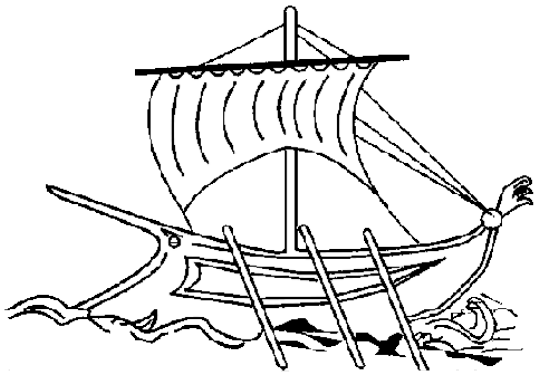
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ



ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

Τ.Ε.Ι. ΠΕΙΡΑΙΑ

ΡΟΜΠΟΤΙΚΟΣ ΒΡΑΧΙΟΝΑΣ

ΝΤΑΦΟΠΟΥΛΟΣ ΕΥΑΓΓΕΛΟΣ

Επιβλέπων Καθηγητής: ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΜΑΡΤΙΟΣ 2016

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

Π. Ράλλη & Θηβών 250 , 12244 Αιγάλεω , Αθήνα – Ελλάδα

Τηλ. 210-5381488

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ

Το περιεχόμενο της πτυχιακής εργασίας με τίτλο «Ρομποτικός Βραχίονας» αφορά στην προσπάθεια ελέγχου των κινήσεων ενός ρομποτικού βραχίονα.
Ο βραχίονας αυτός, πλαστικού υλικού, φέρει συγκεκριμένες διαστάσεις και κινείται εκμεταλευόμενος έξι σερβοκινητήρες. Η κίνηση γίνεται στο χώρο για τη μεταφορά μικρών αντικειμένων, έχοντας σταθερή βάση. Στόχος είναι η ομαλή κίνηση μεταξύ σημείων συγκεκριμένων συντεταγμένων με ταυτόχρονη αποφυγή εμποδίων ανά περίπτωση.
Όσον αφορά το λογισμικό, δηλαδή τον κώδικα που επιτρέπει την παραπάνω λειτουργία του Βραχίονα, το πρόβλημα προσεγγίστηκε με δυο διαφορετικούς τρόπους:

forward kinematics:

Σύμφωνα με αυτόν τον κώδικα χρησιμοποιούμε εξισώσεις, με σκοπό να υπολογίσουμε τη θέση και τον προσανατολισμό της αρπαγής (end-effector) στο χώρο, για δεδομένες τιμές των γωνιών των αρθρώσεών του. Ορίζουμε ως είσοδο τις γωνίες των σερβοκινητήρων και ως αποτέλεσμα έχουμε την τελική θέση του σημείου αρπαγής, η οποία ορίζεται σε τρισδιάστατες συντεταγμένες (x,y,z).

Ο δεύτερος τρόπος είναι πολυπλοκότερος, λόγω ότι υπάρχουν πολλές λύσεις για κάθε μια περίπτωση, και ονομάζεται:

inverse kinematics:

Κατ' αυτόν τον τρόπο ως είσοδο εκλαμβάνουμε τις τρισδιάστατες συντεταγμένες (x,y,z). Μέσω κινηματικών εξισώσεων καθορίζουμε τις γωνίες των σερβοκινητήρων, οι οποίοι καθοδηγούν το σύστημα αρπαγής στο επιθυμητό σημείο μέσα στο χώρο.

Αναλυτικότερα σε ό,τι αφορά τον κώδικα υπολογισμού γωνιών, αυτός δημιουργήθηκε στο **Matlab.** Υλοποίησε δε τις παραπάνω περιγραφές δίνοντας τις απαραίτητες πληροφορίες στο μικροελεγχτή **Arduino uno**, ο οποίος επικοινωνεί άμεσα με τους σερβοκινητήρες.
Σχετικά με τη σύνδεση μεταξύ Matlab/Arduino, αυτή υλοποιείται βάσει πρωτοκόλου σειριακής επικοινωνίας, στέλνοντας τα απαραίτητα δεδομένα από το Matlab στον Arduino.

Τα δεδομένα αυτά κατά κύριο λόγο αφορούν αντιστοιχούν στις κλίσεις, που παίρνουν τα servos.
Δουλειά του μικροελεγκτή απο εδώ και πέρα είναι ο τρόπος, που θα γίνουν αυτές οι κινήσεις, συμπεριλαμβανομένων των εμποδίων, μέσα στο χώρο αλληλεπιδρασης με το βραχίονα.

Για τον έλεγχο των εμποδίων χρησιμοποιήθηκε αισθητήριο όργανο, που μετράει αποστάσεις μέσω υπερήχων (ultra sonic). Το συγκεκριμένο module είναι το HC-SR04 και έχει δυνατότητα μέτρησης αποστάσεως από 2cm μέχρι 400cm. Αυτό το όριο ειναι αρκετό, για να καλύψει τις ανάγκες της συγκεκριμένης εφαρμογής.
Το module τοποθετήθηκε πάνω στη βάση του βραχίονα, για να περιστρέφεται μαζί με αυτόν, με τρόπον, ώστε η διάσταση της προέκτασης να ελέγχεται σε κάθε περίπτωση για κάθε κλίση του σερβοκινητήρα βάσης.

ΕΙΣΑΓΩΓΗ

Η εργασία αυτή αποτελεί την πτυχιακή εργασία στον Τομέα Αυτοματισμού και αφορά στον έλεγχο ενός βραχίονα εκπαιδευτικού μεγέθους.

Ως βιβλιογραφία χρησιμοποιήθηκαν βιβλία και σημειώσεις του Τμήματος Αυτοματισμού αλλά και υλικό από την ηλεκτρονική βιβλιοθήκη του διαδικτύου.

Γενικά ο ρομποτικός βραχίονας αποτελεί ένα «όργανο» οικείο στον Τομέα του Αυτοματισμού. Κι αυτό, διότι κάθε σχεδόν μονάδα παραγωγής ή και μεταφοράς προϊόντων πρέπει να διαθέτει τέτοια μηχανήματα, ώστε οι επί μέρους κατασκευές, συγκολλήσεις, μετακινήσεις των αντικειμένων και λοιπές εργασίες να γίνονται με ακρίβεια, ταχύτητα, ασφάλεια αλλά και με το μικρότερο δυνατόν κόστος. Πράγμα, που είναι δύσκολο και οικονομικά ασύμφορο, αν αυτό γίνεται δια του εργατικού τους δυναμικού.
Ο ρομποτικός βραχίονας από τη δεκαετία του 1950 που επινοήθηκε, εξελίχτηκε σταδιακά. Σήμερα υπάρχουν βραχίονες με διαφορετικά χαρακτηριστικά ως προς το βαθμό ελευθερίας, τη φέρουσα ικανότητα, την ταχύτητα, την επιτάχυνση, την ακρίβεια κλπ. Αλλά είτε ως μικρά «ρομπότ πάγκου» είτε ως μεγαλύτερα είναι πολύ σημαντικά ως προς την αποτελεσματικότητα της λειτουργίας των διαφόρων μονάδων.
Τυπικές εφαρμογές της ρομποτικής είναι η συγκόλληση, οι βαφές, η συναρμολόγηση, η τοποθέτηση όπως συσκευασίες, παλετοποιήσεις, ο έλεγχος προϊόντων, και οι δοκιμές. Και όλα αυτά με υψηλή αντοχή, ταχύτητα και ακρίβεια.
Μερικά ρομπότ προγραμματίζονται για την πιστή εκτέλεση συγκεκριμένων ενεργειών ξανά και ξανά (επαναλαμβανόμενες πράξεις) χωρίς μεταβολές και με υψηλό βαθμό ακρίβειας. Οι δράσεις αυτές καθορίζονται από προγραμματισμένες ρουτίνες που καθορίζουν την κατεύθυνση, την επιτάχυνση, την ταχύτητα, την επιβράδυνση, και την απόσταση από μια σειρά συντονισμένων κινήσεων.
Άλλα ρομπότ είναι πολύ πιο ευέλικτα ως προς τον προσανατολισμό του αντικειμένου το οποίο λειτουργούν, ή ακόμα και την εργασία που πρέπει να εκτελεστεί στο ίδιο το αντικείμενο, το οποίο μπορεί ακόμα να χρειαστεί να προσδιοριστεί από το ίδιο το ρομπότ. Για παράδειγμα, για πιο ακριβή καθοδήγηση, τα ρομπότ συχνά περιέχουν υποσυστήματα μηχανικής όρασης που ενεργούν ως "μάτια". Συνδεόμενα με ισχυρούς υπολογιστές ή ελεγκτές (controllers). Η τεχνητή νοημοσύνη, ή ό,τι μοιάζει με αυτή, γίνεται όλο και πιο σημαντικός παράγοντας στα σύγχρονα βιομηχανικά ρομπότ.

