελειότερο ον που γνώρισε πότε η ανθρωπότητα να αποτελεί προϊόν της φαντασίας του **Ισαάκ Ασίμοφ**, αλλά αυτό δεν σημαίνει ότι η ρομποτική επιστήμη (**robotics**) των τελευταίων χρόνων δεν έχει σημειώσει μεγάλη πρόοδο. Σύμφωνα με εκτιμήσεις επιστημόνων που ασχολούνται με την ρομποτική και γενικότερα με την Τεχνητή Νοημοσύνη, το 2050 ο γρήγορος ρυθμός ανάπτυξης της πληροφορικής **θα οδηγήσει στη δημιουργία ρομπότ τα οποία θα είναι πιο έξυπνα από τον άνθρωπο.** Ανησυχήσατε; Η αλήθεια είναι μία τέτοια εξέλιξη όντως θα προκαλούσε θύελλα συζητήσεων με πολλά υπέρ και κατά. Τα σημερινά ρομπότ έχουν τις υπολογιστικές ικανότητες του εγκεφάλου ενός εντόμου. Με βάση το γεγονός ότι η ισχύς των υπολογιστών διπλασιάζεται κάθε δεκαοκτώ μήνες περίπου, τα ρομπότ θα προσεγγίσουν τη νοημοσύνη των ζώων και, στη συνέχεια, αυτήν του ανθρώπου πολύ γρήγορα. Η ανεξέλεγκτη δύναμη των μηχανών έχει απασχολήσει τον άνθρωπο από πολύ παλιά και συνεχίζει να τον απασχολεί μέχρι σήμερα. Πολυάριθμες ταινίες και βιβλία αναφέρονται σε σενάρια καταστροφής του ανθρωπίνου γένους από υπερεξελιγμένες μηχανές και συστήματα Τεχνητής Νοημοσύνης.

Ας είμαστε όμως ρεαλιστές. Σκοπός της ρομποτικής είναι να βοηθήσει τον άνθρωπο και μέχρι στιγμής δείχνει να τα καταφέρνει μια χαρά. Ουσιαστική εξέλιξη στον συγκεκριμένο κλάδο έχουμε από την δεκαετία του '90 και μετά, όπου εμφανίζονται αρκετά ανθρωπόμορφα ή και ζωόμορφα ρομπότ. Αρχές του 1990 η **NASA** απέτυχε στην χρήση του Dante, ενός ρομπότ με οχτώ πόδια που είχε σαν αποστολή την συλλογή αερίων και μάγματος στην Ανταρκτική. Μία βλάβη στα καλώδια δεν επέτρεψε στο ρομπότ να εισέλθει στο ενεργό ηφαίστειο. Ένα χρόνο μετά ο Dante 2 εισήλθε με επιτυχία στο ηφαίστειο φέροντας εις πέρας την αποστολή του. Το 1996, το βιομετρικό ρομπότ... Robotuna δημιουργείται από τον φοιτητή David Barrett στο Massachusetts Institute of Technology, με σκοπό να μελετήσει πως κολυμπούν τα ψάρια στο νερό. Το πρώτο ανθρωπόμορφο ρομπότ, το P2 αποτέλεσε τον θεμέλιο λίθο των ερευνών της εταιρείας **Honda** πάνω στην δημιουργία ανδροειδών (ανθρωπόμορφα ρομπότ που θα μπορούν να μιμούνται τον άνθρωπο σε εμφάνιση και συμπεριφορά). Είχε ύψος 1.80 μέτρα και οι κινήσεις του ήταν πολύ κοντά στις ανθρώπινες. Το 1997 συμβαίνει κάτι μοναδικό για τα τότε δεδομένα. Το σκακιστικό πρόγραμμα Deep Blue της εταιρείας λογισμικού IBM, κερδίζει τον παγκόσμιο πρωταθλητής σκακιού Gary Kasparov, στο

προχωρημένο επίπεδο. Πολλοί εκφράζουν τότε ανησυχίες για την ραγδαία αύξηση της AI (Artificial Intelligence) και τις επιπτώσεις που μπορεί να έχει στον άνθρωπο. Ο σούπερ αυτός υπολογιστής είχε την ικανότητα να επεξεργάζεται 200000000 κινήσεις το δευτερόλεπτο! Το παιχνίδι μεταδόθηκε ζωντανά **μέσω του Διαδικτύου** και το παρακολούθησαν πάνω από 74 εκατομμύρια άτομα. Συνεχίζοντας την έρευνά της στην ρομποτική, η Honda παρουσιάζει το 1998 το P3, ενώ ένα χρόνο μετά η Sony «μπαίνει» δυναμικά στο παιχνίδι της ρομποτικής με τον AIBO (φώτο), έναν ρομποτικό σκύλο ικανό να αλληλεπιδρά με τον άνθρωπο σαν κανονικό κατοικίδιο. Τα πρώτα μοντέλα που κυκλοφόρησαν έγιναν ανάρπαστα σε 20 μόλις λεπτά–που αλλού; :στηνΙαπωνία.

Το 2000 αποτελεί σταθμό στην επιστήμη της ρομποτικής. Η διαρκώς δραστήρια Honda, παρουσιάζει την πρώτη έκδοση του ASIMO (Advanced Step in Innovative MObility), ενός ρομπότ που είναι σε θέση να τρέχει, να περπατά, να επικοινωνεί με τους ανθρώπους, να αναγνωρίζει εκφράσεις και περιβάλλοντα και να αλληλεπιδρά με αυτά. Το ύψος του είναι 1.30 μέτρα ενώ ζυγίζει 54. Η όψη του θυμίζει έναν μίνι αστρονάυτη και αποτελεί εξέλιξη παλαιότερων ρομποτικών μοντέλων της ίδιας εταιρείας. Την ίδια χρονιά η Sony παρουσιάζει και αυτή με την σειρά της ανθρωπόμορφα ρομπότ με το όνομα **Sony Dream Robots**, Σύμφωνα με τα Ηνωμένα Έθνη, τον Οκτώβριο του 2000 υπήρχαν 742.500 βιομηχανικά ρομπότ στον κόσμο, με περισσότερο από τα μισά αυτών να χρησιμοποιούνται μόνο στην Ιαπωνία. Έκτοτε, οι εξελίξεις στον χώρο των ανθρωπόμορφων ρομπότ παραμένουν στάσιμες. Με εξαίρεση την διαρκή κινητικότητα της Honda, με τα νέα μοντέλα του Asimo (πραγματικά εντυπωσιακό στον τρόπο κίνησής του - δείτε το σχετικό βίντεο εδώ), και το καινούργιο δημιούργημα της NASA, τον **Robonaut** (φωτό), ένα τηλεκατευθυνόμενο ρομπότ-αστρονάυτη που θα βοηθήσει αρκετά στην συλλογή στοιχείων από πλανήτες. Προφανώς και θα πρέπει να περάσουν αρκετά χρόνια ακόμα για να φτάσουμε έστω κοντά στην εικόνα που έχουμε σχηματίσει για τα ρομπότ μέσα από τις ταινίες και τα βιβλία επιστημονικής φαντασίας. Μπορεί και κάτι τέτοιο να μην γίνει ποτέ. **Οι ραγδιαίες εξελίξεις όμως στον τομέα της τεχνολογίας**, είναι αδύνατον να μην επηρρεάσουν και την ρομποτική, η οποία πιθανότατα θα στραφεί περισσότερο στην μελέτη για την δημιουργία μηχανών που θα βοηθήσουν τον άνθρωπο στις βασικές του ανάγκες. Όσοι λοιπόν περιμένουν μία αλά "**Terminator**" εξέγερση των μηχανών, μάλλον θα χρειαστεί να περιμένουν για πολύ ακόμα...

Εφαρμογές ρομπότ σε Συστήματα Κατεργασιών

Τύποι
ρομπότ
Προγρ/σμος
Τυπικές
εφαρμογές
Ευθεία - Ανάστροφη
κινηματική

ΓΕΝΙΚΑ

Βιομηχανικά ρομπότ είναι

αναπρογραμματιζόμενοι και πολυ-λειτουργικοί

διαχειριστές (manipulators) υλικών, τεμαχίων και εργαλείων

που εκτελούν μεταβλητές προγραμματισμένες κινήσεις

για την εκτέλεση ποικιλίας εργασιών.

Αποτελούνται από :

'βραχίονα' συνήθως με 3 βαθμούς ελευθερίας (κύριοι άξονες)

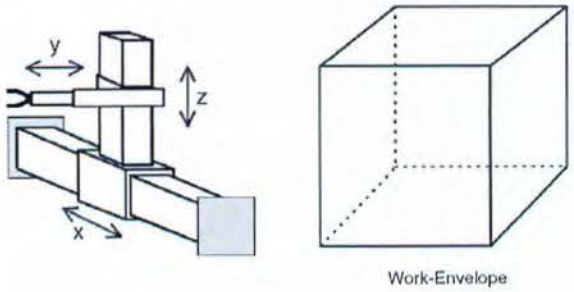
'χέρι' με προσαρμοσμένο εργαλείο (end-effector) με 1-3 ΒΕ

μονάδα τροφοδοσίας

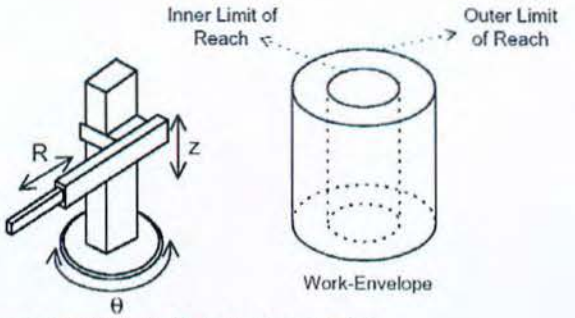
μονάδα ελέγχου

Κύριοι τύποι ρομπότ – κινηματική κατάταξη

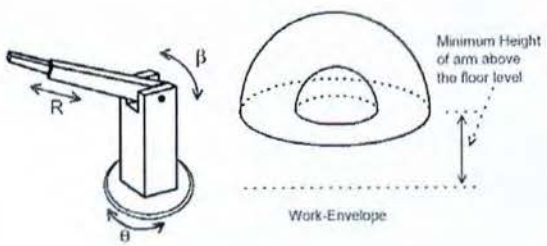
Όγκος εργασίας καρτεσιανού ρομπότ



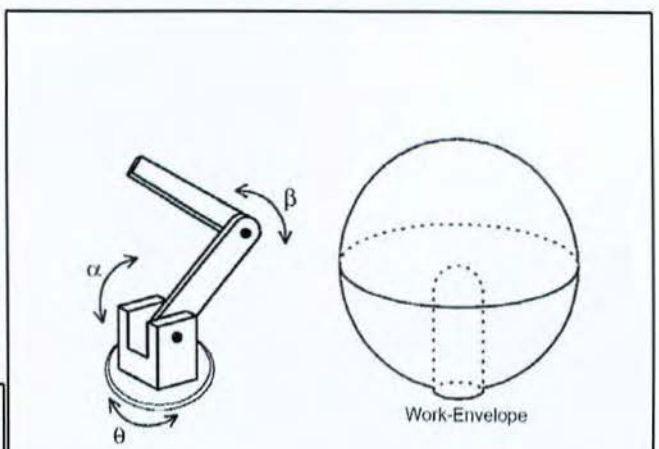
Όγκος εργασίας κυλινδρικού ρομπότ



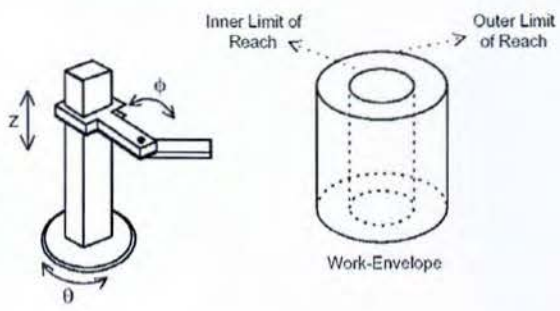
Όγκος εργασίας πολικού ρομπότ



Όγκος εργασίας αρθρωτού ρομπότ



Όγκος εργασίας ρομπότ SCARA



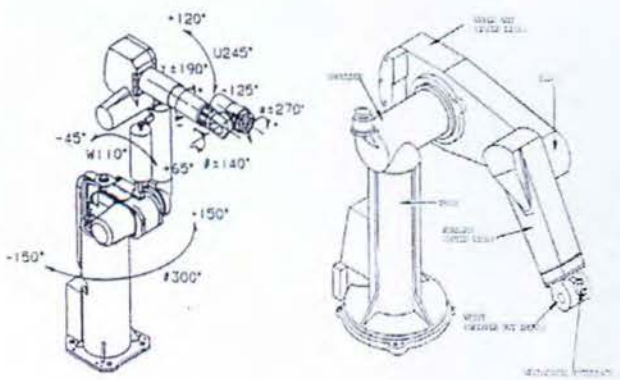
Σύστημα τροφοδοσίας ισχύος

- Πνευματικό
  - Μηχανικά stops
  - Electrolux
- Ηλεκτρικό
  - κινητήρες dc
  - μικρής και μέσης ισχύος ρομπότ
  - PUMA
- Υδραυλικό
  - μεγάλης ισχύος
  - Cincinnati T3

Συστήματα ελέγχου

- Σημείο προς σημείο (point-to-point)
  - κάθε άξονας κινείται ανεξάρτητα μέχρι το τελικό σημείο
  - τυχαία διαδρομή
- Συνεχούς διαδρομής (continuous path)
  - ταυτόχρονος έλεγχος όλων των αξόνων
  - με παρεμβολή σημείων
  - ακριβής διαδρομή

PUMA 562



Βασικές προδιαγραφές παλιότερου design

- ισχύς 22 KVA
- μέγιστο βάρος διαχείρισης (payload) 136 kg
- μέγιστο ύψος 3962 mm
- ακρίβεια 1.25 mm
- πόσο κοντά στο ζητούμενο σημείο πλησιάζει το άκρο του χεριού
- επαναληψιμότητα <1.25 mm
- πόσο κοντά μεταξύ τους είναι τα σημεία όπου φθάνει το άκρο του χεριού, εκτελώντας την ίδια εντολή κατ΄επανάληψη.