Ορισμός παραμέτρων βραχίονα

• Αριθμός αξόνων - δύο άξονες απαιτούνται για να φθάσουν σε οποιοδήποτε σημείο σε ένα επίπεδο. Τρεις άξονες απαιτούνται για την πρόσβαση σε κάθε σημείο του χώρου. Για πλήρη έλεγχο του προσανατολισμού στην άκρη του βραχίονα (δηλαδή τον καρπό), απαιτούνται τρεις περισσότεροι άξονες (εκτροπής, λαβής, και κύλισης). Μερικά σχέδια (π.χ. τα ρομπότ SCARA) έχουν περιορισμούς στη δυνατότητα κίνησης για το κόστος, την ταχύτητα και την ακρίβεια.

• Βαθμός ελευθερίας - που είναι συνήθως ο ίδιος με τον αριθμό των αξόνων.

• Φάκελος εργασίας - η περιοχή του χώρου που μπορεί να φτάσει ένα ρομπότ.

• Κινηματική - η πραγματική ρύθμιση των άκαμπτων μελών και αρθρώσεων του ρομπότ, που ορίζει ποιες κινήσεις του είναι δυνατόν να γίνουν. Οι κατηγορίες της κινηματικής στα ρομπότ περιλαμβάνουν τα αρθρωτά, τα καρτεσιανά, τα παράλληλα και τα SCARA.

• Φέρουσα ικανότητα ή ωφέλιμο φορτίο – Πόσο βάρος μπορεί να σηκώσει ένα ρομπότ.

• Ταχύτητα - Πόσο γρήγορα μπορεί το ρομπότ να φέρει σε λειτουργία το άκρο του βραχίονα του. Αυτό μπορεί να καθορίζεται με βάση την γωνιακή ή γραμμική ταχύτητα του κάθε άξονα ή ως συνδυασμός ταχύτητας, δηλαδή την ταχύτητα στο τέλος του βραχίονα, όταν όλοι οι άξονες κινούνται.

• Επιτάχυνση - Πόσο γρήγορα ένας άξονας μπορεί να επιταχύνει. Δεδομένου ότι αυτό είναι ένας περιοριστικός παράγοντας αφού ένα ρομπότ μπορεί να μην είναι σε θέση να φθάσει την καθορισμένη μέγιστη ταχύτητα για τις μετακινήσεις του σε μικρή απόσταση ή για μια σύνθετη διαδρομή που απαιτεί συχνές αλλαγές κατεύθυνσης.

• Ακρίβεια - Πόσο κοντά ένα ρομπότ μπορεί να φτάσει μια απαιτούμενη θέση. Όταν η απόλυτη θέση του ρομπότ μετριέται σε σχέση με την απαιτούμενη θέση το σφάλμα είναι ένα μέτρο ακρίβειας. Η ακρίβεια μπορεί να βελτιωθεί με εξωτερικούς αισθητήρες, για παράδειγμα ένα σύστημα όρασης ή υπέρυθρων. Δείτε βαθμονομήσεις ρομπότ. Η ακρίβεια μπορεί να ποικίλλει ανάλογα με την ταχύτητα, τη θέση εργασίας στο φάκελο εργασίας και με το ωφέλιμο φορτίο (βλέπε συμμόρφωση).

• Επαναληψιμότητα - Πόσο καλά το ρομπότ θα επιστρέψει σε μια προγραμματισμένη θέση. Αυτό δεν είναι το ίδιο με την ακρίβεια. Μπορεί να είναι όταν του δοθεί εντολή να πάει σε μια συγκεκριμένη θέση X-Y-Z και αυτό να γίνεται με απόκλιση όχι μεγαλύτερη του 1χιλ. Αυτή θα ήταν η ακρίβεια η οποία μπορεί να βελτιωθεί με βαθμονόμηση. Αν όμως αυτή η θέση διδάσκεται στη μνήμη του ελεγκτή, κάθε φορά που επιστρέφει σε εκείνο το σημείο το κάνει με απόκλιση 0,1 χιλ. της θέσης που έχει εκπαιδευτεί. Τότε η επαναληψιμότητα είναι μέσα σε αυτό το 0,1 χιλ.

• Ελέγχου κίνησης - για ορισμένες εφαρμογές, όπως στην απλή συνένωση pick-and-place, το ρομπότ χρειάζεται ελάχιστες επαναληπτικές επιστροφές στις προ-εκπαιδευθέντες θέσεις. Για πιο εξελιγμένες εφαρμογές, όπως η συγκόλληση και το φινίρισμα (σπρέι βαφής), η κίνηση πρέπει να ελέγχονται συνεχώς για να ακολουθήσει μια διαδρομή στο χώρο, με ελεγχόμενη ταχύτητα και προσανατολισμό.

• Τροφοδοσία - μερικά ρομπότ χρησιμοποιούν ηλεκτρικούς κινητήρες, άλλα χρησιμοποιούν υδραυλικούς ενεργοποιητές. Τα πρώτα είναι πιο γρήγορα, ενώ τα τελευταία είναι ισχυρότερα και πιο συμφέρουσα σε εφαρμογές όπως η βαφή με ψεκασμό, όπου μια σπίθα μπορεί να δημιουργήσει έκρηξη. Ωστόσο, χαμηλή πίεση αέρα στο εσωτερικό του βραχίονα μπορεί να αποτρέψει την είσοδο των εύφλεκτων ατμών καθώς και άλλων ρύπων.

• Οδηγός - μερικά ρομπότ συνδέουν ηλεκτροκινητήρες στις αρθρώσεις μέσω γραναζιών. Άλλα συνδέουν τον κινητήρα στην άρθρωση απευθείας (άμεσος οδηγός). Η χρήση γραναζιών έχει ως αποτέλεσμα μια μετρήσιμη «αντίδραση» η οποία δεν υπάρχει σε έναν άξονα. Μικρότεροι ρομποτικοί βραχίονες χρησιμοποιούν συχνά υψηλής ταχύτητας, χαμηλής ροπής κινητήρες συνεχούς ρεύματος, που απαιτούν συνήθως υψηλότερη αναλογία γρανάζωσης. Αυτό έχει το μειονέκτημα της αντίδρασης. Σε τέτοιες περιπτώσεις η αρμονική καθοδήγηση χρησιμοποιείται συχνά.

• Συμμόρφωση – Αυτή είναι η μέτρηση της γωνίας ή της απόστασης που ένας ρομποτικός άξονας θα μετακινηθεί όταν εφαρμοστεί πάνω του μια μία δύναμη. Λόγω της συμμόρφωσης όταν ένα ρομπότ πηγαίνει σε μία θέση μεταφέροντας το μέγιστο ωφέλιμο φορτίο, θα βρεθεί στην θέση αυτή ελαφρώς πιο αργά απ’ όταν δεν θα φέρει φορτίο. Η συμμόρφωση μπορεί επίσης να είναι υπεύθυνη για υπερβάσεις όταν το ρομπότ είναι φορτωμένο με μεγάλο φορτίο, σ’ αυτήν την περίπτωση η επιτάχυνση θα πρέπει να μειώνεται.

ΣΚΟΠΟΣ

Ο κύριος σκοπός αυτής της πτυχιακής εργασίας είναι η δημιουργία λογισμικού που θα επιτρέπει τον έλεγχο ενός ρομποτικού βραχίονα εκπαιδευτικού μεγέθους.

Συγκεκριμένα ζητείται από το Βραχίονα αφ' ενόν μεν να ελέγχονται με ακρίβεια οι κινήσεις του στον τρισδιάστατο χώρο, αφ΄ετέρου να μεταφέρει με ασφάλεια αντικείμενα από ένα σημείο στο άλλο, μέσα στον χώρο αυτόν, αποφεύγοντας τυχόν εμπόδια στη διαδρομή που κινείται.