Cincinnati T3

Βασικές προδιαγραφές νεώτερου design

BE : 6

Ονομαστικό φορτίο : 20 kg

Μέγιστο φορτίο : 40 kg

Μέγιστη απόσταση : 2135 mm

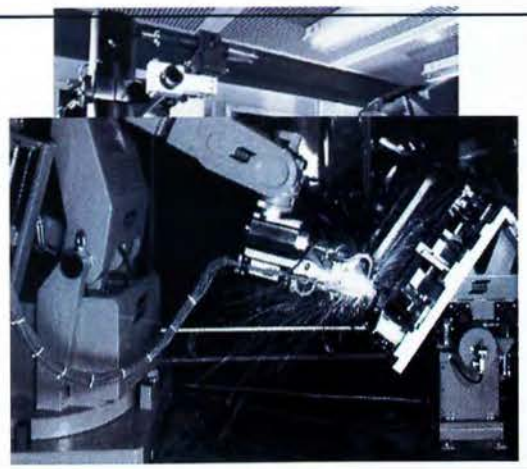
Επαναληψιμότητα : 50 μm

Εφαρμογές

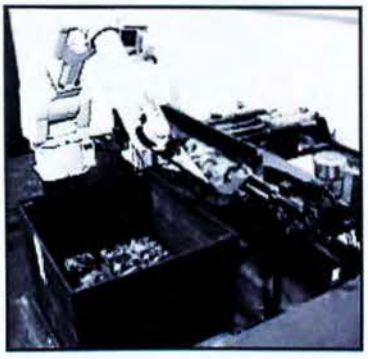
- Διαγραφή τροχιάς χαμηλής ακρίβειας
  - Βαφή με σπρέι
  - Σημειακή συγκόλληση
  - Καθαρισμός με δέσμη νερού
- Μετακίνηση (από σημείο σε σημείο) σχετικά χαμηλής ακρίβειας
  - Ανάληψη - τοποθέτηση μηχανών injection moulding και χυτοπρεσσών
  - Εξυπηρέτηση πρεσσών
  - Εξυπηρέτηση φούρνων
- Διαγραφή τροχιάς μέσης ακρίβειας
  - Ξεγρεζάρισμα
  - Γυάλισμα
  - Φλογοκοπή
- Εφαρμογές υψηλής ακρίβειας
  - Συναρμολόγηση
  - Συγκόλληση με ραφή πολύπλοκης γεωμετρίας

SCARA συναρμολόγηση

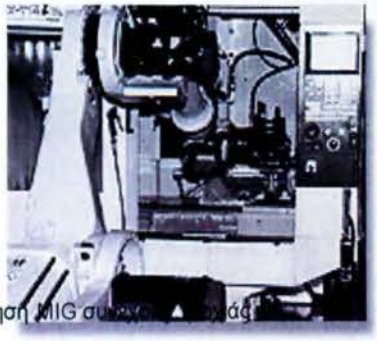
Ξεγρεζάρισμα – γυάλισμα χυτών



Παλλετάρισμα



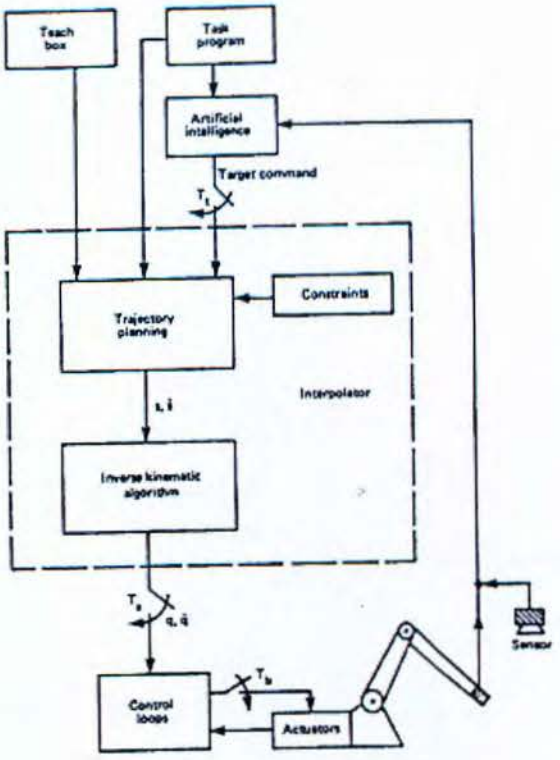
Εξυπηρέτηση τόρνου CNC



Συγκόλληση MIG σ[illegible]

Λειτουργία

- Πρόγραμμα
  - Προγραμματισμός με διδασκαλία
  - Προγραμματισμός εργασιών
  - Τεχνητή νοημοσύνη
- Παρεμβολή
  - Σχεδιασμός τροχιάς
  - Ανάστροφος κινηματικός μετασχηματισμός
- Έλεγχος
  - Βρόχοι ελέγχου (servo)
  - Επενεργητές
- Αιισθητήρια

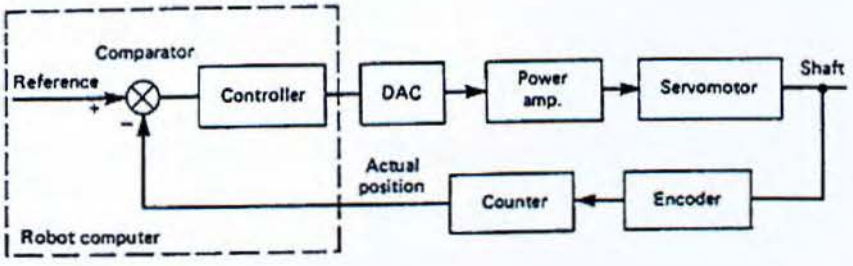


Λειτουργία - έλεγχος ρομπότ

- Servo :
  - υπολογισμός των κατάλληλων σημάτων για την κίνηση κάθε άξονα
- Ευθεία κινηματική ανάλυση :
  - από θέση και ταχύτητα των αξόνων υπολογίζεται η θέση του άκρου του ρομπότ
- Ανάστροφη κινηματική ανάλυση (όχι πάντα εφικτό):
  - από τη θέση του άκρου βρίσκεται η θέση κάθε άξονα
- Δυναμική ανάλυση
  - βάσει κινηματικής και φορτίων (αδράνεια, δύναμη Coriolis κλπ)
  - βελτιστοποιούντα ι οι εντολές κίνησης των αξόνων.
- Στοιχεία ευφυίας

- Αισθητήρια
  - δύναμη (load cell...)
  - απόσταση (με υπέρηχους..)
- Όραση
  - camera,
  - Ανάλυση / αναγνώριση εικόνων

Αρχή ελέγχου κλειστού βρόχου αξόνων ρομπότ



Τρόποι προγραμματισμού

- Με διδασκαλία (teach mode)
  - συσκευή για χειροκίνητη μετατόπιση των αξόνων
  - αποθήκευση στη μονάδα ελέγχου
  - Με γλώσσα προγραμματισμού on-line ή offline.
  - Υπάρχουν πάνω από 100 γλώσσες προγραμματισμού
- Κινηματικής προσομοίωση με γραφικά για προγραμματισμό αξόνων
  - Προγραμματισμός off-line

Τα δύο προβλήματα κινηματικής του ρομπότ

- Πρώτο - ευθύ
  - Από τις θέσεις των ΒΕ να προσδιορισθεί η θέση του άκρου
    - Το απλούστερο σχετικά πρόβλημα
    - Έχει μοναδική λύση
- Δεύτερο - ανάστροφο
  - Από την επιθυμητή θέση του άκρου να βρεθούν οι θέσεις των ΒΕ από τις οποίες αυτή προκύπτει
    - Πολυπλοκότερο πρόβλημα
    - Μπορεί να έχει περισσότερες από μια λύσεις ή και καμμία
- Η γενική προσέγγιση
  - Με βάση ομογενείς μετασχηματισμούς

Ένας **μηχανικός βραχίονας** είναι ένα ρομπότ, συνήθως προγραμματισμένο, με παρόμοιες λειτουργίες με ένα ανθρώπινο χέρι. Οι δεσμοί ενός τέτοιου βραχίονα συνδέονται μεταξύ τους με αρθρώσεις, που επιτρέπουν είτε περιστροφική κίνηση (όπως σε ένα αρθρωτό ρομπότ) ή μεταγραφική (γραμμική) μετατόπιση. Οι σύνδεσμοι του βραχίονα, μπορεί να θεωρηθεί, ότι αποτελούν μια κινηματική αλυσίδα. Το τέλος των επιχειρήσεων της κινηματικής αλυσίδας του βραχίονα ονομάζεται «τέλος επίδρασης» και είναι ανάλογη με το ανθρώπινο χέρι. Το τέλος επίδρασης μπορεί να είναι σχεδιασμένο για να εκτελεί οποιαδήποτε επιθυμητή εργασία, όπως, συγκόλληση, πιάσιμο, περιστροφή κλπ.,(βλέπε[3]) ανάλογα με την εφαρμογή. Για παράδειγμα τα όπλα ρομπότ στις αυτοκινητοβιομηχανικές γραμμές συναρμολόγησης εκτελούν μια ποικιλία εργασιών, όπως θερμοσυγκόλληση, μέρη περιστροφής και τοποθέτησης κατά τη συναρμολόγηση.

Στο διάστημα, το Διαστημικό Λεωφορείο Remote System Manipulator επίσης γνωστό ως Canadarm ή SSRMS και ο διάδοχός του Canadarm2, είναι παραδείγματα των πολλαπλών βαθμών ελευθερίας ρομποτικών βραχίονων, που έχουν χρησιμοποιηθεί για να εκτελέσουν μια ποικιλία εργασιών, όπως επιθεωρήσεις του Διαστημικού Λεωφορείου χρησιμοποιώντας ειδικά ανεπτυγμένη κάμερα έκρηξης και αισθητήρες που επισυνάπτονται στο τέλος της επίδρασης και της δορυφορικής εγκατάστασης με ελιγμούς ανάκτησης από τον κόλπο φορτίου του διαστημικού λεωφορείου.

Οι βραχίονες ρομπότ μπορεί να είναι αυτόνομοι ή ρυθμιζόμενοι και μπορεί να χρησιμοποιηθούν για την εκτέλεση πολλών εργασιών με μεγάλη ακρίβεια.

Ο ρομποτικός βραχίονας μπορεί να είναι σταθερού ή κινητού τροχού και μπορούν να σχεδιαστούν για τις βιομηχανικές εφαρμογές ή για το σπίτι.

Είδη

Καρτεσιανό ρομπότ ή **γραμμικό ρομπότ** : Βιομηχανικό ρομπότ του οποίου οι τρεις κύριοι άξονες ελέγχου είναι γραμμικοί (δηλαδή μετακινούνται σε ευθεία γραμμή και δεν γυρίζουν) και είναι σε γωνία 90 μοιρών ο ένας με τον άλλο. Μεταξύ άλλων προτερημάτων, αυτή η μηχανική διάταξη απλοποιεί τον έλεγχο ρομπότ και την επίλυση του βραχίονα. Τα ρομπότ καρτεσιανών συντεταγμένων με τον οριζόντιο άξονα στηριγμένο και στα δύο άκρα του μερικές φορές ονομάζονται και **ρομπότ γκάντρι** (gantry robots). Συνήθως είναι πολύ μεγάλα.

Μια συνηθισμένη εφαρμογή αυτού του τύπου ρομπότ είναι η μηχανή αριθμητικού ελέγχου με υπολογιστή (computer numerical control machine ή CNC machine). Η απλούστερη εφαρμογή χρησιμοποιείται στους μύλους και στις μηχανές επιλογής όπου ένας δείκτης μετακινείται σε ένα πεδίο χ-ψ ενώ ένα εργαλείο υψώνεται ή κατεβαίνει σε ένα επίπεδο για να ζωγραφίσει ένα ακριβές σχέδιο.

Κυλινδρικό ρομπότ: Χρησιμοποιείται για τη συναρμολόγηση, το χειρισμό σε εργαλειομηχανές συγκόλλησης τόπου, και το χειρισμό σε μηχανές χύτευσης. Πρόκειται για ένα ρομπότ του οποίου οι άξονες σχηματίζουν ένα κυλινδρικό σύστημα συντεταγμένων.

Σφαιρικό ρομπότ / Polar ρομπότ : Χρησιμοποιείται για το χειρισμό σε εργαλειομηχανές, συγκόλλησης τόπου, χύτευσης, μηχανές περικοπής, συγκόλλησης αερίου και συγκόλληση με τόξο. Πρόκειται για ένα ρομπότ του οποίου οι άξονες σχηματίζουν ένα πολικό σύστημα συντεταγμένων.

Αρκτικόλεξο SCARA σημαίνει επιλεκτική συνέλευση βραχίονα συμβατού ρομπότ ή επιλεκτικά συμβατού αρθρωτού ρομποτικού βραχίονα.

Το 1981, Sankyo Seiki, Pentel και η NEC παρουσίασε μια τελείως νέα αντίληψη για τα ρομπότ συναρμολόγησης. Το ρομπότ αναπτύχθηκε υπό την καθοδήγηση του Hiroshi Makino, καθηγητής στο Πανεπιστήμιο του Yamanashi. Το ρομπότ ονομάστηκε επιλεκτική Συνέλευση Συμμόρφωσης βραχίονα ρομπότ, SCARA. Ο βραχίονάς του ήταν άκαμπτος στο Z-άξονα και εύκαμπτος στους XY-άξονες, που του επέτρεπαν να προσαρμόζεται στις οπές των X-Y-αξόνων.

Λόγω των παράλληλων αξόνων κοινής διάταξης του SCARA, ο βραχίονας είναι ελαφρώς συμβατός προς την κατεύθυνση XY αλλά άκαμπτος στη «Z» κατεύθυνση, εξ ου και ο όρος: Επιλεκτική συμμόρφωση. Αυτό είναι ωφέλιμο για πολλούς τύπους εργασιών συναρμολόγησης, δηλαδή, εισάγοντας μια στρογγυλή καρφίτσα σε μια στρογγυλή τρύπα, χωρίς δεσμευτικό χαρακτήρα.

Το δεύτερο χαρακτηριστικό της SCARA είναι η συνένωση δύο συνδέσμων διάταξης βραχίονα παρόμοιο με τα ανθρώπινα χέρια μας, εξ ου και ο όρος που συχνά χρησιμοποιείται «Αρθρωτά». Αυτή η δυνατότητα επιτρέπει στον βραχίονα να επεκταθεί σε περιορισμένους χώρους και στη συνέχεια να ανακαλέσει ή να «διπλωθεί» έξω από το δρόμο. Αυτό είναι χρήσιμο για τη μεταφορά τμημάτων από το ένα κύτταρο στο άλλο ή για φόρτωση / εκφόρτωση σταθμών που είναι κλειστοί.

Το SCARA είναι γενικά πιο γρήγορο και καθαρότερο σε σύγκριση με το καρτεσιανό σύστημα. Ενιαίο βάθρο mount τους απαιτεί ένα μικρό αποτύπωμα και προσφέρει μια εύκολη, απρόσκοπτη μορφή στήριξης. Από την άλλη πλευρά,το SCARA μπορεί να είναι πιο ακριβό από το αντίστοιχα καρτεσιανό σύστημα και το λογισμικό ελέγχου απαιτώντας την αντίστροφη μηχανική για την γραμμική παρεμβολή κίνησης. Αυτό το λογισμικό έρχεται συνήθως με το SCARA αν και συνήθως είναι διαφανής για τον τελικό χρήστη.

Αρθρωτά ρομπότ: Χρησιμοποιειούνται για εργασίες συναρμολόγησης, χύτευσης, περικοπή μηχανών, συγκόλλησης αερίου, συγκόλληση με τόξο και χρωμάτων με ψεκασμό. Πρόκειται για ένα ρομπότ του οποίου το χέρι έχει τουλάχιστον τρεις περιστροφικές αρθρώσεις.

Παράλληλα ρομπότ: Μια χρήση του είναι μια κινητή πλατφόρμα με εξομοιωτές πτήσης και χειρισμό πιλοτηρίου. Πρόκειται για ένα ρομπότ του οποίου τα όπλα έχουν συντρέχουσες πολυγωνικές ή περιστροφικές αρθρώσεις.