Για να επιτευχθεί ο σκοπός αυτός, πρέπει επομένως να επιτυγχάνονται τρεις επί μέρους στόχοι.

Προκειμένου ο βραχίονας να ελεγχθεί ως προς τις μετακινήσεις του στο χώρο:

Για την επίλυση των αποστάσεων μέσω μαθηματικών εξισώσεων προκειμένου να εντοπίζονται οι συντεταγμένες του βραχίονα στον τρισδιάστατο χώρο, υλοποιείται ένας κώδικας, στο Πρόγραμμα Matlab. Βάσει αυτού ο βραχίονας μπορεί να μετακινηθεί σε συγκεκριμένα σημεία μέσα στο χώρο, εκμεταλλευόμενος τριγωνομετρικές εξισώσεις απλοποιώντας το πρόβλημα σε κάθε βήμα.

Προκειμένου να ελεγχθούν χρόνοι, παύσεις, ανίχνευση εμποδίων, κατάλληλες κινήσεις για την ομαλότητα του βραχίονα:

Χρησιμοποιείται ο μικροελεγκτής Arduino, ο οποίος είναι υπεύθυνος για τον τρόπο κίνησης του ρομποτικού βραχίονα. Αυτο σημαίνει οτι το λογισμικό που χρειάζεται να αναπτυχθεί εμπεριέχει τη μέθοδο επικοινωνίας με τον υπολογιστή αλλά και τις κατάλληλες συναρτήσεις για τον ορθό έλεγχο των σερβοκινητήρων του βραχίονα.

Προκειμένου οι κινήσεις να γίνονται με ασφάλεια:

Απαιτείται ανίχνευση των αντικειμένων, που θα μπορούσαν να αποτελέσουν εμπόδιο στις κινήσεις του. Έτσι, χρησιμοποιήται ένα αισθητήριο υπερήχων για τη σάρωση του χώρου στο χρονικό διάστημα, κατά το οποίο χρειάζεται η επίβλεψη αυτού. Οπότε το πρόγραμμα του ελεγκτή μπορεί να πραγματοποιεί δυναμικές αλληλεπιδράσεις με το φάκελο εργασίας φέροντας με ακρίβεια και ασφάλεια τη μεταφορά αντικειμένων ή απλά την κίνησή του μέσα στο τριδιάστατο χώρο.

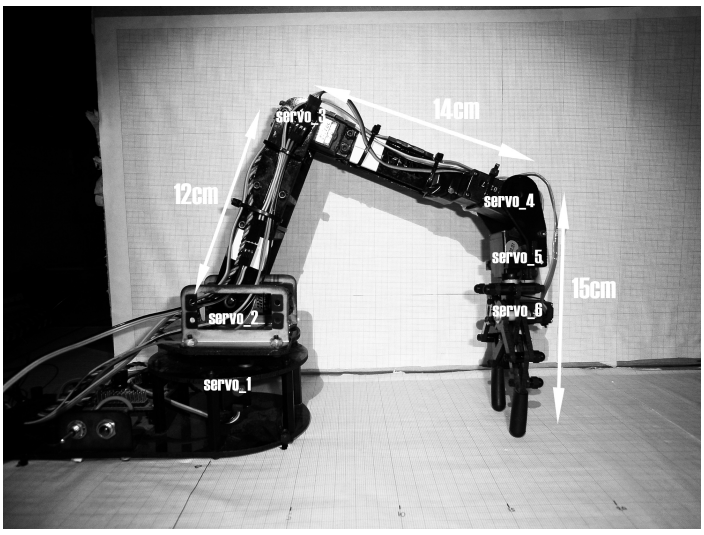
ΜΕΘΟΔΟΛΟΓΙΑ

Περιγραφή στοιχείων και προγραμμάτων

Ρομποτικός βραχίονας

Χρησιμοποιήθηκε ένας Ρομποτικός Βραχίονας από πλαστικό που αποτελείται από 6 servo κινητήρες και έχει 6 βαθμούς ελευθερίας.
Η βάση του Βραχίονα έχει ύψος 8.10 cm.
Το πρώτο στέλεχος έχει μήκος 12 cm, το δεύτερο 14 cm και το τρίτο 15 cm.

Όσον αφορά τους servo κινητήρες, ο κάθε ένας από αυτούς έχει δυνατότητα κλίσης 180 μοιρών με διαφορετική δύναμη λόγω διαφοράς μεγέθους του ενός απο τον άλλον. Τέσσερις από αυτούς παίζουν το ρόλο ‘ώμου , αγκώνα’. Αυτοί οι σερβοκινητήρες έχουν ίδιες προδιαγραφές όσον αφορά τη ροπή, 61 με 76 oz-in [απο 4.8 μεχρι 6 volt αντίστοιχα].
Οι σερβοκινητήρες που έχουν το ρολο ‘καρπού και δαχτύλων’, είναι διαφορετικού τύπου (micro) άρα με διαφορετική άσκηση ροπής 38.8 με 44.4 oz-in. Ο πρώτος μπορεί να περιστρέψει το Βραχίονα σε οριζόντιο επιφάνεια (περιστροφικά).
Οι δεύτερος, τρίτος και τέταρτος καλύπτουν κατακόρυφες επιφάνειες ομοίως 180 μοιρών (πρισματικά).
Ο πέμπτος servo κινητήρας περιστρέφει τον καρπό του Βραχίονα κατά 180 μοίρες και ο έκτος ανοιγοκλείνει τη «δαγκάνα» του έχοντας έτσι τη δυνατότητα «αρπαγής» και μεταφοράς αντικειμένων.



Εικόνα 1

Πρόγραμμα MATLAB

Το Matlab αποτελεί ένα εμπορικό εργαλείο το οποίο προσφέρει ένα διαδραστικό προγραμματιστικό περιβάλλον στον χρήστη και χρησιμοποιείται σε ένα μεγάλο εύρος εφαρμογών. Ενσωματώνει μια υψηλού επιπέδου γλώσσα προγραμματισμού, κατάλληλη για τη μοντελοποίηση και επίλυση σύνθετων μαθηματικών, και όχι μόνο, προβλημάτων. Το όνομά του προέρχεται από τις λέξεις matrix laboratory πράγμα που υποδηλώνει το γεγονός ότι η λειτουργία του βασίζεται εξ ολοκλήρου στη χρήση πινάκων, στοιχεία των οποίων μπορεί να είναι πραγματικοί ή μιγαδικοί αριθμοί. Ανάμεσα σε ένα πλήθος άλλων ευκολιών που προσφέρει επιτρέπει τον εύκολο χειρισμό πινάκων, τη γραφική απεικόνιση (plotting) συναρτήσεων και δεδομένων, την υλοποίηση αλγορίθμων, την δημιουργία γραφικών διεπαφών και τη συνεργασία και διαλειτουργικότητα με προγράμματα γραμμένα σε άλλες γλώσσες προγραμματισμού. Λόγω, δε του ότι το Matlab βρίσκει εφαρμογή σε ποικίλα επιστημονικά πεδία (επεξεργασία σήματος, νευρωνικά δίκτυα, συστήματα ελέγχου), πρόσθετα πακέτα, που ονομάζονται toolboxes, ενσωματώνονται σε αυτό και προσφέρουν χρήσιμες εξειδικευμένες συναρτήσεις.
Δημιουργήθηκε από τον Moler στη δεκαετία του 1970, αρχικά σαν εργαλείο διαχείρισης των βιβλιοθηκών της γλώσσας προγραμματισμού fortran: Linpack (γραμμική άλγεβρα) και Eispack (ιδιοτιμές και ιδιοδιανύσματα). Η έκδοση που χρησιμοποιείται σήμερα είναι η 8 και πέρα από τις δυνατότητες που παρείχαν οι προηγούμενες αυτή έχει εμπλουτιστεί με νέες βιβλιοθήκες αλλά και με καινούριο πιο φιλικό γραφικό περιβάλλον. Το Matlab χρησιμοποιείται κυρίως για μαθηματικούς υπολογισμούς, ανάπτυξη αλγορίθμων, προσομοίωση, μοντελοποίηση και προτυποποίηση. Επίσης, για ανάλυση δεδομένων, οπτικοποίηση , επιστημονικά και μηχανολογικά γραφικά και ανάπτυξη εφαρμογών.
Είναι ένα διαδραστικό σύγχρονο εργαλείο του οποίου βασικό στοιχείο είναι η διάταξη (array). Αυτό μας επιτρέπει τη λύση πολλών υπολογιστικών προβλημάτων, ιδιαίτερα όσων περιλαμβάνουν σχηματισμούς πινάκων και διανυσμάτων σε χρόνο μικρότερο απ’ ότι απαιτούν οι γλώσσες προγραμματισμού Fortran και C.