Ανθρωπόμορφα ρομπότ: Μορφοποιημένα με έναν τρόπο που μοιάζουν με ανθρώπινο χέρι, δηλαδή με ανεξάρτητα δάχτυλα και τους αντίχειρες.

Κατά την πολυετή εξέλιξη της επιστήμης της ρομποτικής προέκυψαν διάφορα είδη ρομποτικών μηχανισμών, οι οποίοι διαφέρουν σημαντικά στη μορφή, αποτελούνται όμως από αντίστοιχα επιμέρους υποσυστήματα. Τα τελευταία είναι αυτά που αναφέραμε παραπάνω, δηλαδή το μηχανολογικό υποσύστημα, το υποσύστημα αίσθησης και το σύστημα ελέγχου.

Τα σπουδαιότερα είδη ρομπότ είναι τα παρακάτω:

Ρομπότ Σταθερής Βάσης: τα ρομπότ αυτά αποτελούνται από διαδοχικά στερεά σώματα (σύνδεσμοι) που συνδέονται μέσω αρθρώσεων σχηματίζοντας μία 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 το βραχίονα, τον καρπό και το εργαλείο.

Κινούμενα Ρομπότ: ως κινητά ρομπότ χαρακτηρίζονται όλα εκείνα τα ρομπότ που έχουν τη 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 το βαθμό αυτονομίας τους. Έτσι έχουμε:

- AGVs: τα AGVs (Automatic Guided Vehicles) έχουν περιορισμένη αυτονομία κίνησης, δεδομένου ότι η τροχιά τους είναι προκαθορισμένη μέσω καλωδίων στο έδαφος ή πομπών στον περιβάλλοντα χώρο (Σχήμα 2).

- Αυτόνομα Έντροχα Ρομπότ: τα ρομπότ αυτά λειτουργούν με αρκετά υψηλό βαθμό αυτονομίας. Πιο συγκεκριμένα μπορούν και λειτουργούν χωρίς συνεχή εξωτερική επίβλεψη και είναι ικανά να εκτελούν εργασίες αυτόνομα δεχόμενα μόνο ορισμένες υψηλού επιπέδου εντολές (Σχήμα 3).

• Βαδίζοντα Ρομπότ: τα ρομπότ αυτά χρησιμοποιούν μηχανικά πόδια για την κίνησή τους και όχι συμβατικούς τροχούς όπως στις προηγούμενες δύο κατηγορίες. Τα κυριότερα πλεονεκτήματα της συγκεκριμένης υλοποίησης είναι η μεγάλη δυνατότητα αποφυγής εμποδίων και η ικανότητα αναρρίχησης σε ανώμαλα εδάφη και μη επίπεδες επιφάνειες. Από τα πιο συνηθισμένα ρομπότ αυτής της κατηγορίας είναι τα δίποδα ενώ δεν αποκλείονται και εφαρμογές με περισσότερα από δύο πόδια, π.χ. ρομπότ που μοιάζουν και κινούνται όπως οι αράχνες (Σχήμα 4).

• ROVs: τα ROVs (Remotely Operated Vehicles) ανήκουν στην κατηγορία των μη επανδρωμένων υποβρύχιων ρομπότ. Όπως υποδηλώνει το όνομά τους δεν έχουν μεγάλο βαθμό αυτονομίας, μιας και είναι συνδεδεμένα με το μητρικό πλοίο μέσω καλωδίου, το οποίο και καλύπτει τις ανάγκες του ρομπότ σε ενέργεια και επικοινωνιές. Τα ρομπότ αυτού του τύπου έχουν σχήμα κοτιού και κινούνται γενικά σε χαμηλές ταχύτητες (σχήμα 5).

• Εναέρια ρομπότ: πρόκειται για μη επανδρωμένα ιπτάμενα ρομπότ, όπως ελικόπτερα και αεροπλάνα. Τα ρομπότ αυτά έχουν διαρκώς αυξανόμενες εφαρμογές, όμως εξαιτίας της μειωμένης ακόμα σταθερότητας και ασφάλειας στη συμπεριφορά τους χρησιμοποιούνται για στρατιωτικούς κυρίως σκοπούς (Σχήματα 7 και 8).

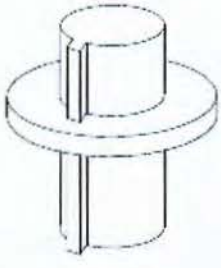
Όλες οι παραπάνω κατηγορίες ρομπότ αποτελούνται από τα βασικά υποσυστήματα που έχουμε ήδη αναφέρει. Τα υποσυστήματα αυτά θα περιγραφούν με μεγαλύτερη λεπτομέρεια στις παραγράφους που θα ακολουθήσουν δίνοντας όμως έμφαση κυρίως στην πρώτη κατηγορία ρομπότ. Ο λόγος είναι ότι από όλα τα είδη ρομπότ, αυτό που σήμερα έχει φτάσει σε ένα επίπεδο ώριμης τεχνολογίας είναι οι βιομηχανικοί ρομποτικοί βραχίονες. Αυτό βέβαια δεν σημαίνει ότι τα επόμενα χρόνια δεν θα υπάρξουν σημαντικές τεχνολογικές εξελίξεις και στις υπόλοιπες κατηγορίες ρομπότ.

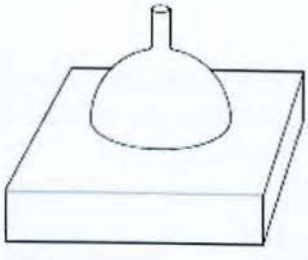
Βιομηχανικοί Ρομποτικοί Βραχίονες: Βασικές Έννοιες και Είδη

Όπως έχουμε ήδη σημειώσει ένας ρομποτικός βραχίονας αποτελείται από μία σειρά

διαδοχικών στερεών σωμάτων που ονομάζονται σύνδεσμοι. Οι σύνδεσμοι συνδέονται ανά δύο μεταξύ τους μέσω αρθρώσεων σχηματίζοντας μία κινηματική αλυσίδα. Οι αρθρώσεις μπορεί να είναι :

περιστροφικές : υλοποιούν σχετική περιστροφική κίνηση μεταξύ δύο διαδοχικών συνδέσμων
πρισματικές : σχετική μεταφορική κίνηση μεταξύ δύο διαδοχικών συνδέσμων,
σφαιρικές : υλοποιούν σφαιρική περιστροφική κίνηση μεταξύ δύο διαδοχικών συνδέσμων και παρέχουν στην κατασκευή από έναν βαθμό κινητικότητας (Σχήματα 9 - 11). Με τη σειρά της, μία κινηματική αλυσίδα χαρακτηρίζεται ως ανοικτή όταν υπάρχει μία μόνο διαδοχή συνδέσμων που να συνδέει τα δύο άκρα του βραχίονα και κλειστή όταν οι σύνδεσμοι που τη συνιστούν σχηματίζουν βρόχο.





Βαθμοί Κινητικότητας και Βαθμοί Ελευθερίας

Κρίνεται σκόπιμο να επισημάνουμε τη διαφορά που υπάρχει ανάμεσα στους βαθμούς κινητικότητας ενός βραχίονα και τους βαθμούς ελευθερίας που απαιτούνται για την εκτέλεση ενός έργου. Για ένα βραχίονα το πλήθος των βαθμών κινητικότητας είναι σταθερό και ίσο με το πλήθος των αρθρώσεών του (πρισματικών ή/και περιστροφικών). Από την άλλη πλευρά οι βαθμοί ελευθερίας είναι άμεσα συνδεδεμένοι με το συγκεκριμένο έργο που καλείται να φέρει εις πέρας ο βραχίονας. Για τη γενική περίπτωση που θέλουμε να τοποθετήσουμε και να προσανατολίσουμε ένα αντικείμενο στον τρισδιάστατο χώρο απαιτούνται 6 βαθμοί ελευθερίας (3 για να τοποθετήσουμε ένα σημείο του αντικειμένου στο χώρο και 3 για να προσανατολίσουμε το αντικείμενο ως προς ένα σύστημα συντεταγμένων αναφοράς). Είναι προφανές ότι ένας ρομποτικός βραχίονας με 6 βαθμούς κινητικότητας μπορεί να αντεπεξέλθει σ'αυτό το έργο, όπως επίσης και σε οποιοδήποτε άλλο έργο που απαιτεί μέχρι 6 βαθμούς ελευθερίας.

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 στη συνέχεια.

Ωφέλιμο Φορτίο – Επαναληψιμότητα – Ακρίβεια

Από τα πιο σημαντικά μεγέθη ενός βιομηχανικού βραχίονα είναι το ωφέλιμο φορτίο, η επαναληψιμότητα και η ακρίβεια. Πιο 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.
    - Επαναληψιμότητα: εκφράζει τη δυνατότητα του βραχίονα να γυρίσει στο ίδιο σημείο μετά από αρκετές επαναλήψεις και δίνεται ως εύρος μέσα στο οποίο ο βραχίονας θα τερματίσει την κίνηση. Η απόκλιση οφείλεται στο ότι κατά τη λειτουργία του το ρομπότ είναι δυνατό να χάσει λίγο από τη μέτρηση της θέσης με αποτέλεσμα να μη μπορεί να επιστρέψει στη συγκεκριμένη θέση μετά από ορισμένους κύκλους λειτουργίας. Δεδομένου ότι στις συνήθεις βιομηχανικές εφαρμογές οι επιθυμητές κινήσεις διδάσκονται στο ρομπότ αντιλαμβάνεται κανείς τη σπουδαιότητα της επαναληψιμότητας.

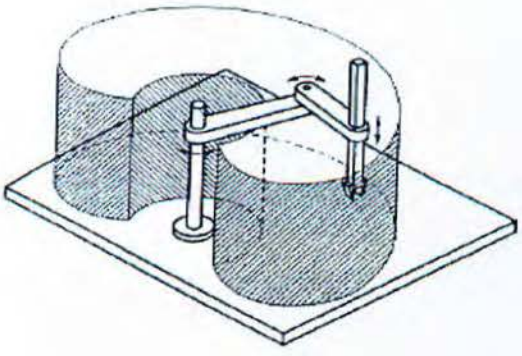
- Ακρίβεια: είναι η ικανότητα του ρομπότ να πηγαίνει ακριβώς στη θέση που του έχει δοθεί εντολή να πάει. Η ακρίβεια εξαρτάται κυρίως από τη διακριτότητα των εξαρτημάτων ελέγχου, τη μηχανολογική σύνδεση των μελών του και το ελάχιστο επιτρεπόμενο σφάλμα που επιβάλλει η ευστάθεια της λειτουργίας των σέρβο.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

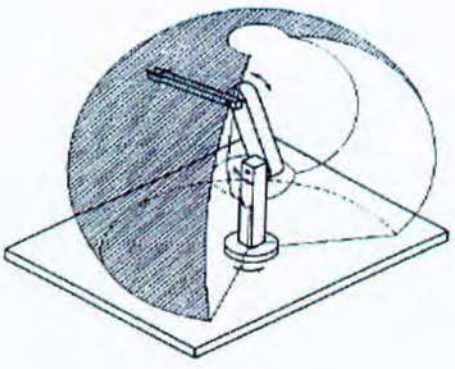
Ταξινόμηση Βραχιόνων βάσει της Γεωμετρικής Διαμόρφωσής τους

Ο τύπος και η διαδοχή των αρθρώσεων ενός βραχίονα επιτρέπει την ταξινόμησή των ρομπότ σε διάφορες 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 δύο κάθετοι μεταξύ τους (Σχήμα 12). Η καρτεσιανή δομή παρέχει μεγάλη δυσκαμψία και σταθερή ακρίβεια σε ολόκληρο το χώρο εργασίας που είναι ένα παραλληλεπίπεδο. Βασικό μειονέκτημα της κατασκευής είναι η μειωμένη επιδεξιότητα κίνησης, λόγω της πρισματικής φύσης των αρθρώσεων.

Βραχίονες Gantry: οι βραχίονες Gantry είναι στην ουσία καρτεσιανοί, διαφέρουν όμως από τους τελευταίους στον τρόπο προσέγγισης τους αντικειμένου ενδιαφέροντος (Σχήμα 13). Ειδικότερα ο βραχίονας Gantry προσεγγίζει το αντικείμενο από πάνω, τη στιγμή που ένας κλασικός καρτεσιανός βραχίονας προσεγγίζει το αντικείμενο από το πλάι. Άμεσες συνέπειες της διαφοροποίησης αυτής είναι η αύξηση του χώρου εργασίαςκαι της δυσκαψίας, καθώς επίσης και η δυνατότητα χειρισμού μεγάλων και βαριών αντικειμένων.

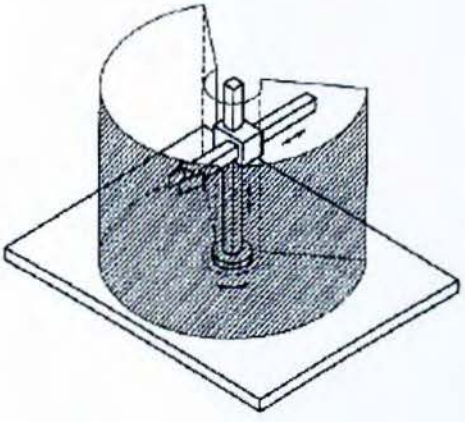


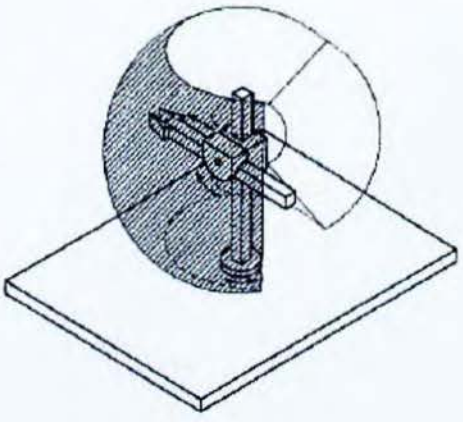


Κυλινδρικοί Βραχίονες: στους κυλινδρικούς βραχίονες η πρώτη πρισματική άρθρωση της καρτεσιανής δομής έχει αντικατασταθεί από μία περιστροφική άρθρωση (Σχήμα 14). Οι συγκεκριμένοι βραχίονες χαρακτηρίζονται από καλή δυσκαμψία, όμως η ακρίβεια της θέσης του καρπού μειώνεται καθώς η οριζόντια μετατόπιση αυξάνεται. Ο χώρος εργασίας στην περίπτωση αυτή είναι τμήμα κυλίνδρου. Σημαντικό μειονέκτημα της συγκεκριμένης γεωμετρίας είναι το ότι ο βραχίονας εισέρχεται στο χώρο εργασίας και τον περιορίζει.

Σφαιρικοί Βραχίονες: στους βραχίονες αυτούς αντικαθίσταται πλέον και η δεύτερη πρισματική άρθρωση της καρτεσιανής δομής με περιστροφική (Σχήμα 15). Η

μηχανολογική πολυπλοκότητα αυξάνει, ενώ η δυσκαμψία μειώνεται. Επιπλέον 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 τη δυνατότητα χειρισμού αντικειμένων που βρίσκονται στο έδαφος.





Βραχίονες SCARA: η γεωμετρία SCARA είναι ειδική και περιλαμβάνει δύο περιστροφικές και μία πρισματική άρθρωση τοποθετημένες κατά τέτοιο τρόπο ώστε οι άξονες κίνησης να είναι παράλληλοι μεταξύ τους (Σχήμα 16). Το όνομα SCARA προέρχεται από τα αρχικά των λέξεων Selective Compliance Assembly Robot Arm. Η συγκεκριμένη γεωμετρία παρέχει μεγάλη δυσκαμψία σε κατακόρυφη φόρτιση και ελαστικότητα σε οριζόντια. Η ακρίβεια τοποθέτησης του καρπού μειώνεται με την αύξηση της απόστασης του από τον άξονα της πρώτης άρθρωσης.

Ανθρωπομορφικοί Βραχίονες: η ανθρωπομορφική γεωμετρία υλοποιείται με τρεις διαδοχικές περιστροφικές αρθρώσεις. Ειδικότερα, ο άξονας περιστροφής της πρώτης άρθρωσης είναι κατακόρυφος και κάθετος στους άξονες περιστροφής των επομένων δύο αρθρώσεων, οι οποίοι είναι παράλληλοι μεταξύ τους (Σχήμα 17). Η συγκεκριμένη δομή παρέχει τη μεγαλύτερη επιδεξιότητα από όλες τις προηγούμενες, καθώς όλες οι αρθρώσεις είναι περιστροφικές. Ωστόσο η ακρίβεια του καρπού δεν είναι σταθερή εντός του χώρου εργασίας που έχει τη μορφή σφαίρας.

2.
Περιγραφή Λειτουργίας Ρομποτικού Βραχίονα

Κάθε ρομποτικός βραχίονας είναι προγραμματισμένος να κάνει μια πολύ συγκεκριμένη κίνηση ή πράξη ή πλήθος κινήσεων. Αυτές μπορεί να είναι μια μεταφορά μια κόλληση να βάψει ή να ψεκάσει μια επιφάνεια και λοιπά.

Στην κατασκευή μας κάνουμε μια επίδειξη βασιζόμενοι στην παραπάνω λογική ότι θα αναγνωρίσει ότι υπάρχει ένα αντικείμενο στη θέση Α και θα το μετακινήσει στη θέση Β ακολουθώντας ένα πλήθος ενδιαμέσων θέσεων τις οποίες έχει διδαχθεί από το προγραμματιστή και κάνοντας κινήσεις είτε σε έναν μόνο άξονα είτε σε περισσότερους ταυτοχρόνως.

Σαν θέση Α ορίζουμε το μέρος στο χώρο εργασίας όπου βρίσκεται το αντικείμενο που πρόκειται να σηκώσει ο βραχίονας για να το μεταφέρει εν τέλει στη θέση Β.

Ομοίως η θέση Β είναι η περιοχή μέσα στο χώρο εργασίας που θα καταλήξει το αντικείμενο. Όλα αυτά φυσικά γίνονται υπό κάποιες προϋποθέσεις.

Ο ρομποτικός μας βραχίονας έχει τέσσερις βαθμούς ελευθερίας καθώς διαθέτει τέσσερις σερβοκινητήρες. Όπως αναφέραμε και πιο πριν θα μπορούσαμε να χρησιμοποιήσουμε αντί για σερβοκινητήρες και άλλους μηχανισμούς για να μεταδώσουμε την κίνηση στο σταθερό μέρος του βραχίονα, όπως υδραυλικά συστήματα, πνευματικά, κινητήρες συνεχούς ρεύματος ή και εναλλασσόμενου χρησιμοποιώντας ιμάντες τεντωτήρες ρουλεμάν και άλλα μηχανολογικά στοιχεία. Τα συστήματα αυτά είναι πιο σύνθετα και αχρείαστα στη περίπτωσή μας καθώς εμείς αναπαριστούμε μια εξομοίωση και καθώς επίσης το κόστος τους είναι τεράστιο.

Οι βαθμοί ελευθερίας είναι οι εξής:

Degrees of Freedom (DOF)

Ένας που κάνει την αριστερά – δεξιά κίνηση του ώμου

Ένας που κάνει την πάνω- κάτω κίνηση του ώμου.

Ένας DOF που κάνει την πάνω κάτω κίνηση του αγκώνα.

Ένας που ανοίγει και κλείνει το τελικό στοιχείο δράσης, δηλαδή την αρπάγη.

Στο τρόπο μετάδοσης επίσης δεν χρησιμοποιούμε ρουλεμάν, ιμάντες αλυσίδες και τα λοιπά διότι οι κινητήρες είναι αρκετά δυνατοί να αντέξουν το βάρος μόνοι τους χωρίς καμία επιπλέον βοήθεια εκτός του κινητήρα που βρίσκεται στον αγκώνα ο οποίος έχει και ρουλεμάν για αντιστήριξη.

Ο ρομποτικός μας βραχίονας δίνει την δυνατότητα στο χρήστη να μπορεί να επιλέξει αν θέλει να κάνει την ίδια λειτουργία που είναι προγραμματισμένος να κάνει ή εάν θέλει να γίνεται ο έλεγχος χειροκίνητα μέσω του χρήστη από τα δυο joystick που διαθέτει.

Αυτή η εναλλαγή γίνεται με τη χρήση ενός διακόπτη που δίνει σήμα στο μΕ για το ποια λειτουργία να εκτελέσει μεταξύ της αυτόματης και της χειροκίνητης.

Συνεπώς όπως αναφέραμε και προηγουμένως για να μετακινηθεί το αντικείμενο από τη θέση Α στη θέση Β θα πρέπει ο διακόπτης να είναι γυρισμένος στο αυτόματο.

Μια άλλη προϋπόθεση είναι να μην υπάρχει αντικείμενο στη θέση Β. Αυτό γίνεται γνωστό με το αν δέχεται κάποιο σήμα ο μΕ από το button που υπάρχει στη θέση Β που μας υποδηλώνει σε τι κατάσταση βρίσκεται η εκεί πέρα θέση. Και τα δυο button πιέζονται από το βάρος του αντικειμένου.

Ο βραχίονας είναι προγραμματισμένος να μην κάνει απολύτως τίποτα αν δεν δεχθεί σήμα από την θέση Α. Δηλαδή αν έχει ήδη κάνει ένα κύκλο λειτουργίας θα παραμείνει στη μεσαία θέση (θα την ορίσουμε παρακάτω) εξετάζοντας τα στοιχεία που έχει σαν δεδομένα για το πώς θα πράξει, διαφορετικά εάν δεν έχει λειτουργήσει ούτε μια φορά θα βρίσκεται στην HARDHOME θέση έως ότου να πάρει σήμα από την θέση Α και εφόσον το αντικείμενο από τη θέση Β έχει απομακρυνθεί.

Από την στιγμή που δεν υπάρχει τίποτα στη θέση Β και έρθει σήμα από τη θέση Α ο βραχίονας θα ξεκινήσει είτε από την HARDHOME είτε από τη μεσαία θέση θα ανοίξει την αρπάγη και θα πάει πάρα πολύ γρήγορα σε μια θέση σχετικά κοντά στην Α άλλα όχι και τόσο κοντά. Οι μεταβαλλόμενες ταχύτητες στου ρομποτικούς βραχίονες μας δίνουν την ικανότητα να αυξήσουμε την παραγωγικότητα μειώνοντας το χρόνο της διεργασίας. Αφού πλησιάσει σχετικά κοντά στην Α μειώνει την ταχύτητα κάλλο ώστε να μην δημιουργηθεί καμιά ζημιά ή βλάβη στο αντικείμενο από την βίαιη συμπεριφορά του και φτάνει σε ένα σημείο πολύ κοντά. Τότε με πολύ μαλακές κινήσεις πλησιάζει το αντικείμενο και η αρπάγη το συγκρατεί.

Η αντίστροφη διαδικασία τώρα ακολουθείται. Με μαλακές και αργές κινήσεις ανυψώνει το αντικείμενο και κινώντας αρκετούς άξονες μαζί φτάνει πάλι στη μεσαία θέση. Ενώ ο

βραχίονας κρατάει το αντικείμενο δεν γίνεται καμία απότομη κίνηση σε αντίθεση με πριν που δεν κρατούσε τίποτε.

Κατά την διάρκεια των επαναλήψεων από την θέση Α στη Β ο βραχίονας δεν χρειάζεται να επιστρέφει στη HARDHOME διότι είναι χρονοβόρο και πρακτικά άχρηστο. Αντιθέτως ορίζουμε μια μεσαία θέση όπου είναι κάπου ενδιάμεσα μεταξύ των τερματικών θέσεων Α και Β που μας σώζει χρόνο και κάνει το προγραμματισμό πιο εύκολο.

Από την μεσαία θέση κινείται γραμμικά και ομαλά φτάνοντας κάπου πολύ κοντά στη θέση Β. Εκεί κινείται ακόμα πιο μαλακά ακουμπά το αντικείμενο στη θέση ανοίγει την αρπάγη και απομακρύνεται πολύ σιγά. Όταν έχει κάποια απόσταση από το αντικείμενο κινείται πιο γρήγορα και όταν φτάσει σε αρκετή απόσταση από αυτό τότε σπασμωδικά εκτοξεύεται στη μεσαία θέση περιμένοντας άλλη εντολή, αφού πρώτα απομακρυνθεί το αντικείμενο από τη θέση Β.

Η ίδια λειτουργία επαναλαμβάνεται όσο ό βραχίονας είναι ρυθμισμένος να παραμένει στην Αυτόματη λειτουργία.

Τέλος αξίζει να σημειωθεί ότι κανένα από τα joystick δεν παρεμβαίνει στην αυτόματη λειτουργία καθώς ο μΕ δεν λαμβάνει υπ' όψιν του όσα σήματα έρχονται από αυτά.

Όταν βρίσκεται το αντικείμενο στη θέση Β ένα buzzer ηχεί 3 φορές που προειδοποιεί τον ενδιαφερόμενο για την παραλαβή του.

Όταν ξεκινήσει για πρώτη φορά ο ρομποτικός βραχίονας κάνοντας κάποιες κινήσεις πηγαίνει και στέκεται σε μια από εμάς συγκεκριμένη θέση που την έχουμε ορίσει σαν την HARDHOME του. Γίνεται κάθε φορά που κάνουμε Reset στο μΕ και την χρησιμοποιούμε σαν μια θέση αρχικοποίησης για τυχών τροποποιήσεις που μπορεί να γίνουν στο μέλλον στο χώρο εργασίας για να έχουμε ένα πρώτο σημείο αναφοράς.

Εάν γυρίσουμε το διακόπτη στη θέση χειροκίνητο ο χρήστης έχει τη δυνατότητα να το ελέγχει από τα joystickκαι τα τυχών σήματα που μπορεί να έρχονται από τα buttons τα αγνοεί. Στο συγκεκριμένο βραχίονα για λόγους επίδειξης έχουμε μειώσει ή αυξήσει το εύρος λειτουργίας όσο αφορά τις μοίρες που κινούνται οι κινητήρες. Δηλαδή την δεξιά και αριστερά κίνηση την έχουμε περιορίσει και την πάνω κάτω κίνηση του ώμου την έχουμε περιορίσει έτσι ώστε ο χειριστής να μην μπορεί να χτυπήσει τον βραχίονα στο έδαφος. Αυτό θα μπορούσαμε να το είχαμε κάνει και με τερματοδιακόπτες που με την πίεση τους θα σταματούσε αυτόματα η περεταίρω κίνηση σε αυτή την κατεύθυνση. Όμως εδώ έγινε προγραμματιστικά αφού ο βραχίονας είναι ένα σύστημα κλειστού βρόγχου που μας επιστρέφει σαν feedback τις μοίρες που βρίσκεται και μάλιστα μπορούμε και διορθώνουμε και το σφάλμα. Οι δυο μεγάλοι κινητήρες χρησιμοποιούν και παραπάνω από μια μόνο ανάδραση καθώς χρησιμοποιούμε και μια μέτρηση από το εσωτερικό ποτενσιόμετρο τους.

Ο αγκώνας έχει πλήρες εύρος κινήσεων και η αρπάγη έχει μια ιδιόμορφη λειτουργία που μόλις συγκρατήσει το αντικείμενο το κρατάει αυτόματα εκεί και με ένα ακόμα σήμα από το ίδιο ποτενσιόμετρο μπορεί να το αφήσει.

Περισσότερες πληροφορίες για το πώς γίνονται όλα αυτά δίνονται πιο αναλυτικά στο κομμάτι του κώδικα.

3.

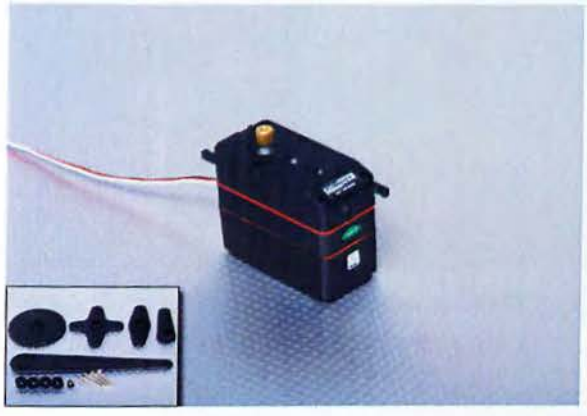
Ηλεκτρονική κατασκευή

Ο ρομποτικός βραχίονας αποτελείται από τέσσερα ηλεκτρονικά μέρη.

1) Από τους σερβοκινητήρες
2) Από το breadboard και την εξωτερική παροχή

3) Από την πλατφόρμα του μΕ Arduino Uno μαζί με το Shield του
4) Από την πλακέτα με τα joystick

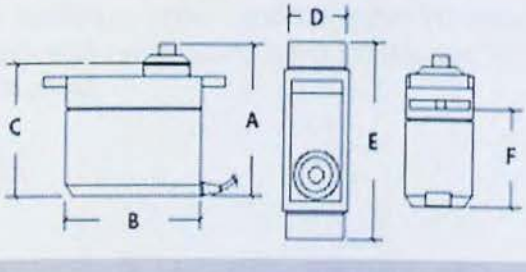
1) Οι δυο σερβοκινητήρες που μετακινούν τη βάση – ώμο είναι οι Turnigy S- 8166M Ακολουθούν τα χαρακτηριστικά τους:



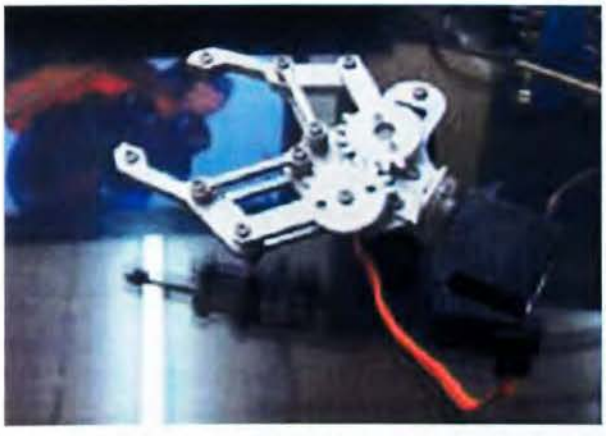
Turnigy S8166M Servo 154g / 33kg
Weight: 154g
Dimensions: 75 x 59 x 27mm
Torque: 33kg-cm
Speed: .21 deg/sec
Bearing: Twin
Gear Type: Metal

Product Config Table

Weight (g)	154
Torque (kg)	33
Speed (Sec/60deg)	0.21
A(mm)	68
B(mm)	61
C(mm)	62
D(mm)	30
E(mm)	75
F(mm)	50
	0
	0



Οι άλλοι δυο σερβοκινητήρες που χρησιμοποιούνται στο να μετακινούν την αρπάγη πάνω κάτω και στο να την ανοιγοκλείνουν.