Τα Πέντε Βασικά μέρη του MATLAB

1. Περιβάλλον ανάπτυξης / Είναι ένα σύνολο από εργαλεία που βοηθάνε στην χρήση των συναρτήσεων και των αρχείων του Matlab .Τα εργαλεία αυτά συμπεριλαμβάνουν την επιφάνεια εργασίας του Matlab (Matlab desktop) ,τη γραμμή εργασιών (command window), το ιστορικό εντολών (command history), έναν κειμενογράφο (editor) και έναν διορθωτή (debugger), έναν περιηγητή για τη βοήθεια, το χώρο εργασίας (workspace), τα αρχεία και την αναζήτηση.

2. Βιβλιοθήκη μαθηματικών συναρτήσεων / Είναι μια τεράστια συλλογή από υπολογιστικούς αλγορίθμους που καλύπτουν ένα ευρύτατο φάσμα που ξεκίνα από στοιχειώδεις συναρτήσεις, όπως το άθροισμα, το ημίτονο, και εκτείνεται σε πιο σύνθετες συναρτήσεις, όπως αντιστροφή πίνακα, συναρτήσεις Bessel και μετασχηματισμούς Fourrier.

3. Γλώσσα / Είναι μια υψηλού επιπέδου γλώσσα με δηλώσεις δομής ελέγχου, δομές δεδομένων, συναρτήσεις/εξόδους και χαρακτηριστικά αντικειμενοστραφούς προγραμματισμού.

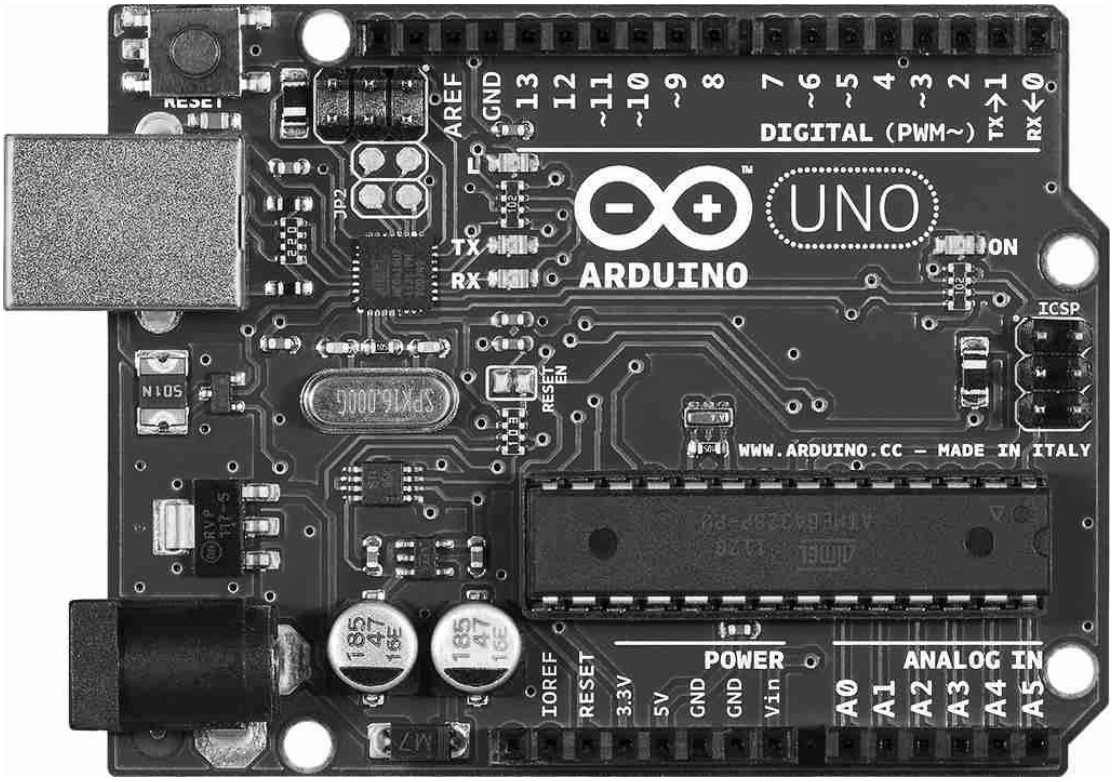
4. Γραφικά / Το Matlab μπορεί με ευκολία να αναπαραστήσει γραφικά διανύσματα και πίνακες και δίνει τη δυνατότητα σχολιασμού και εκτύπωσης αυτών των γραφημάτων. Περιλαμβάνει επιπέδου για απεικόνιση διανυσμάτων σε διαγράμματα δυο και τριών διαστάσεων, επεξεργασία εικόνας, δυναμική κίνηση ομοιωμάτων και γραφικά παρουσίασης. Περιλαμβάνει επιπλέον συναρτήσεις χαμηλού επιπέδου που επιτρέπουν στο χρήστη να προσαρμόσει πλήρως την εμφάνιση των γραφημάτων.

5. Περιβάλλον εφαρμογής προγραμμάτων (Application Program Interface - API) / Είναι μια βιβλιοθήκη που επιτρέπει στο χρήστη να γράψει κώδικα στις γλώσσες προγραμματισμού C και Fortran που αλληλεπιδρούν με το Matlab. Δίνει τη δυνατότητα κλήσης υπορουτίνων από το Matlab (dynamic linking), κλήση του Matlab ως υπολογιστικής μηχανής και για την ανάγνωση και ανάπτυξη αρχείων τύπου MAT (MATfiles).

Εφαρμογές του MATLAB

Το Matlab μπορεί να χρησιμοποιηθεί σε ένα μεγάλο εύρος εφαρμογών που απαιτούν σύνθετους μαθηματικούς υπολογισμούς. Είναι προσανατολισμένο για να λύνει τα προβλήματα αριθμητικά, δηλαδή με αριθμητική πεπερασμένης ακρίβειας. Χρησιμοποιείται για την ανάπτυξη αλγορίθμων που χρησιμοποιούνται για την επίλυση μαθηματικών προβλημάτων. Για την ανάπτυξη ενός αλγορίθμου απαιτείται ο ορισμός και η πλήρης κατανόηση του προβλήματος προς επίλυση, ο καθορισμός του, δηλαδή ο καθορισμός μιας συγκεκριμένης διαδικασίας βημάτων τα οποία οδηγούν σε συγκεκριμένα αποτελέσματα με ικανοποιητικά ελάχιστο δυνατό κόστος (μνήμης και χρόνου),η υλοποίηση του σε υπορουτίνα / πρόγραμμα, αν αυτό κριθεί απαραίτητο. Επίσης απαιτείται ο έλεγχος της υπορουτίνας προγράμματος για διάφορες περιπτώσεις δεδομένων, ειδικά όλων των ακραίων περιπτώσεων και τέλος η χρήση υπορουτίνας / προγράμματος για επίλυση σχετικών προβλημάτων. Μια άλλη εφαρμογή του Matlab όπως αναφέρθηκε στην προηγούμενη παράγραφο είναι η μοντελοποίηση και η προσομοίωση προβλημάτων, κάνοντας χρήση του Simulink, το οποίο είναι ένα πρόγραμμα ενσωματωμένο στο Matlab, η ανάλυση και η οπτικοποίηση δεδομένων και η δημιουργία γραφικών παραστάσεων. Το Matlab διαθέτει δυναμικές και ευέλικτες γραφικές ικανότητές. Σχήματα διαφόρων τύπων μπορούν να δημιουργηθούν σχετικά εύκολα. Τέλος η δημιουργία εφαρμογών για την επεξεργασία συστημάτων και την επίλυση σύνθετων προβλημάτων Τα toolboxes (εργαλειοθήκες) του Matlab είναι ένα ευρύ σύνολο από ήδη ανεπτυγμένους αλγορίθμους που παρέχονται στο χρήστη με τη μορφή συναρτήσεων του προγράμματος και βοηθούν στη λύση ειδικής κατηγορίας προβλημάτων. Κάποια έτοιμα πακέτα τέτοιων αλγορίθμων είναι το control systems toolbox,το signal processing toolbox,το simulink, το model predictive control toolbox,κτλ.