Ακολουθούν μερικά από τα χαρακτηριστικά τους καθώς ο κατασκευαστής δεν διαθέτει όλες τις λεπτομέρειες για δικούς του λόγους.

It's powered by two servo motors with 2.3kg/cm of torque at 6V and 1.8kg/cm at 4.8V. The gripper opens to a maximum of 5 cm.

Όλοι οι σερβοκινητήρες λειτουργούν με εξωτερική παροχή στα +5V που την κατασκευάσαμε εμείς καθώς και όλα τα κυκλώματα έχουν μια κοινή γείωση.

Ο κάθε σερβοκινητήρας έχει και μία είσοδο όπου μπορούμε να στείλουμε δεδομένα και να ελέγχουμε την θέση και την ταχύτητά του.

Ειδικότερα για τους κινητήρες της βάσης χρειάστηκε να τους ανοίξουμε και να επέμβουμε στο ήδη υπάρχον κύκλωμα καθώς χρειαστήκαμε να έχουμε και μια μέτρηση από το ποτενσιόμετρο που έχουν μέσα τους, για λόγους που θα εξηγήσουμε στο κομμάτι του κώδικα.

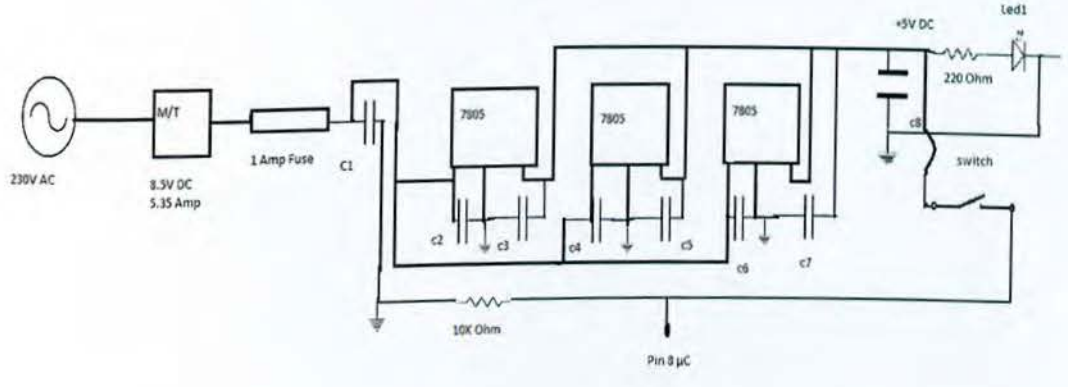
Η συνδεσμολογία έχει ως εξής:

Gripper ->pin3
Elbow -> pin5
Shoulderup&down ->pin6
Shoulderright&left ->pin9
Και το σήμα από τα εσωτερικά ποτενσιόμετρα
Των δυο κινητήρων συνδέθηκε στα A1 και A4 αντίστοιχα του μΕ.

2) Από τα 230V AC με ένα τροφοδοτικό υποβιβάζουμε και μετασχηματίζουμε την τάση στα 8.5V DC στα 5,65 Amps όπου περνάμε το ρεύμα μέσα από μια ασφάλεια από 1 έως 1.2 Amps (στηριζόμενη στην δικιά της ασφαλειοθήκη)και τερματίζει με καλώδιο 1,5mm^2 στην είσοδο της συστοιχίας των ολοκληρωμένων 7805 σταθεροποίηση της τάσης στα 5V. Οι γειώσεις είναι κοινές για όλο το κύκλωμα. Μεταξύ της εισόδου της αρχικής τάσης και της γείωσης καθώς και μεταξύ της γείωσης και της εξόδου σε κάθε ένα από τα ολοκληρωμένα 7805 βρίσκονται κεραμικοί πυκνωτές στοιχισμένοι παράλληλα. Συνεπώς στην έξοδο από τα ολοκληρωμένα και μεταξύ της γείωσης έχουμε διαφορά δυναμικού ακριβώς 5V. Φυσικά στην έξοδο έχουμε τοποθετήσει σειρά από ηλεκτρολυτικούς πυκνωτές για εξομάλυνση του ρεύματος, που δίνουν μια διαφορετική συμπεριφορά στο όλο σύστημα όταν το φορτίο είναι μέγιστο. Με μετρήσεις έχουμε βρει ότι το σύστημα μπορεί να χρειαστεί έως και 1Ampere στα 5V. Το οποίο ρεύμα είναι οριακά επικίνδυνο για την λειτουργία των κινητήρων. Χρησιμοποιώντας την ασφάλεια μπορούμε να γλιτώσουμε από δυσάρεστες συνέπειες.

Επιπρόσθετα πάνω στο breadboard είναι συνδεδεμένα μια αντίσταση 220Ω σε σειρά με ένα μπλε Led που μας φανερώνει ότι υπάρχει τάση στη πλακέτα καθώς επίσης είναι συνδεδεμένη στα 5 V και η μία απόληξη από ένα διακόπτη που καθορίζει στο σύστημα αν είναι σε αυτόματη ή χειροκίνητη λειτουργία. Η άλλη απόληξη του διακόπτη είναι συνδεδεμένη σε μια αντίσταση 10KΩ που καταλήγει στη γείωση. Εν μέσω του καλωδίου του διακόπτη υπάρχει άλλο ένα καλώδιο που καταλήγει στο pin8 του μΕ, δημιουργώντας έτσι έναν διαιρέτη τάσης για να έχουμε στην είσοδο του μΕ είτε λογικό «1» είτε λογικό «0». Τέλος ενώνουμε την γείωση του breadboard με Arduino και με του Shield κλείνοντας έτσι το κύκλωμα,(βλέπε[6]).

Ακολουθεί το κύκλωμα στο breadboard:

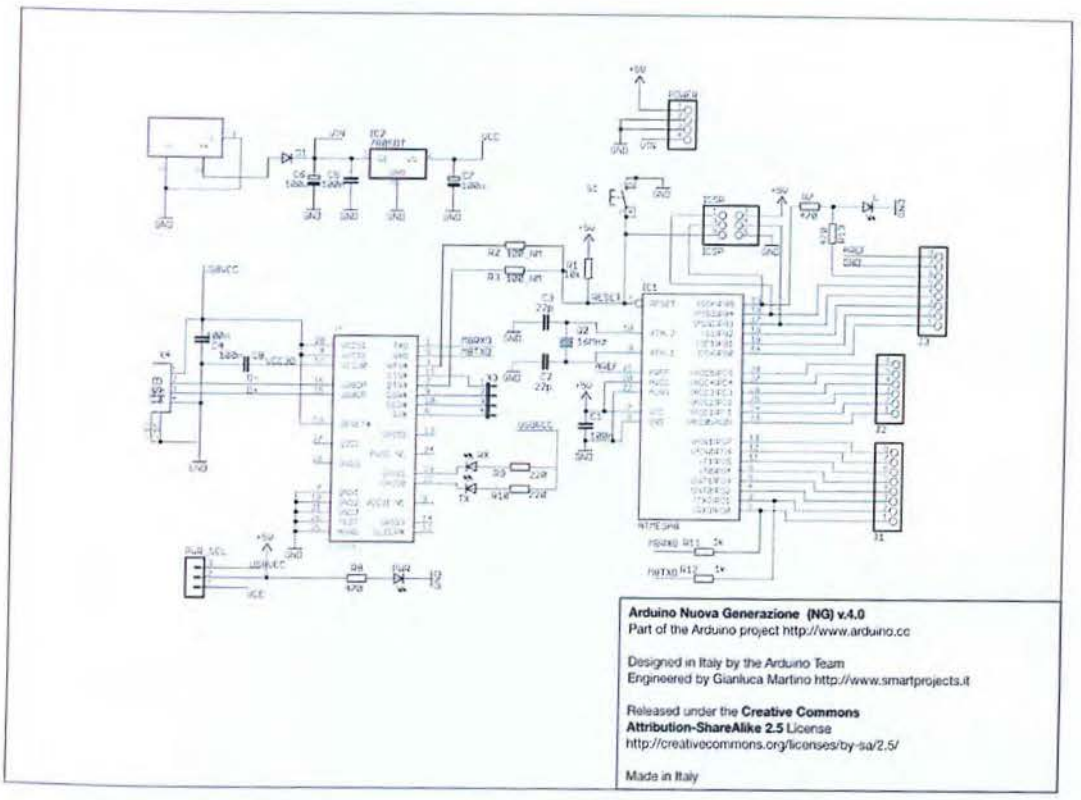


3) Arduino Uno and Shield

Χαρακτηριστικά:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328)
of which 0.5 KB used by bootloader	
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic:



Datasheet και πληροφορίες για την πλατφόρμα:

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance

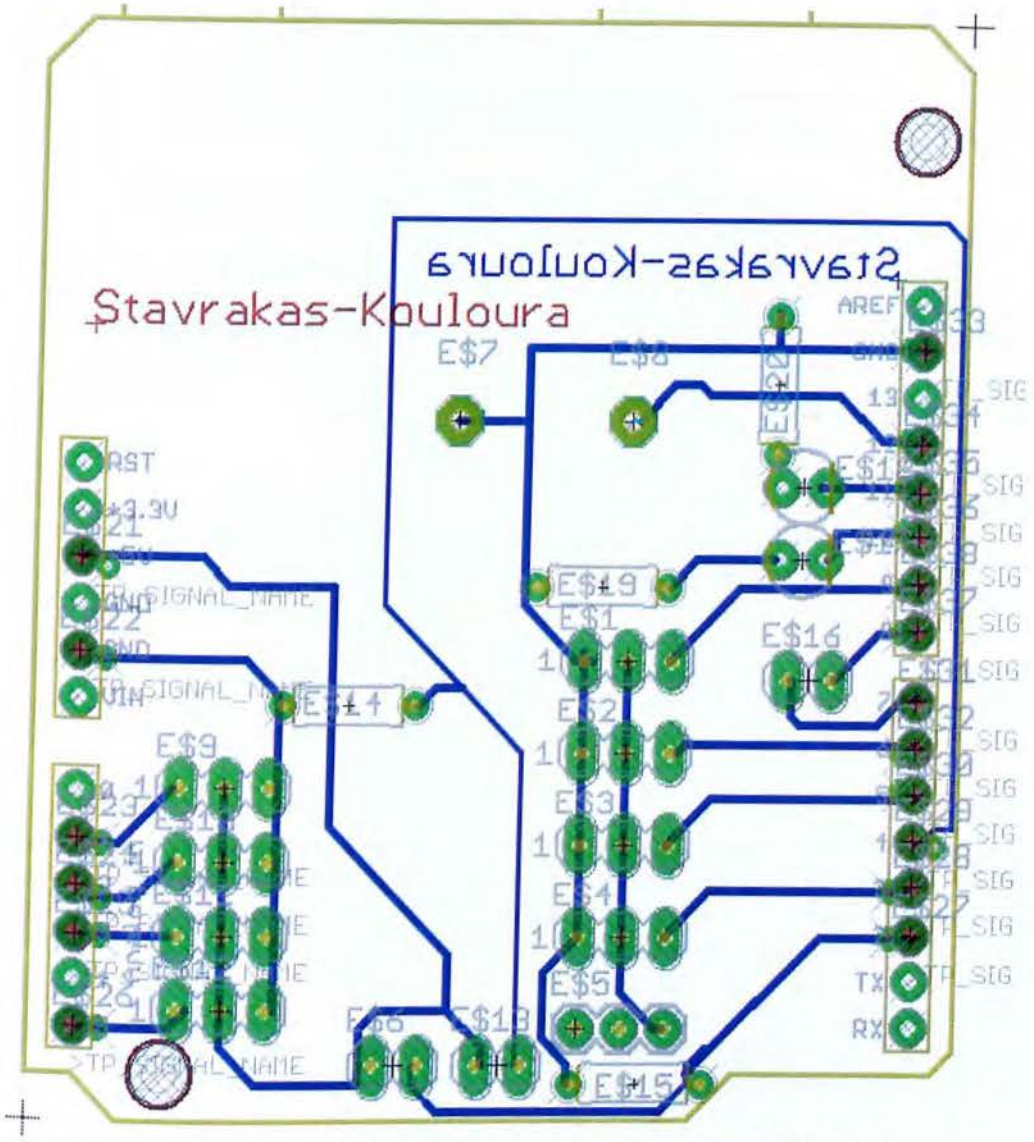
between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Shield:

Σαν Shield ορίζουμε την πλακέτα PCB η οποία έχει τις ίδιες διαστάσεις με αυτή του μΕ και κουμπώνει ακριβώς στις θηλυκές υποδοχές της πλακέτας δημιουργώντας ένα συμπαγές σύστημα. Η συγκεκριμένη είναι διπλής επιφανείας δηλαδή στο ένα επίπεδο βρίσκονται τα εξαρτήματα και κάποια pin για υποδοχές και στο άλλο ο χαλκός με τις ενώσεις και τα αρσενικά pins που κουμπώνουν στο ελεγκτή.

Η χρησιμότητα του είναι να προστατεύσει τις επαφές που καταλήγουν από τους κινητήρες στο μΕ καθώς όταν κινείται ο βραχίονας υπάρχει δυνατότητα να παρασυρθούν και είτε να κοπεί το σήμα είτε να γίνει κάποιο βραχυκύκλωμα.
Επίσης απλουστεύεται πολύ το κύκλωμα στο breadboard κάνοντας το πιο εύκολο στην διατήρηση και επισκευή του βραχίονα, σταθεροποιεί όλες τις επαφές και είναι και πιο εμφανίσιμο.

Το Shield δημιουργληθηκε με το πρόγραμμα Eagle.
Ακολουθεί το διάγραμμα:



Το Shield τροφοδοτείται από το breadboard δηλαδή έχει παντού τάση 5 V έως 1 Amp και είναι γειωμένο μεταξύ τους. Οι καλωδιολωρίδες από τους κινητήρες τερματίζονται πάνω στο shield όπου υπάρχει μια κάθετη σειρά που αντιστοιχεί στα 5V βραχυκυκλώνοντας κάθε pin και άλλη μια κάθετη σειρά που αντιστοιχεί στη γείωση και τέλος μια σειρά που αντιστοιχεί στα σήματα που αποστέλλονται από το μΕ.

Υπενθυμίζουμε ότι:

Gripper ->pin3
Elbow -> pin5
Shoulderup&down ->pin6
Shoulderright&left ->pin9
Και το σήμα από τα εσωτερικά ποτενσιόμετρα
Των δυο κινητήρων συνδέθηκε στα A1 και A4 αντίστοιχα του μΕ.

Στα pin 11 και 10 είναι συνδεδεμένα 2 led χρώματος κόκκινο και πράσινο αντίστοιχα με αντίσταση 220Ω το καθένα και τερματίζουν στη γείωση. Αυτά τα led τα χρησιμοποιούμε για να δηλώσουμε σε ποια θέση βρίσκεται ο διακόπτης, δηλαδή σε ποιο mode είναι. Έτσι χρησιμοποιούμε το 11 για το αυτόματο και το 10 για τη χειροκίνητη λειτουργία.

Στο pin 12 είναι συνδεδεμένο το buzzer και τερματίζει φυσικά στη γείωση
Στο pin 2 συνδέεται το button A το οποίο είναι αρνητικής λογικής και τροφοδοτείται με 5 V και δημιουργεί ένα διαιρέτη τάσης χρησιμοποιώντας μια αντίσταση 10ΚΩ που τερματίζει στη γείωση.
Στο pin 4 ομοίως είναι συνδεδεμένο ένα button B και καταλήγει και αυτό σε έναν διαιρέτη τάσης.
Το pin 7 είναι συνεχώς HIGH για πιθανών μελλοντική χρήση.
Το pin 8 έχει οριστεί σαν input και χρησιμοποιείται από το διακόπτη για εναλλαγή από manual σε automatic mode.

Το analog pin 0 και το analog pin 4 έχουν χρησιμοποιηθεί για να επεξεργαστούμε τις αναλογικές τιμές των ποτενσιομέτρων που βρίσκονται εντός των δυο μεγάλων κινητήρων και τις χρησιμοποιήσαμε για να κάνουμε μια πιο ομαλή εκκίνηση των κινητήρων διαβάζοντας τες και αντιστοιχώντας τες στις αντίστοιχες μοίρες.

Τα analog pin 1 ,2, 3 και 5 χρησιμοποιούνται από τα ποτενσιόμετρα που κινούν τους κινητήρες . Τα ποτενσιόμετρα αυτά καθώς και τα buttons λειτουργούν με +5V που παίρνουν όμως από την πλακέτα του Arduino.

Κατασκευή πλακέτας:
Όπως αναφέραμε η σχεδίαση έγινε στο πρόγραμμα Eagle. Έπειτα τυπώθηκε σε διάφανη μεμβράνη η οποία επικολλήθηκε πάνω στην πλακέτα. Στη συνέχεια η πλακέτα εκτίθηκε σε πολύ δυνατό φως και αποτυπώθηκε το σχεδιάγραμμα πάνω της. Η πλακέτα ξεπλένεται με καυστική σόδα και στη συνέχεια με τριχλωριούχο σίδηρο όπου απομακρύνεται η περιοχή που δεν έχει χαλκό. Αφαιρούμε τα χημικά χρησιμοποιώντας ισοπροπυλική αλκοόλη.
Επειδή είναι double layer κάνουμε με dremel τις απαραίτητες τρύπες για να ενώσουμε το ένα επίπεδο με το άλλο . Στο τέλος τοποθετούμε τα εξαρτήματα και τα αντίστοιχα pins και κάνουμε όλες τις κολλήσεις. Πριν τοποθετηθεί στη πλατφόρμα του Arduino γίνονται όλοι οι έλεγχοι για να διαπιστωθεί ότι λειτουργεί σωστά.

4) Πλακέτα με τα joystick

Ένα joystick δεν είναι τίποτα άλλο παρά μόνο δυο μεταβαλλόμενες αντιστάσεις με μηχανική συνδεσμολογία που η κίνηση του μοχλού μεταβάλλει την τιμή αυτών ταυτόχρονα. Χρησιμοποιώντας αυτή τη τεχνική είμαστε σε θέση να ελέγξουμε με ένα joystick 2 κινητήρες. Και τα δυο joystick έχουν την ίδια τάση στα 5 V από τον Arduino που καταλήγει στο ακριανό ποδαράκι από το κάθε ένα ποτενσιόμετρο δηλαδή σε τέσσερα συνολικά και τα υπόλοιπα ακριανά ποδαράκια καταλήγουν στην γείωση. Τα μεσαία χρησιμοποιούνται για να διαβάσει ο μΕ την αντίστοιχη τάση. Όταν μεταβάλλεται η αντίσταση κουνώντας το μοχλό αλλάζει και η διαφορά δυναμικού στο μεσαίο pin. Ο Arduino διαβάζει αυτή την τιμή με έναν 8 bit Analog to digital converter.

Μηχανολογική κατασκευή

Υπάρχει μια ξύλινη πλατφόρμα που αντιπροσωπεύει το χώρο εργασίας και έχει διαστάσεις 35X 50X 3,5 cm. Τα ξύλα είναι επικολλημένα μεταξύ τους με βίδες. Πάνω στο χώρο εργασίας υπάρχουν δυο ακόμα ξύλινες βάσεις που αντιπροσωπεύουν το χώρο εισαγωγής ενός αντικειμένου που χαρακτηρίζουμε σαν θέση Α και το χώρο παραλαβής που χαρακτηρίζουμε σαν θέση Β. Οι διαστάσεις αυτές και για τα δυο μέρη είναι 7X7 cm. Εντός αυτών υπάρχουν κολλημένα τα δυο buttons. Πάνω στο χώρο εργασίας υπάρχει άλλη μια βάση που στηρίζει ολόκληρο το ρομποτικό βραχίονα διαστάσεων 15X15. Ο ένας κινητήρας είναι σφηνωμένος μέσα σε αυτή τη βάση που τον παγιδεύει και τον κρατά ακίνητο. Υπάρχει επίσης η κλέμα που δίνει τη βασική τροφοδοσία καθώς και η ασφαλειοθήκη.

Ο κινητήρας αυτός έχει μία ειδική βάση που μπορεί να προσκολληθεί πάνω στο κεντρικό του γρανάζι και να βιδωθεί. Το πιο σημαντικό στοιχείο πριν να προχωρήσουμε στην κατασκευή είναι να καθορίσουμε το εύρος λειτουργίας δηλαδή τις μοίρες που μπορεί να λειτουργήσει. Έχοντας υπ' όψιν αυτό το παράγοντα προσκολλούμε τον άλλο κινητήρα στη βάση του προηγούμενου κάνοντας κάποιες παρεμβάσεις στο περίβλημα του. Η κόλληση έγινε με θερμοκόλλα. Ομοίως καθορίζουμε τις μοίρες του δεύτερου κινητήρα έτσι ώστε να προσδιορίσουμε το εύρος λειτουργίας του στο χώρο εργασίας. Βιδώνουμε σε αυτόν τον αντάπτορά του και προσκολλήσαμε μια μεταλλική γωνία. Στο τέλος της γωνίας βιδώσαμε την αρπάγη που έχει δυο βαθμούς ελευθερίας, κάνοντας μια κίνηση από 0 εως 180 μοίρες πάνω κάτω και μια κίνηση που ανοιγοκλείνει τη δαγκάνα από 0 έως 90 μοίρες.

Υπάρχει άλλη μία μικρή λεπτή βάση όπου έχουμε κολλήσει το breadboard με το κύκλωμα του, τον Arduino μαζί με το shield του και την πλακέτα με τα joystick καθώς επίσης και ένα διακόπτη.

Τα καλώδια των κινητήρων είναι πιασμένα με τέτοιο τρόπο ώστε να μπορεί να έχει περιθώριο στις κινήσεις του χωρίς να τεντωθούν ή να πιαστούν πουθενά. Είναι πιασμένα με δεματικά και όλοι οι κινητήρες χρησιμοποιούν προέκτασεις στα καλώδια τους. Είναι μαζεμένα όλα μαζί μέσα σε ένα σπιράλ το οποίο τα βοηθάει να μην εμπλακούν πουθενά και τους δίνει βάρους ώστε να μην κουνιούνται οι επαφές.

4.

Επεξήγηση Κώδικα

Ο Manager - interface του Arduino όπου γράφουμε το κώδικά μας λειτουργεί σε C++

Θα αναλύσουμε το κάθε κομμάτι ξεχωριστά και σε βάθος ώστε να ξεκαθαρίσουμε το πώς λειτουργεί ο βραχίονας.

Θα ξεκινήσουμε με τις αρχικοποιήσεις, στη συνέχεια θα αναλύσουμε την συνάρτηση setup() μετά την συνάρτηση loop() και τέλος όλες τις βοηθητικές συναρτήσεις που καλούνται στα διάφορα μέρη του προγράμματος μας.

Αρχικοποιήσεις:

```
#include <Servo.h>  //including servo library and functions

Servo myservo1;   // creating "Servo" objects
Servo myservo2;
Servo myservo3;
Servo myservo5;
```

Στο παραπάνω κομμάτι κώδικα εισάγουμε και κάνουμε χρήση της βιβλιοθήκης Servo.h η οποία προφανώς έχει δημιουργηθεί από κάποιο άλλο προγραμματιστή και μας γλιτώνει από κόπο και χρόνο. Μαζί με την βιβλιοθήκη δίνονται φυσικά και οι απαραίτητες συναρτήσεις που μπορεί να χρησιμοποιήσει ο προγραμματιστής. Ενδεικτικά αναφέρουμε τις attach() ,write() , writeMicroseconds() ,read(), attached() και detach(). Όσες από αυτές χρησιμοποιήσουμε παρακάτω θα τις αναλύσουμε, διαφορετικά όλες οι πληροφορίες διαθέτονται από την ιστοσελίδα του προγραμματιστή.

Στη συνέχεια δημιουργούμε τέσσερα αντικείμενα τύπου Servo διότι τόσοι είναι και οι βαθμοί ελευθερίας δηλαδή και οι κινητήρες μας στο βραχίονα.

Όπου ο myservo1 αντιστοιχεί στο κινητήρα της βάσης, ο myservo2 αντιστοιχεί στον αγκώνα, ο myservo3 αντιστοιχεί στη βάση που κάνει τη κίνηση πάνω κάτω και ο myservo5 αντιστοιχεί στην αρπάγη.

```
boolean a;     // defines if there is an item at the position A
boolean b;     // defines if there is an item at the position B
```

Πιο πάνω ορίζουμε δυο μεταβλητές τύπου Boolean που θα παίρνουν τις τιμές True ή False
Αναλόγως και λαμβάνουμε υπόψη μας και το ηλεκτρονικό κομμάτι, δηλαδή το ότι ο διακόπτης A είναι αρνητικής λογικής, δηλαδή όταν είναι πατημένος επιστρέφει False.

```
int potpin1 = 1;  // analog pin used to connect the potentiometer
int potpin2 = 2;
int potpin3 = 3;
int potpin5 = 5;
```

Για τους αντίστοιχους κινητήρες που αναγράφονται πιο πάνω ορίζουμε και τα αντίστοιχα pins όπου τερματίζουν τα μεσαία pins από τα ποτενσιόμετρα. Αυτή η διαφορά στην αντίσταση μας δίνει μια διαφορά και στην τάση φυσικά που την διαβάζει ο μΕ και την καταχωρεί σαν ακέραια τιμή από 0 έως 1023 χρησιμοποιώντας τον 8bit ADC.
Ο τρόπος με ον οποίον κάνουμε αυτή τη δήλωση μας βοηθά στο να ορίσουμε σε ποιο pin καταλήγει το ποδαράκι από το ποτενσιόμετρο. Για παράδειγμα στο αναλογικό ποδαράκι 1 αποθηκεύεται μια ακέραια τιμή με την ονομασία potpin1. Αυτή η τεχνική θα χρησιμοποιηθεί παρακάτω, διότι εδώ προσωρινά έτσι όπως το έχουμε δηλώσει σημαίνει ότι η ακέραια μεταβλητή potpin1 έχει την τιμή 1 κ.ο.κ..

```
int val1;      // variable to read the value from the analog pin
int val2;
int val3;
int val5;
```

Τις πιο πάνω ακέραιες τιμές τις χρησιμοποιούμε για να αποθηκεύσουμε τελικά τις τιμές που θα έχουμε από τα αντίστοιχα ποτενσιόμετρα. Η val προέρχεται από τη λέξη value.

```
int switcher =8; //switch from automatic to manual
```

Με αυτό τον τρόπο ορίζουμε μια ακέραια μεταβλητή που την ονομάζουμε switcher της δίνουμε την τιμή 8 και θα το χρησιμοποιήσουμε παρακάτω για να δηλώσουμε στο πρόγραμμά μας ότι ο διακόπτης είναι στο pin 8 του μΕ

```
boolean man;   // used to store the switch state
```

ορίζουμε μια Boolean μεταβλητή που την ονομάζουμε man και θα χρησιμοποιηθεί παρακάτω για την μεγαλύτερη if δήλωση του προγράμματος που θα καθορίζει εάν το σύστημα θα βρίσκεται σε αυτόματη ή σε χειροκίνητη λειτουργία.

Η συνάρτηση setup() :

Η συνάρτηση setup() είναι μια απαραίτητη συνάρτηση για την λειτουργία της πλατφόρμας του Arduino και είναι τύπου void δηλαδή δεν επιστρέφει κάποια τιμή.
Εκεί μέσα γίνονται όλες οι κυρίως αρχικοποιήσεις ή το σετάρισμα των αντικειμένων ή και άλλων συναρτήσεων από βιβλιοθήκες ή μη που έχουμε δηλώσει στο πρόγραμμά μας.

```
void setup() {        // setting up parameters function
 Serial.begin(9600);  // function for serial print
 myservo1.attach(9);  // attaches the servo on pin number to the servo object
 myservo2.attach(6);
 myservo3.attach(5);
 myservo5.attach(3);
 hardhome();
 pinMode(switcher,INPUT); // define pin mode
 pinMode(2,INPUT);      // input for the position A
 pinMode(4,INPUT);      // input for the position B
        }
```

Δηλαδή ξεκινάμε καλώντας τη συνάρτηση Serial.begin(9600); Όπου αυτή η συνάρτηση χρησιμοποιείται εδώ για να προετοιμάσει το μΕ ότι κάπου στο πρόγραμμα θα χρειαστεί να ανεβάσει δεδομένα στην οθόνη του μΕ και μάλιστα του ορίζει την ταχύτητα επικοινωνίας του με τη θύρα baud rate στα 9600.

Από την βιβλιοθήκη servo χρησιμοποιούμε την συνάρτηση attach που δηλώνει σε ποιο ποδαράκι του μΕ έχουμε τερματίσει το καλωδιάκι από το σήμα του servo. Δηλαδή στην περίπτωσή μας ο κινητήρας 5 που αντιστοιχεί στην αρπάγη έχει τερματιστεί στο pin 3 του Arduino. Αντίστοιχα και όλοι οι άλλοι. Αξίζει να σημειωθεί ότι τα pins 3,5,6,9 είναι και αυτά που μπορούν να χρησιμοποιήσουν παλμούς PWM pulse with modulation.

Έπειτα καλείται η βοηθητική συνάρτηση hardhome() που έχει οριστεί παρακάτω και εφόσον βρίσκεται μέσα στη setup() θα τρέξει μια φορά οπωσδήποτε.