Μικροελεγκτής ARDUINO



Εικόνα 2

Ο 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 διαβάσει το περιβάλλον λαμβάνοντας σήματα από διάφορους αισθητήρες και μπορεί να το επηρεάσει ελέγχοντας φώτα, μηχανισμούς και άλλα. Ο μικροελεγκτής που βρίσκεται σε κάθε πλακέτα arduino μπορεί να προγραμματιστεί σε γλώσσα arduino programming language και το προγραμματιστικό περιβάλλον του arduino είναι βασισμένο στη γλώσσα Processing. Οι arduino εφαρμογές μπορούν να λειτουργούν μόνες τους ή μπορούν να επικοινωνούν με διάφορα προγράμματα σε ένα υπολογιστή.

Επιπλέον, η γλώσσα προγραμματισμού, οι διάφορες βιβλιοθήκες και το ολοκληρωμένο περιβάλλον ανάπτυξης που υπάρχουν για τον προγραμματισμό της πλατφόρμας Arduino αποτελούν ανοιχτό λογισμικό.

Μία πλακέτα Arduino αποτελείται από έναν cmos 8 ψηφίων Atmel AVR μικροελεγκτή 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.

Ο μικροελεγκτής σε κάθε πλατφόρμα Arduino είναι προγραμματισμένος και περιέχει έναν ειδικό φορτωτή εκκίνησης ο οποίος απλοποιεί κυρίως την διαδικασία της τοποθέτησης προγραμμάτων στην ενσωματωμένη μνήμη Flash της πλατφόρμας, με αποτέλεσμα να μην είναι απαραίτητη κάποια εξωτερική συσκευή προγραμματισμού μικροελεγκτών AVR. Οι πιο πρόσφατες πλατφόρμες Arduino μπορούν να προγραμματιστούν μέσω της θύρας USB που διαθέτουν, με την βοήθεια όμως ενός κυκλώματος μετατροπής από USB σε σειριακή όπως το FTDI FT232.

Γενικά όλες οι πλακέτες είναι προγραμματισμένες μέσω μιας σειριακής σύνδεσης RS-232, αλλά ο τρόπος με τον οποίο αυτό υλοποιείται ποικίλλει ανάλογα με την έκδοση. Οι σειριακές πλακέτες Arduino περιέχουν ένα απλό κύκλωμα αντιστροφής για την μετατροπή ανάμεσα στα σήματα των επιπέδων RS-232 και TTL.

Ο μικροελεκτής ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

• 2Kb μνήμης SRAM που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά μας για να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή αν γίνει reset.

• 1Kb μνήμης EEPROM η οποία μπορεί να χρησιμοποιηθεί για «ωμή» εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματά μας κατά το runtime. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset οπότε είναι το ανάλογο του σκληρού δίσκου.

• 32Kb μνήμης Flash, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των δικών μας προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή μας. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset.

Το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία του Arduino με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.

Επιπλέον, στην πάνω πλευρά του Arduino βρίσκονται 14 θηλυκά pin, αριθμημένα από 0 ως 13, που μπορούν να λειτουργήσουν ως ψηφιακές είσοδοι και έξοδοι. Λειτουργούν στα 5V και καθένα μπορεί να παρέχει ή να δεχτεί το πολύ 40 mA ρεύμα. Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές είσοδοι/έξοδοι έχουν και δεύτερη λειτουργία. Συγκεκριμένα:

Τα pin 0 και 1 λειτουργούν ως RX και TX της σειριακής όταν το πρόγραμμά μας ενεργοποιεί την σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμά μας στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά μας ενεργοποιήσουμε το σειριακό interface, χάνουμε 2 ψηφιακές εισόδους/εξόδους.

• Τα pin 2 και 3 λειτουργούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Συγκεκριμένα μπορούμε να τα ρυθμίσουμε μέσα από το πρόγραμμά μας ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

• Τα pin 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορούμε να συνδέσουμε λόγου χάρη ένα LED σε κάποιο από αυτά τα pin και να ελέγξουμε πλήρως την φωτεινότητά του με ανάλυση 8bit (256 καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο) αντί να έχουμε απλά την δυνατότητα αναμμένο-σβηστό που παρέχουν οι υπόλοιπές ψηφιακές έξοδοι. Είναι σημαντικό να καταλάβουμε ότι το PWM δεν είναι πραγματικά αναλογικό σύστημα και ότι θέτοντας στην έξοδο την τιμή 127, δεν σημαίνει ότι η έξοδος θα δίνει 2.5V αντί της κανονικής τιμής των 5V, αλλά ότι θα δίνει ένα παλμό που θα εναλλάσσεται με μεγάλη συχνότητα και για ίσους χρόνους μεταξύ των τιμών 0 και 5V.

Στην κάτω πλευρά του Arduino, με τη σήμανση ANALOG IN, θα βρούμε μια ακόμη σειρά από 6 pin, αριθμημένα από το 0 ως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στον μικροελεγκτή. Για παράδειγμα, μπορούμε να τροφοδοτήσουμε ένα από αυτά με μια τάση την οποία μπορεί να κυμαίνεται με ένα ποτενσιόμετρο από 0V ως μια τάση αναφοράς Vref η οποία, αν δεν κάνουμε κάποια αλλαγή είναι προ ρυθμισμένη στα 5V. Τότε, μέσα από το πρόγραμμά μας μπορούμε να «διαβάσουμε» την τιμή του pin ως ένα ακέραιο αριθμό ανάλυσης 10-bit, από 0 (όταν η τάση στο pin είναι 0V) μέχρι 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V, ή σε όποια τάση επιθυμείτε (μεταξύ 2 και 5V) τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας.

Το Arduino μπορεί να τροφοδοτηθεί με ρεύμα είτε από τον υπολογιστή μέσω της σύνδεσης USB, είτε από εξωτερική τροφοδοσία που παρέχεται μέσω μιας υποδοχής φις των 2.1mm (θετικός πόλος στο κέντρο) και βρίσκεται στην κάτω αριστερή γωνία του Arduino. Η τάση λειτουργίας της πλατφόρμα Arduino είναι 5V.

Για να μην υπάρχουν προβλήματα, η εξωτερική τροφοδοσία πρέπει να είναι από 7 ως 12V και μπορεί να προέρχεται από ένα κοινό μετασχηματιστή του εμπορίου, από μπαταρίες ή οποιαδήποτε άλλη πηγή DC. Δίπλα από τα pin αναλογικής εισόδου, υπάρχει μια ακόμα συστοιχία από 6 pin με την σήμανση POWER.

Η λειτουργία του καθενός έχει ως εξής:

• Το πρώτο, με την ένδειξη RESET, όταν γειωθεί (σε οποιοδήποτε από τα 3 pin με την ένδειξη GND που υπάρχουν στο Arduino) έχει ως αποτέλεσμα την επανεκκίνηση του Arduino.

• Το δεύτερο, με την ένδειξη 3.3V, μπορεί να τροφοδοτήσει τα εξαρτήματά μας με τάση 3.3V. Η τάση αυτή δεν προέρχεται από την εξωτερική τροφοδοσία αλλά παράγεται από τον ελεγκτή Serial-over- USB και έτσι η μέγιστη ένταση που μπορεί να παρέχει είναι μόλις 50mA.

• Το τρίτο, με την ένδειξη 5V, μπορεί να τροφοδοτήσει τα εξαρτήματά μας με τάση 5V. Ανάλογα με τον τρόπο τροφοδοσίας του ίδιου του Arduino, η τάση αυτή προέρχεται είτε άμεσα από την θύρα USB (που ούτως ή άλλως λειτουργεί στα 5V), είτε από την εξωτερική τροφοδοσία αφού αυτή περάσει από ένα ρυθμιστή τάσης για να την «φέρει» στα 5V.

• Το τέταρτο και το πέμπτο pin, με την ένδειξη GND, είναι φυσικά γειώσεις.

• Το έκτο και τελευταίο pin, με την ένδειξη Vin έχει διπλό ρόλο. Σε συνδυασμό με το pin γείωσης δίπλα του, μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας του Arduino, στην περίπτωση που δεν μας βολεύει να χρησιμοποιήσουμε την υποδοχή του φις των 2.1mm. Αν όμως έχουμε ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, μπορούμε νε χρησιμοποιήσουμε αυτό το pin για να τροφοδοτήσουμε εξαρτήματα με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

Το IDE του Arduino είναι γραμμένο σε Java και μπορεί να τρέξει σε πολλαπλές πλατφόρμες. Περιλαμβάνει επεξεργαστή κώδικα (επεξεργαστή κειμένου με διάφορα εύχρηστα εργαλεία) και μεταγλωττιστή, και έχει την ικανότητα να φορτώνει εύκολα το πρόγραμμα μέσω σειριακής θύρας από τον υπολογιστή στην πλακέτα.

Το περιβάλλον ανάπτυξης είναι βασισμένο στην Processing, ένα περιβάλλον ανάπτυξης σχεδιασμένο να εισαγάγει στον προγραμματισμό νέους χρήστες μη εξοικειωμένους με την ανάπτυξη λογισμικού. Η συγκεκριμένη γλώσσα προγραμματισμού προέρχεται από την Wiring, μια γλώσσα που μοιάζει με την C η οποία παρέχει παρόμοια λειτουργικότητα για μια πιο περιορισμένης σχεδίασης πλακέτα, της οποίας το περιβάλλον ανάπτυξης βασίζεται επίσης στην Processing.

Στην πραγματικότητα, το ολοκληρωμένο περιβάλλον ανάπτυξης είναι αυτό το οποίο κατά την μεταγλώττιση αποθηκεύει σε ένα προσωρινό αρχείο τον πηγαίο κώδικα του προγράμματος που έχουμε συγγράψει και στην συνέχεια προσθέτει σε αυτό όλα όσα είναι απαραίτητα (πρωτότυπα συναρτήσεων, υλοποίηση συνάρτησης main(), ενσωμάτωση επικεφαλίδων) ώστε να είναι δυνατή η μεταγλώττιση του.

Η μεταφορά των δεδομένων του τελικού αρχείου στην πλατφόρμα Arduino γίνεται μέσω USB. Για την επίτευξη αυτής της διαδικασίας χρησιμοποιείται από το ολοκληρωμένο περιβάλλον ανάπτυξης το εργαλείο avrdude και ο φορτωτής εκκίνησης της πλατφόρμας Arduino.

Στην προκειμένη περίπτωση η χρηήση του μικροελεγκτή χρησιμοποιήθηκε για τη σειριακή επικοινωνία με το Matlab και ενσύρματη σύνδεση με τους σερβοκινητήρες.

Αισθητήριο υπερήχων (ultra sonic sensor)

Ο Αισθητήρας Υπερήχων HC-SR04, που χρησιμ οποιήθηκε, μπορεί να υπολογίσει απόσταση από 2cm έως 400cm με ακρίβεια ενός εκατοστού.
Όπως βλέπουμε στην εικόνα (3) υπάρχει ενα trigger και ενα echo. Το trigger παράγει υπερήχους και το echo δέχεται υπερήχους. Οπότε, ο απαιτούμενος χρόνος που απαιτείται για να γυρίσει το σήμα από τον πομπό στο δέκτη μετριέται και μετασχηματίζεται σε απόσταση.

Χρησιμοποίση παλμού ο οποίος να έχει τουλάχιστον 10us διάρκεια.

Το εξάρτημα αυτόματα στέλνει 8 παλμπούς συχνότητας 40kHz και ανιχνέυει αν

υπάρχει παλμός που επέστρεψε.

Η απόσταση λαμβάνεται υπόψιν σύμφωνα με το χρόνο του σηματός στη αποστολή

και λήψη του σήματος .Εξαρτάται βεβαίως απο την ταχύτητα που ήχου που ανέρχεται

στα 340 m/s.Η τάση λειτοργίας του εξαρτήματος είναι 5V και παρέχεται απο το μικροελεγκτή.Το ρεύμα λειτουργίας είναι 15mA και η συχνότητα λειτουργίας είναι 40 Hz.Η γωνία μέτρησης που επιτυγχάνει είναι οι 15 μοίρες.

Εικόνα 3

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΜΕΘΟΔΟΙ

Denavit-Hartenberg

- Θεωρία

Η μέθοδος των Denavit-Hartenberg (D&H) αναπτύχθηκε για την περιγραφή της κινηματικής σύνθετων μηχανισμών απο ένα ελάχιστο σύνολο παραμέτρων. Αργότερα προσαρμόσθηκε στην περιγραφή της κινηματικής ρομποτικών μηχανισμών. Η μέθοδος D&H παράγει μια συστηματική παραγωγή πινάκων Τ που συνδέουν τη θέση και προσανατολισμό ενός συνδέσμου ως τον προηγούμενο.

Βήματα εφαρμογής D-H :

1. Αριθμούμε τις αρθρώσεις απο 1 έως n αρχίζοντας απο τη βάση και τελειώνουμε στο εργαλείο (yaw , pitch , roll) με αυτή τη σειρά.
2. Τοποθετούμε το σύστημα συντεταγμένων L_0 στη βάση του ρομπότ.
   Έχουμε τον άξονα z_0 ευθυγραμμισμένο με τον άξονα της κίνησης της άρθρωσης 1 (περιστροφική ή πρισματική), τοποθετόντας κ=1.
3. Ευθυγραμμίζουμε τον $z_κ$ με τον άξονα της κίνησης (περιστροφική ή πρισματική) της άρθρωσης κ+1.
4. Καθορίζουμε το σημείο αναφοράς του $L_κ$ με την διχοτόμηση του $z_κ$ και $z_{κ-1}$ και αν είναι παράλληλοι χρησιμοποιούμε μια κοινή κάθετη στον $z_κ$, $z_{κ-1}$.
5. Επιλέγουμε $x_κ$ που είναι κάθετο στον $z_κ$, $z_{κ-1}$. Αν οι $z_κ$, $z_{κ-1}$ είναι παράλληλοι, σημειώνουμε $x_κ$ πέρα από τον $z_{κ-1}$.
6. Επιλέγουμε $y_κ$ σύμφωνα με το δεξί μας χέρι (z αντίχειρας, x δείκτης, y μέσος). Βάζουμε κ = κ + 1. Αν κ < n πηγαίνουμε στο βήμα 2 αλλιώς συνεχίζουμε.
7. Καθορίζουμε τον προσανατολισμό του L_n στο εργαλείο. Ευθυγραμμίζουμε τον z_n με προσεγγιστικό διάνυσμα κύλισης (approach vector roll), y_n με το διάνυσμα ανύψωσης (sliding vector pitch) και x_n με κανονικό διάνυσμα στροφής (normal vector yaw). Βάζουμε κ = 1.
8. Τοποθετούμε bκ στη διχοτόμηση των $x_κ$ και $z_{κ-1}$.

9. Υπολογίζουμε θκ τη γωνία περιστροφής και $z_{κ-1}$ ώστε ο $x_{κ-1}$ να γίνει παράλληλος του xκ.
10. Υπολογίζουμε dκ την απόσταση του $L_{κ-1}$ από το σημείο dκ bκ κατά μήκος του zκ-1.
11. Υπολογίζουμε ακ την απόσταση του bκ απο το σύστημα $L_κ$ κατά μήκος του xκ.
12. Υπολογίζουμε ακ τη γωνία περιστροφής του xκ, ώστε ο $z_{κ-1}$ να γίνει παράλληλος του $z_κ$. Βάζουμε κ = κ + 1. Αν κ <= n, πηγαίνουμε στο βήμα 8, διαφορετικά σταματάμε. Για να βρούμε τον πίνακα του βραχίονα με τα βήματα (μέθοδο) D-H, λαμβάνουμε υπόψην μας το σύστημα των συντεταγμένων του κάθε ρομπότ. Βρίσκουμε τα στοιχεία-παραμέτρους (όπως λ.χ. τους βαθμούς ελευθερίας, συντεταγμένες κ.λ.π.), στη συνέχεια τον πίνακα του βραχίονα και τέλος την εξίσωση του βραχίονα.