Η δεσμευμένη λέξη pinMode(pin,I/O) μας ορίζει σαν τι χρησιμεύει το pin, δηλαδή σαν είσοδο ή σαν έξοδο. Εδώ ο switcher έχει τη τιμή 8 δηλαδή είναι το ποδαράκι 8 και είναι είσοδος. Το ίδιο ισχύει και για τα pin 2 και 4 που όμως δηλώνεται κατευθείαν το ποδαράκι χωρίς να πρέπει να αποθηκευτεί σε μεταβλητή πρώτα. Αυτά χρησιμοποιούνται για να τερματίσουμε τα buttons των θέσεων A και B αντίστοιχα. Θα δέχονται 5V ή 0 ή κάποια απροσδιόριστη κατάσταση για είσοδο.

Η συνάρτηση loop είναι και αυτή μια συνάρτηση τύπου void αφού φυσικά δεν επιστρέφει τίποτα. Θα μπορούσε κανείς να πει ότι θεωρητικά δεν είναι απαραίτητη για να λειτουργήσει γενικά ένα σύστημα απλά θα εκτελούνταν όλες οι εντολές μόνο μια φορά και τέλος. Σπανίως να απαιτηθεί να δημιουργήσει ένα τέτοιο σύστημα ο μηχανικός. Εφόσον θέλουμε να ελέγχουμε συνεχώς αν έχουν υπάρξει οποιεσδήποτε μεταβολές είτε στα αισθητήρια είτε οπουδήποτε θα πρέπει να κάνουμε συνεχώς ελέγχους σε ένα μεγάλο βρόγχο. Έτσι κάνουμε χρήση αυτής της εντολής που θα μπορούσαμε επίσης να την δηλώσουμε και αλλιώς π.χ. while(1) { εντολές;},(βλέπε[2]).

```
void loop() {        // main loop function
printtoscrean();
man = digitalRead(switcher);
 a = digitalRead(2);
 b = digitalRead(4);
```

Μέσα στον βρόγχο καλούμε την βοηθητική συνάρτηση printtoscrean() και αυτή εκτελείται. Θα περιγράψουμε την λειτουργία της ξεχωριστά.
Στη συνέχεια χρησιμοποιώντας την εντολή digitalRead και τη μεταβλητή διαβάζουμε από το ψηφιακό pin που είναι τερματισμένος ο διακόπτης την τιμή αυτή και την αποθηκεύει στην Boolean μεταβλητή man. Το ίδιο κάνουμε και για τις μεταβλητές a,b που δεν χρησιμοποιούμε μεταβλητή μέσα στη παρένθεση απλά γράφουμε τον αριθμό του pin.

Ακολουθεί η κύρια if του προγράμματος που διευθετεί τον αν το ρομπότ θα κινείται αυτόματα ή χειροκίνητα

```
if (man ==HIGH) {
   manual();      //calls the manual function
```

Ελέγχει αν η man μεταβλητή είναι «1» και αν είναι τότε καλεί την Βοηθητική συνάρτηση manual() που θα αναλύσουμε ξεχωριστά. Σε περίπτωση που δεν είναι τότε κινείται κάτω από το βρόγχο της «else».

```
                    }
else {
```

Είναι λογικό ότι εφόσον ο προηγούμενος βρόγχος χρησιμεύει για την χειροκίνητη λειτουργία ότι αυτό θα είναι για την αυτόματη.

Αναλυτικότερα συνεχίζει με την εντολή digitalWrite (pinnumber, STATE); όπου μπορούμε να αλλάξουμε την έξοδο σε οποιοδήποτε ποδαράκι του μΕ από λογικό μηδέν σε λογικό ένα.
Άρα εδώ πέρα ανάβει ένα λεντάκι πάνω στο shield του Arduino που μας υποδηλώνει ότι βρίσκεται στην αυτόματη λειτουργία. Το λεντάκι που δηλώνει ότι είναι στη χειροκίνητη είναι φυσικά κλειστό.

```
digitalWrite(10,HIGH);        // turns on the led for the automatic operation
```

Η παρακάτω συνθήκη μας εξασφαλίζει ότι ο βραχίονας δεν θα κάνει απολύτως τίποτα άμα υπάρχει αντικείμενο στη θέση Β και υπάρχει και αντικείμενο στη θέση Α.
Αντιθέτως αν η θέση Β είναι ελεύθερη και η θέση Α είναι κατειλημμένη τότε θα ξεκινήσει και θα τρέξει την παρακάτω ρουτίνα. Όμως προγραμματιαστικά έχουμε δηλώσει σαν να φαίνεται ότι θα ξεκινήσει ακόμα και αν δεν υπάρχει αντικείμενο στη θέση Α. Αυτό γίνεται διότι ο διακόπτης της θέσης Α είναι αρνητικής λογικής δηλαδή λαμβάνουμε μια κατάσταση High σαν LOW.

```
        if (a ==LOW && b==LOW)  //by using two inputs (the one corresponding to the A position is a
negative logic button)
 {                      //we manage to control that the robot will do nothing unless there is an item to
the position A
   delay(1500);                 // waiting to be picked up AND there is not an item to the position B
```

Αρχικά περιμένει 1.5 δευτερόλεπτα για να προλάβουμε να ακουμπήσουμε άνετα το αντικείμενο στη θέση Α.
Στη συνέχεια τρέχει μια σειρά από βοηθητικές συναρτήσεις που θα αναλύσουμε ξεχωριστά την κάθε μία αλλά με τους τίτλους και τις διεργασίες που κάνουν βοηθούν το προγραμματιστή να μπει στη λογική του πως ακριβώς λειτουργεί ο βραχίονας βήμα προς βήμα. Το γεγονός ότι είναι χωρισμένος σε τμήματα μας δίνει τη δυνατότητα να διορθώσουμε λάθη ή να κάνουμε τροποποιήσεις σε συγκεκριμένα κομμάτια, για παράδειγμα αν θέλουμε να αυξήσουμε την ταχύτητα στο κομμάτι που κινείται ο βραχίονας για να πιάσει το αντικείμενο θα επεξεργαστούμε μόνο τη συνάρτηση gotograb().

```
opengripper();
gotograb();
closegripper();
goto_middle_position_with_item();
leave_the_item();
opengripper();
goto_middle_position_without_item();
```

Η λογική του πως λειτουργούν οι βοηθητικές void συναρτήσεις είναι εμφανέστατη και από τους τίτλους τους. Δηλαδή μόλις ξεκινήσει ο βραχίονας ανοίγει την αρπάγη, εξασφαλίζοντας ότι δεν χρειάζεται να το κάνει κάπου ενδιάμεσα. Στη συνέχεια πηγαίνει να σηκώσει το αντικείμενο. Αφού φτάσει στη σωστή θέση και απόσταση κλείνει την αρπάγη και το συγκρατεί. Επειδή οι ταχύτητες και κάποιοι άλλοι παράμετροι αλλάζουν όταν κρατάει το αντικείμενο του έχουμε δώσει την συνάρτηση που πηγαίνει στη μεσαία θέση που αναφέραμε πιο πάνω. Ύστερα κινείται προς τη θέση B και μόλις φτάσει σε συγκεκριμένες μοίρες ανοίγει την αρπάγη. Τέλος εφόσον δεν υπάρχει αντικείμενο κινείται κάπως διαφορετικά στην μεσαία θέση.

Οποιεσδήποτε αλλαγές θέλουμε να κάνουμε πηγαίνουμε επεμβαίνουμε απλά στη ανάλογη συνάρτηση.

```
  }
      if (b==HIGH && a==HIGH) { // makes the buzzer go on and off 3 times
   for (int i=0; i<3; i++) {
digitalWrite (12,HIGH);
delay(500);
digitalWrite(12, LOW);
delay(500);              }

                        }
  }


          }
```

Στο πιο πάνω κομμάτι κώδικα δεν είναι τίποτα άλλο από το να αναβοσβήνει τρεις φορές το buzzer με μια απλή επανάληψη for και μια τοπική μεταβλητή εντός της for. Του δίνει ρεύμα κάνει μια καθυστέρηση για μισό δευτερόλεπτο και του κόβει το ρεύμα για μισό δευτερόλεπτο.

Οι υπόλοιπες αγκύλες τερματίζουν τις if, else, for και loop.

Συνεχίζουμε με τις βοηθητικές συναρτήσεις:

```
void hardhome(){              //we define hardhome function by setting servo parameters
 for (int j = 0; j<2; j++){   // buzzer on and off twice
  digitalWrite(12,HIGH);
  delay(500);
  digitalWrite(12,LOW);
  delay(500);
                }
delay(500);
myservo1.write(120);
delay(500);
myservo2.write(30);
delay(500);
myservo3.write(180);
delay(500);
myservo5.write(90);
delay(500);

      }
```

Η πιο πάνω συνάρτηση void χρησιμοποιείται για να «καρφώσουμε» τους κινητήρες σε κάποιες συγκεκριμένες θέσεις που θέλουμε εμείς.
Αρχικά αναβοσβήνουμε το buzzer 2 φορές.
Με τη εντολή write(degrees) που μας δίνεται από την βιβλιοθήκη Servo.h μπορούμε να κουνήσουμε το κινητήρα ακριβώς στις μοίρες που θέλουμε εμείς.
Εφόσον χρησιμοποιούμε τη βιβλιοθήκη που φυσικά εμπεριέχει κάποιες classes και structures αυτή η δήλωση γίνεται με τη μορφή object.write(degrees);

Όπως φαίνεται εδώ είναι myservo1.write(120) δηλαδή κούνα το κινητήρα 1 στις 120 μοίρες. Η μετάβαση αυτή γίνεται σχετικά γρήγορα και για όλες αυτές τους δίνουμε μια καθυστέρηση μισού δευτερολέπτου. Έτσι όλοι οι κινητήρες κινούνται ακριβώς εκεί που τους ορίσαμε.

Η συνάρτηση void manual()

```
void manual() {    /* we define the manual function by using the values from the potentiometers */

  digitalWrite(11,HIGH);

  val1 = analogRead(potpin1);          // shoulder left and right movement
  val1 = map(val1, 0,1023 , 0, 90);    // distribution of the signal input to degrees
  myservo1.write(val1);               //moving the servo to specific degree in function with the value
  delay(50);                        // fructions of seconds until the servo reaches the set angle and discharge
current

  val2 = analogRead(potpin2);          // shoulder up and down movement
  val2 = map(val2, 0, 1023, 30, 130);
  myservo2.write(val2);
  delay(50);

  val3 = analogRead(potpin3);          // elbow up and down movement
  val3 = map(val3, 0, 1023, 0, 180);
  myservo3.write(val3);
  delay(50);

  val5 = analogRead(potpin5);          //gripper open and close movement
  val5 = map(val5, 0, 1023, 0, 125);
  myservo5.write(val5);
  delay(50);
        }
```

Αρχικά ανάβει ένα led πάνω στο shield που υποδηλώνει ότι βρισκόμαστε στην χειροκίνητη λειτουργία.

Στη συνέχεια χρησιμοποιεί την εντολή analogRead και διαβάζει την τιμή από το ποτενσιόμετρο. Το potpin1 έχουμε δηλώσει στην αρχή του κώδικά εκεί που κάνουμε τις αρχικοποιήσεις ότι έχει την τιμή 1. Δηλαδή ότι το ποδαράκι του ποτενσιόμετρου τερματίζεται στο pin1. Ομοίως και για όλα τα άλλα ποτενσιόμετρα. Αυτή την τιμή την διαβάζει και την καταχωρεί στη ακέραια μεταβλητή val1

Στη συνέχεια χρησιμοποιεί την συνάρτηση map. Αυτή η συνάρτηση δεν κάνει τίποτε άλλο παρά μόνο να αντιστοιχεί αριθμούς με αριθμούς. Δηλαδή εάν είχαμε μια μεταβλητή K και μπορούσε να πάρει τιμές από το 0 έως το 100 και θέλαμε αυτές τις τιμές να τις αντιστοιχήσουμε σε έναν 8bit καταχωρητή που παίρνει τιμές από 0 έως 1023 δηλαδή 1024 τιμές θα το συντάσσαμε έτσι map(K, 0,1023,0,100); Αρχικά μπαίνει η μεταβλητή έπειτα η αρχική και η τελική μεταβλητή του καταχωρητή και έπειτα η αρχική και η τελική τιμή που μπορεί να πάρει η K.

Στην περίπτωσή μας οι τιμές αυτές είναι οι μοίρες του κινητήρα. Εμείς μπορούμε να διαλέξουμε το εύρος που θέλουμε ή το εύρος που μας επιβάλλει ο χώρος εργασίας.
Για παράδειγμα δεν θέλουμε ο βραχίονας να ακουμπήσει ποτέ κάτω στο έδαφος άρα μπορούμε να κόψουμε τις αντίστοιχες μοίρες και να μην ανησυχεί ο χειριστής ότι μπορεί να κάνει κάποια ζημιά στο μηχάνημα.

Αφού η map κάνει την αντιστοίχηση πρέπει κάπου να αποθηκεύσει την τιμή που θα βρει.
Για να μην κάνουμε σπατάλη στις μεταβλητές και για να κάνουμε το κώδικα πιο εύκολο χρησιμοποιούμε πάλι την ίδια μεταβλητή val
Η τελική σύνταξη αυτής της εντολής είναι :

```
val1 = map(val1, 0,1023 , 0, 90);
```

Άρα το μόνο που απομένει είναι να κινήσουμε το βραχίονα στις μοίρες που θέλουμε δηλαδή στις μοίρες που αντιστοιχούν στην θέση που βρίσκεται το ποτενσιόμετρο εκείνη τη στιγμή. Αυτό το κάνουμε με την εντολή

```
myservo1.write(val1);
```

Τέλος κάνουμε ένα μικρό delay για κάποια milliseconds ώστε να δώσουμε χρόνο στο κινητήρα αρχικά να φτάσει και αφετέρου να αποφορτιστούν τα αντίστροφα ρεύματα.

Η ίδια διαδικασία γίνεται και για τους τέσσερις κινητήρες και όλο αυτό το κομμάτι κώδικα που συνιστά τη χειροκίνητη λειτουργία βρίσκεται μέσα σε έναν ατέρμων βρόγχο οπότε οι τιμές από τα ποτενσιόμετρα διαβάζονται συνεχώς και έχουμε συνεχώς κίνηση.

Η συνάρτηση printtoscreen()

```
void printtoscrean() {

    Serial.println (val1);
    Serial.println (val2);
    Serial.println (val3);
    Serial.println (val5);
    Serial.println (man);
    delay(200);
                }
```

Είναι και αυτή μια void συνάρτηση που απλώς μας εμφανίζει στην οθόνη τις τιμές στις οποίες βρίσκεται ο κινητήρας, δηλαδή τις μοίρες του για να έχουμε μια γενική ιδέα του πως κινείται. Εξηγήσαμε προηγουμένως ότι οι μοίρες είναι σχετικές και εξαρτώνται από το πώς έχουν τοποθετηθεί οι κινητήρες στη κατασκευή. Αυτή η συνάρτηση μας δίνει μια εικόνα για αυτούς που πρέπει να κινηθούν «ανάποδα». Η εντολή Serial.println(μεταβλητή) σχετίζεται άμεσα με την συνάρτηση Serial.begin(9600) που βρίσκεται στην void setup() του κώδικά μας. Χωρίς αυτήν δεν λειτουργεί. Παρατηρούμε ότι η τελευταία μεταβλητή είναι τύπου Boolean που παίρνει οποιαδήποτε τιμή εκτός του μηδέν όταν είναι αληθής και 0 όταν είναι ψευδής,(βλέπε[1]). Υπενθυμίζουμε ότι η τιμή αυτή μας δείχνει σε ποια λειτουργία βρίσκεται ο βραχίονας.