- Πράξη

Ο ομογενής πίνακας T_0 που καθορίζει τη θέση των κ (= 6) συστήματος συντεταγμένων που βγάζουμε από τον αλγόριθμο D-H, με εναφορά στο σύστημα συντεταγμένων της βάσης I_0 είναι ο διαδοχικός πολλαπλασιασμός των πινάκων $T^κ{}_{κ-1}$, αυτός εκφράζεται ως εξής:

$$T_0^κ = T_0{}^1\ T_1{}^2\ T_2{}^3 T_3{}^4 T_4{}^5 T_5{}^6$$

Τελικά μετά τον πολλαπλασιασμό των πινάκων καταλήγουμε στην παρακάτω μορφή της εξίσωσης που ονομάζεται εξίσωση βραχίονα.

$$T_0^κ = T_{tool}^{base} = \begin{vmatrix} & R(q) & & . & P(q) \\ | & ... & ... & ... & . & ... & | \\ | & 0 & 0 & 0 & . & 1 & | \end{vmatrix}$$

Ο υποπίνακας R(q), είναι διαστασεων n x n και καθορίζει τον προσανατολισμό του βραχίονα, ενώ ο πάνω δεξιά n x P(q) καθορίζει τη θέση του εργαλείου. Οι n στήλες του R(q) δείχνουν την κατεύθυνση των τριών διανυσμάτων (vector) του εργαλείου σε σχέση με τη βάση ενώ τα n στοιχεία του P(q) δείχνουν τη θέση του εργαλείου με αναφορά τη βάση του βραχίονα.

Jacobian

- Θεωρία

Ένας ρομποτικός βραχίονας με έξι βαθμούς ελευθερίας μπορει να τοποθετήσει το άκρο του σε οποιοδήποτε σημείο του χώρου με οποιοδήποτε προσανατολισμό και ένας βραχίονας με τρείς βαθμούς μπορεί να το τοποθετήσει σε οποιοδήποτε σημείο του επιπέδου. Σε περιπτώσεις επιπέδου ρομποτικού βραχίονα με τρείς βαθμούς ελευθερίας η διαφορική κινηματική εξίσωση έχει μοναδική λύση όταν η διάταξη του ρομπότ είναι μη ιδιόμορφη. Εάν όμως το ρομπότ έχει περισότερους βαθμούς ελευθερίας απο τρείς (πλεονάζοντες βαθμοί ελευθερίας) τότε υπάρχει απειρία λύσεων που δίνουν τις ίδιες συντεταγμένες στο τελικό στοιχείο δράσης.

Η μήτρα J ονομάζεται Ιακωβιανή μήτρα του ρομπότ ή απλα Ιακωβιανή και για κάθε διάταξη των αρθρώσεων παριστά τη διαφορική σχέση ανάμεσα στις μετατοπίσεις των αρθρώσεων και την παρούσα θέση του τελικού στοιχείου δράσης.
Τα στοιχεία J είναι συναρτήσεις των μετατοπίσεων των αρθρώσεων και συνεπώς μεταβάλλονται με τη μορφή του ρομπότ.

Υπολογισμός Ιακωβιανής μήτρας σε μια απειροστή μεταφορά και περιστροφή του τελικού στοιχείου δράσης. Έστω dx_1 διάνυσμα της απειροστής μεταφοράς και $d\varphi_1$ το 3 x 1 διάνυσμα της απειροστής περιστροφής του τελικού στοιχείου δράσης, ως προς το σύστημα συντεταγμένων $O_0\, x_0\, y_0\, z_0$ της βάσης. Για ευκολία ορίζουμε το 6 x 1 διάυσμα απειροστής μετατόπισης dp:

$$dp = \begin{vmatrix} dx_T \\ d\varphi_T \end{vmatrix}$$

Διαιρώντας τα δύο μέλη της με dt παίρνουμε:

$$p = \begin{vmatrix} v_T \\ \omega_T \end{vmatrix}$$

Όπου v_T και ω_T είναι αντίστοιχα η γραμμική ταχύτητα και η γωνιακή ταχύτητα του τελικού στοιχείου δράσης.

$$p = J\,q$$

όπου $q = [q_1 , q_2 .. q_n]^T$ είναι το n x 1 διάνυσμα των αρθρώσεων. Οι πρώτες τρείς γραμμές της 6 x n Ιακωβιανής μήτρας J αντιστοιχούν στην γραμμική ταχύτητα v_T και οι υπόλοιπες τρείς στην γωνιακή ω_T του τελικού στοιχείου δράσης. Κάθε στήλη της J παριστά τη γραμμική και γωνιακή ταχύτητα που προκαλείται απο την αντίστοιχη άρθρωση. Έστω J_{Li} και J_{Ai} τα 3 x 1 διανύσματα στήλης της J που αντιχστοιχούν στις γραμμικές και γωνιακές ταχύτητες, αντίστοιχα. Τότε έχουμε την ακόλουθη παράσταση της J :

$$J = \begin{vmatrix} J_{L1} , J_{L2} .. J_{Ln} \\ J_{A1} , J_{A2} .. J_{An} \end{vmatrix}$$

Έτσι η γραμμική ταχύτητα v_T του τελικού στοιχείου δράσης γράφεται :

$$v_T = J_{L1}\,q_1 + J_{L2}\,q_2 + .. J_{Ln}\,q_n$$

Εάν η άρθρωση i είναι πρισματική, προκαλεί μια γραμμική ταχύτητα του τελκιού στοιχείου δράσης κατά τη διεύθυνση του άξονα της άρθρωσης. Εάν b_{i-1} είναι το μοναδιαίο διάνυσμα κατα μήκους του άξονα της άρθρωσης i και d_i η βαθμωτή γραμμική ταχύτητα της πρισματικής άρθρωσης, τότε έχουμε :

$$J_{Li}\,q_i = b_{i-1}\,d_i$$

γιατί ως γνωστό, στο συμβολισμό D-H η ταχύτητα d_i μετρείται κατά μήκος του άξονα z_{i-1}. Εάν η άρθρωση είναι στροφική τότε περιστρέφει όλο το σύστημα των συνδέσμων απο το σύνδεσμο i μεχρι το n με μια γωνιακή ταχύτητα :

$$\omega_i = b_{i-1}\,\theta_i$$

Η γωνιακή ταχύτητα προκαλεί μια αντίστοιχα γραμμική ταχύτητα στο τελικό στοιχείο δράσης. Έστω $r_{i-1,T}$ το διάνυσμα θέσης του τελικού στοιχείου δράσης απο το σημείο O_{i-1}. Τότε η γραμμική ταχύτητα που οφείλεται στη γωνιακή ταχύτητα ω_i είναι ίση με :

$$J_{Li}\, q_i = \omega_i \times r_{i-1,T} = (b_{i-1} \times r_{i-1,T})\, \theta_i$$

όπου "x" συμβολίζει το διανυσματικό γινόμενο. Συνεπώς η γραμμική ταχύτητα του τελικού στοιχείου δράσης υπολογίζεται απο την : $J_{Li}\, q_i = b_{i-1}\, d_i$ εάν η άρθρωση που την προκαλεί είναι πρισματική, ή απο την : $J_{Li}\, q_i = \omega_i \times r_{i-1,T} = (b_{i-1} \times r_{i-1,T})\, \theta_i$ εάν η άρθρωση είναι στροφική. Η αντίστοιχ εξίσωση της : $v_T = J_{L1}\, q_1 + J_{L2}\, q_2 + .. J_{Ln}\, q_n$
για την γωνιακή ταχύτητα ω_T του τελικού στοιχείου δρασης είναι :

$$\omega_T = J_{A1}\, q_1 \, , J_{A2}\, q_2 \, .. \, J_{An}\, q_n$$

Εάν η άρθρωση είναι πρισματική, δεν προκαλεί καμία γωνιακή ταχύτητα στο τελικό στοιχείο δράσης, δηλαδή :

$$J_{A1}\, q_1 = 0$$

Τέλος, εάν η άρθρωση είναι στροφική, η γωνιακή ταχύτητα που προκαλείται δίνεται απο την σχέση :

$$J_{Ai}\, q_i = \omega_i = b_{i-1}\, \theta_i$$

Συνοψίζοντας, οι στήλες της Ιακωβιανής δίνονται απο τη σχέση :

$$\begin{vmatrix} J_{Li} \\ J_{Ai} \end{vmatrix} = \begin{vmatrix} b_{i-1} \\ 0 \end{vmatrix}$$ για πρισματική αάρθρωση

και τη σχέση :

$$\begin{vmatrix} J_{Li} \\ J_{Ai} \end{vmatrix} = \begin{vmatrix} b_{i-1} \times r_{i-1,T} \\ b_{i-1} \end{vmatrix}$$ για στροφική άρθρωση