Η συνάρτηση gotograb()

```
void gotograb(){   /*function that follows all the positions until the A area without holding an item.
Approaches the item with variable speed */
delay(15);
myservo1.write(120);
myservo3.write(30);
delay(500);
myservo2.write(30);
delay(500);
myservo2.write(100);
delay(100);
myservo2.write(110);
delay(100);
myservo2.write(130);
delay(200);
for (int w=30; w<40; w++) {
myservo3.write(w);
delay(120);          }
for (int w=50; w<73; w++) {
myservo3.write(w);
delay(70);
                    }
              }
```

Όπως υποδηλώνει και το όνομά της είναι αυτή που δίνει στο βραχίονα όλες τις απαραίτητες κινήσεις για να πάει να πιάσει το αντικείμενο.
Είναι μια σχετικά απότομη κίνηση του βραχίονα για να εξοικονομήσουμε χρόνο. Το πετυχαίνουμε αυτό αλλάζοντας τα delay. Όταν πλησιάσει αρκετά μπαίνουμε σε μια επανάληψη με τοπική μεταβλητή όπου παίρνει όλες τις ενδιάμεσες τιμές με μικρό delay με αποτέλεσμα να κινείται πολύ γραμμικά και μαλακά.

Η συνάρτηση gotomiddlepositionwithitem()

Εδώ κρατάμε ήδη το αντικείμενο και ξεκινά η ανύψωσή του. Άρα αρχικά πρέπει να κινηθούμε αργά και στη συνέχεια λίγο πιο γρήγορα.

```
void goto_middle_position_with_item()                { //using variable speed and moving most of the
servos silmutaneously
 for (int n=73; n>50; n--){    // using local variables into the loops
   myservo3.write(n);
   delay(150);
                    }
    for (int n=50; n>10; n--)    { //changing speed at the same servo
   myservo3.write(n);
```

```
delay(50);                }
```

Στις δυο πιο πάνω for κινούμε τον ίδιο κινητήρα δίνοντας του όλες τις ενδιάμεσες τιμές
Από τις 73 στις 10 μοίρες. Όμως το κάνουμε σε δυο κομμάτια διαδοχικά το ένα με το άλλο και αυτό γιατί θέλουμε απλά όταν είναι κοντά στο έδαφος να κινηθεί αργά και μόλις αποκτήσει κάποια απόσταση να κινηθεί πιο γρήγορα. Παίρνει όλες τις τιμές ανα μοίρα και έτσι είναι πολύ γραμμικό και ευσταθές.

```
for (int k =133; k>90; k--)       { //elevating gently servo2
 myservo2.write(k);
 delay(50);                }
```

Εδώ ανεβάζουμε τον ώμο πιο ψηλά με μαλακές κινήσεις.

Δηλώνουμε μια float τοπική μεταβλητή για την ταυτόχρονη κίνηση και την αρχικοποιούμε.

```
float f=90.5;      //calculated the exact degrees of the servo2 for the silmutaneous movement
```

```
for (int h=120; h>25; h--){ //silmutanious movement of servo1 and servo2
myservo2.write(f);
f= f-0.5;              // by reducing the degrees we elevate the shoulder higher
myservo1.write(h);
delay(60);
                }
```

Στη πιο πάνω επανάληψη κινούμε ταυτόχρονα και τους δυο άξονες του ώμου. Η f μειώνεται με σταθερό ρυθμό και στην κατάληξη της έχει ακέραιες μοίρες. Γι' αυτό είχε αρχική τιμή 90,5 ώστε στη τελευταία επανάληψη να είναι ακέραιος.

```
for (int x=10; x<100; x++) {  //moving servo3 at 100 degrees

 myservo3.write(x);
 delay(1);
 Serial.print("angle of servo  ");
 Serial.println(f);          } //it prints the degrees of the servo2
                              }
```

Στο τέλος κινούμε και τον αγκώνα σε κάποια επιθυμητή μας θέση που αντιστοιχεί στις 100 μοίρες.

Εκτυπώνουμε τοπικά την f για να γνωρίζουμε ποια είναι αυτή η γωνία για να την χρησιμοποιήσουμε στην επόμενη μας κίνηση χωρίς να παρουσιαστεί καμία απότομη μεταβολή.

Η void συνάρτηση leave_the_item()

Το ρομπότ κινείται από τη μεσαία θέση προς τη Β για να ακουμπήσει όπως λέει και το όνομα το αντικείμενο.

```
void leave_the_item(){
  float z=100.00;
   for (int s=43; s<133; s++) {
    z= z-1.5;
    myservo2.write(s);
    myservo3.write(z);
    delay(60);
    Serial.print("angle of servo  ");
    Serial.println(z);
   }
   for (int m=0; m<75; m++){
    myservo3.write(m);
    delay(60);}
              }
```

Με την ίδια τεχνική που αναφέραμε και προηγουμένως δηλώνοντας ένα δεκαδικό και κάνοντας μια ταυτόχρονη επανάληψη κινούμε δυο άξονες και εκτυπώνουμε τις τελικές τους γωνίες. Από εκεί και πέρα μόλις φτάσουν στην επιθυμητή απόσταση τότε κινούμε μόνο τον έναν με την παραπάνω επανάληψη γραμμικά σταθερά και αργά.

Η συνάρτηση void go_to_middle_position_without_item()

Αφού αφήσει το αντικείμενο πρέπει όσο είναι ακόμα κοντά του το τελικό στοιχείο δράσης να κινηθεί αργά. Μολις η αποσταση αυξηθεί να κινηθεί πιο γρήγορα και μόλις είναι ακόμα πιο μακρια να παέι ακόμα πιο γρήγορα.

```
void goto_middle_position_without_item() {

  for (int x=73; x>30; x--){
  myservo3.write(x);
  delay(70);
                }
```

Απομακρύνεται μόνο η αρπάγη

```
  for (int q=133; q>70; q--)
{
myservo2.write(q);
delay(70);
}
```

Απομακρύνεται ο ώμος

```
myservo3.write(100);
myservo2.write(40);
myservo1.write(20);
delay(50);
```

Καρφώνονται πολύ απότομα στην μεσαία θέση όλοι οι κινητήρες

```
}
```

Η void συνάρτηση opengripper()

```
void opengripper() {   //function that opens the gripper

for(int i=89; i<178; i++) {
myservo5.write(i);
delay(15);                    }

        }
```

Εδώ σε σχέση με το αντικείμενο κάνει μια μικρή απότομη κίνηση ανοίγει η αρπάγη γρήγορα με μια απλή επανάληψη.

Η void συνάρτηση close_gripper()

```
void closegripper(){  //closing gripper function

for (int i=178; i>89; i--) {
myservo5.write(i);
delay(15);
                              }
                    }
```

Κλείνει η αρπάγη μέχρι ένα σημείο. Αν επιχειρούσαμε μέχρι το μηδέν υπάρχει περίπτωση να καεί ο κινητήρας από το ρεύμα.

Να αναφέρουμε ότι ο κώδικας είναι πλήρως στοιχισμένος με τις αγκύλες του

Και περιέχει αρκετά σχόλια για να διευκολύνει το προγραμματιστή.
Επίσης στις επαναλήψεις όλες οι μεταβλητές είναι τοπικές και χρησιμοποιούνται από τις τοπικές συναρτήσεις χωρίς να κάνουμε υπερφόρτωση τελεστών σε αυτές.

Ολόκληρος ο κώδικας προς διευκόλυνση του αναγνώστη:

```
#include <Servo.h>  //including servo library and functions

Servo myservo1;   // creating "Servo" objects
Servo myservo2;
Servo myservo3;
Servo myservo5;

boolean a;     // defines if there is an item at the position A
boolean b;     // defines if there is an item at the position B

int potpin1 = 1;  // analog pin used to connect the potentiometer
int potpin2 = 2;
int potpin3 = 3;
int potpin5 = 5;


int val1;      // variable to read the value from the analog pin
int val2;
int val3;
int val5;

int switcher =8; //switch from automamatic to manual
boolean man;    // used to store the switch state
```

```
void setup() {       // setting up parameters function
 Serial.begin(9600); // function for serial print
 myservo1.attach(9); // attaches the servo on pin number to the servo object
 myservo2.attach(6);
 myservo3.attach(5);
 myservo5.attach(3);
 hardhome();
 pinMode(switcher,INPUT); // define pin mode
 pinMode(2,INPUT);      // input for the position A
 pinMode(4,INPUT);      // input for the position B
        }


void loop() {        // main loop function
printtoscrean();
man = digitalRead(switcher);
 a = digitalRead(2);
 b = digitalRead(4);
if (man ==HIGH) {
   manual();      //calls the manual function
          }
else {
     digitalWrite(10,HIGH);       // turns on the led for the automatic operation
          if (a ==LOW && b==LOW)  //by using two inputs (the one corresponding to the A position is a
negative logic button)
 {                        //we manage to control that the robot will do nothing unless there is an item to
the position A
   delay(1500);                // waiting to be picked up AND there is not an item to the position B
   opengripper();
   gotograb();
   closegripper();
   goto_middle_position_with_item();
   leave_the_item();
   opengripper();
   goto_middle_position_without_item();



 }
       if (b==HIGH && a==HIGH) { // makes the buzzer go on and off 3 times
  for (int i=0; i<3; i++) {
digitalWrite (12,HIGH);
delay(500);
digitalWrite(12, LOW);
delay(500);             }

                        }
 }
```

```
        }

 void printtoscrean() {

     Serial.println (val1);
     Serial.println (val2);
     Serial.println (val3);
     Serial.println (val5);
     Serial.println (man);
     delay(200);
                }


 void hardhome(){            //we define hardhome function by setting servo parameters
  for (int j = 0; j<2; j++){   // buzzer on and off twice
    digitalWrite(12,HIGH);
    delay(500);
    digitalWrite(12,LOW);
    delay(500);
                }
  delay(500);
  myservo1.write(120);
  delay(500);
  myservo2.write(30);
  delay(500);
  myservo3.write(180);
  delay(500);
  myservo5.write(90);
  delay(500);

       }

 void manual() {    /* we define the manual function by using the values from the potentiometers */

  digitalWrite(11,HIGH);

  val1 = analogRead(potpin1);         // shoulder left and right movement
  val1 = map(val1, 0,1023 , 0, 90);    // distribution of the signal input to degrees
  myservo1.write(val1);              //moving the servo to specific degree in function with the value
  delay(50);                    // fructions of seconds until the servo reaches the set angle and discharge
current

  val2 = analogRead(potpin2);         // shoulder up and down movement
  val2 = map(val2, 0, 1023, 30, 130);
  myservo2.write(val2);
  delay(50);
```

```
  val3 = analogRead(potpin3);          // elbow up and down movement
  val3 = map(val3, 0, 1023, 0, 180);
  myservo3.write(val3);
  delay(50);

  val5 = analogRead(potpin5);          //gripper open and close movement
  val5 = map(val5, 0, 1023, 0, 125);
  myservo5.write(val5);
  delay(50);
        }

  void gotograb(){   /*function that follows all the positions until the A area without holding an item.
Approaches the item with variable speed */
delay(15);
myservo1.write(120);
myservo3.write(30);
delay(500);
myservo2.write(30);
delay(500);
myservo2.write(100);
delay(100);
myservo2.write(110);
delay(100);
myservo2.write(130);
delay(200);
for (int w=30; w<40; w++) {
myservo3.write(w);
delay(120);          }
for (int w=50; w<73; w++) {
myservo3.write(w);
delay(70);
                  }
            }

void opengripper() {   //function that opens the gripper

for(int i=89; i<178; i++) {
myservo5.write(i);
delay(15);                  }

            }



void closegripper(){  //closing gripper function
```

```
for (int i=178; i>89; i--) {
myservo5.write(i);
delay(15);
                        }

                    }

void goto_middle_position_with_item()                { //using variable speed and moving most of the
servos silmutaneously
 for (int n=73; n>50; n--){    // using local variables into the loops
   myservo3.write(n);
   delay(150);
                    }
    for (int n=50; n>10; n--)    { //changing speed at the same servo
   myservo3.write(n);
   delay(50);                }

   for (int k =133; k>90; k--)       { //elevating gently servo2
    myservo2.write(k);
    delay(50);                  }

    float f=90.5;       //calculated the exact degrees of the servo2 for the silmutaneous movement
    for (int h=120; h>25; h--){ //silmutanious movement of servo1 and servo2
    myservo2.write(f);
    f= f-0.5;              // by reducing the degrees we elevate the shoulder higher
    myservo1.write(h);
    delay(60);
                    }
 for (int x=10; x<100; x++) {  //moving servo3 at 100 degrees

    myservo3.write(x);
    delay(1);
    Serial.print("angle of servo  ");
    Serial.println(f);             } //it prints the degrees of the servo2
                                    }
void leave_the_item(){
 float z=100.00;
  for (int s=43; s<133; s++) {
   z= z-1.5;
   myservo2.write(s);
   myservo3.write(z);
   delay(60);
   Serial.print("angle of servo  ");
   Serial.println(z);
  }
  for (int m=0; m<75; m++){
   myservo3.write(m);
   delay(60);}
```

```
            }

 void goto_middle_position_without_item() {

   for (int x=73; x>30; x--){
   myservo3.write(x);
   delay(70);
                    }
   for (int q=133; q>70; q--)
{
myservo2.write(q);
delay(70);
}
myservo3.write(100);
myservo2.write(40);
myservo1.write(20);
delay(50);

}
```

5.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Δρ.Δ.ΚΥΤΑΓΙΑΣ,"C++ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ",2000
[2]JESSE LIBERTY,"C++PROGRAMMMING LANGUAGE",1999
[3]WIKIPEDIA
[4]ΔΗΜ.Χ.ΒΟΥΚΑΛΗΣ-ΕΙΡΗΝΗ.Δ.ΒΟΥΚΑΛΗ,''ΡΟΜΠΟΤΙΚΗ-ΑΥΤΟΜΑΤΑ'',ΑΘΗΝΑ 2006
[5]ΚΩΝ/ΝΟΣ ΤΖΑΦΕΣΤΑΣ,''ΡΟΜΠΟΤΙΚΗ 2 ΕΥΦΥΗ ΚΑΙ ΕΠΙΔΕΞΙΑ ΡΟΜΠΟΤΙΚΑ ΣΥΣΤΗΜΑΤΑ'',2003
[6]RICHARD C.JAEGER,"ΜΙΚΡΟΗΛΕΚΤΡΟΝΙΚΗ",ΘΕΣ/ΝΙΚΗ 2003,1999