Το διανύσματα b_{i-1} και $r_{i-1,T}$ είναι συναρτήσεις των μετατοπίσεων των αρθρώσεων και μπορούν να υπολογισθούν με τη βοήθεια των μετασχηματισμών συντεταγμένων. Η διεύθυνση του άξονα i - 1 ως προς το πλαίσιο i - 1 παρίσταται με το διάνυσμα $b = [0\ 0\ 1]^T$, γιατί ο άξονας αυτός έχει τη διεύθυνση του άξονα zi_{-1}. Το διάνυσμα αυτό b μπορεί να μετασχηματισθεί σε ένα διάνυσμα ορισμένο ως προς το πλαίσιο της βάσης, δηλαδή b_{i-1}, με τη βοήθεια 3 x 3 μητρών περιστροφής $R^{j-1}_{j}\ (q_j)$. Ητοι :

$$bi\text{-}1 = R^{0}_{1}\ (q_1)\ ..\ R^{i-2}_{i-1}\ (q_{i-1})\ b$$

Το διάνυσμα της θέσης μπορεί να υπολογισθεί με τη βοήθεια των 4 x 4 μητρών $A^{j-1}_{j}\ (q_j)$. Έστω $X_{i-1,T}$ το 4 x 1 αυξημένο διάνυσμα r και $X = [0\ 0\ 1]^T$ το αυξημένο διάνυσμα θέσης που παριστά την αρχή του συστήματος συντεταγμένων. Τότε το $r_{i-1,T}$ μπορεί να βρεθεί από την σχέση :

$$X_{i-1,T} = A^{0}_{1}\ (q_1)\ ..\ A^{n-1}_{n}\ (q_n)\ X - A^{0}_{1}\ (q_1)\ ..\ A^{i-2}_{n-1}\ (q_{i-1})\ X$$

όπου ο πρώτος όρος αντιστοιχέι στο διάνυσμα θέσης απο την αρχή O_0 μέχρι το τελικό στοιχείο δράσης και ο δεύτερος απο το O_0 μέχρι το O_{i-1}.

- Πράξη

Η σχέση p = Jq μας δίνει την μεταφορική και γωνιακή ταχύτητα του τελικού στοιχείου δράσης σαν γραμμική συνάρτηση των ταχυτήτων των αρθρώσεων. Στην πράξη χρειαζόμαστε να υπολογίζουμε τις ταχύτητες q_i των αρθρώσεων που οδηγουν σε επιθυμητή μεταφορική και γωνιακή ταχύτητα το τελικό στοιχείο δράσης του ρομπότ. Προς τούτο πρέπει να επιλύσουμε την εξίσωση ως προς q. Στην προκειμένη περίπτωση η λύση για έξι βαθμούς ελευθερίας q_1, q_2 .. q_6, στα οποία η Ιακωβιανή μήτρα είναι τετραγωνική διαστάσεων 6x6. Εάν δε η J είναι και ομαλή (αντιστρεψιμη) τότε η λύση της ως προς q είναι : $q = J^{-1}p$, η οποία μας δίνει τις ταχύτητες που χρειάζονται οι αρθρώσεις για να πάρουμε μια δεδομένη ταχύτητα (γραμμική/γωνιακή) p στο τελικό στοιχείο δράσης και είναι γνωστη ως σχήμα επιλιμένης ταχυτητας ή σχήμα ελέγχου ταχύτητας. Επειδή η Ιακωβιανή μήτρα μεταβάλλεται με τη διάταξη του ρομποτικού βραχίονα, είναι πιθανό να υπάρξουν διατάξεις για τις οποίες είναι μη αναστρέψιμη. Σε αυτές τις περιπτώσεις η αντίστροφη Ιακωβιανή J^{-1} δεν υπάρχει και η εξίσωση p = Jq δεν έχει λύση.

Ο πίνακας J θα έχει τη μορφή :

$$J = \begin{vmatrix} .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \end{vmatrix}$$

Χρησιμοποιήθηκε μια οριζόντια επιφάνεια, όπου τοποθετήθηκε ο Βραχίονας. Επειδή το ύψος της βάσης ειναι συγκεκριμένο, οι μετρήσεις που ακολούθησαν, αυτό το ύψος το έλαβαν υπόψιν.

Η δημιουργία του κώδικα διακρίνεται σε δυο επί μέρους: τη forward kinematics και την inverse kinematics.

Σε αυτό το σημείο φαίνεται η **δομή του κώδικα** του μικροελεγκτή σε απλοϊκή μορφή για την άμεση κατανόησή του.

Αρχή Προγράμματος

Δήλωση μεταβλητών
|
Εισαγωγή δεδομένων
|
Διαμέτρηση τιμών
|
Έναρξη κίνησης βραχίονα
|
Αρχικό σημείο
|
Αρπαγή αντικειμένου
|
Αρχικοποίηση θέσης
|
Τοποθέτηση αντικειμένου
|
Αρχικοποίηση θέσης _

Κλάσεις
|
blockP() // Παύση κίνησης
|
blockD() // Αναμονή για αρπαγή αντικειμένου από τη θέση ορισμού
|
blockX() // Αναμονή για τοποθέτηση αντικειμένου στη θέση ορισμού_

Τέλος Προγράμματος

FORWARD KINEMATICS

Σύμφωνα με τη **forward kinematics**, ως είσοδος κάθε φορά είναι οι γωνίες των κινητήρων. Ειδικότερα, η συνολική είσοδος του Matlab για κάθε περίπτωση μετακίνησης αντικειμένου είναι οι κλίσεις των σερβοκινητήρων, δηλαδή τέσεροις αριθμοί που αντιστοιχούν σε κάθε έναν σερβοκινητήρα. Το επόμενο βήμα είναι ο υπολογισμός των τρισδιάστατων συντεταγμένων (δύο διαφορετικές στην περίπτωση αυτή) και αργότερα ακολουθεί η σύνδεση με τον μικροελεγκτή όπου δεν διαφέρει η μεθοδολογια της **inverse kinematics** λόγω οτι τα δεδομένα που στέλνει κάθε φορά είναι τα στοιχεία που αντιστοιχούν στις κλίσεις των σερβοκινητήρων και στην άλλη περίπτωση.
Προκειμένου να υπολογιστούν τα αρχικά και τελικά σημεία της κίνησης, χρησιμοποιήθηκαν τριγωνομετρικές εξισώσεις δεδομένου, ότι σε κάθε στάση ο Βραχίονας με τις αποστάσεις δημιουργούν τρίγωνα, των οποίων αρχικά είναι γνωστές μόνον οι γωνίες των κινητήρων και τα γεωμετρικά μήκη του βραχίονα. Βάσει αυτών υπολογίζονται μέσω του Matlab και τα υπόλοιπα στοιχεία των τριγώνων καθώς και το σημείο κατάληξης του άκρου του Βραχίονα δηλαδή τα σημεία συντεταγμένων (x,y,z).

Πρόγραμμα Forward Kinematics (Matlab)

```
clear all                                   % reset
clc

arm1 = 12;                                  % arms length
arm2 = 14;
arm3 = 15;

GA = 8.1;                                   % ground distance

deg1 = 'type degrees of 1: ';
deg2 = 'type degrees of 2: ';
deg3 = 'type degrees of 3: ';
deg4 = 'type degrees of 4: ';
deg5 = 'type degrees of 5: ';
deg6 = 'type degrees of 6: ';
d1 = input(deg1)
d2 = input(deg2)
d3 = input(deg3)
d4 = input(deg4)
d5 = input(deg5)
d6 = input(deg6)
```

Ορισμός των μεγεθών άκρων του βραχίονα, το ύψος της βάσης και εισαγωγή επιθυμητών κλίσεων που επιλέγουμε για την αρχή της επεξεργασίας αυτών των τιμών στον παρακάτω κώδικα.

```
D3 = d3 - 90 + d2;                          % degree calibration
D4 = d4 + D3 - 90;
x1 = arm1 * cosd(d2);                       % 2D calculation
h1 = arm1 * sind(d2);
x2 = arm2 * cosd(D3);
h2 = arm2 * sind(D3);
x3 = arm3 * cosd(D4);
h3 = arm3 * sind(D4);
x = x1 + x2 + x3;
h = h1 + h2 + h3 + GA;
```

Εύρεση ύψους και απόστασης απο το κεντρικό σημείο βάσης. Σε αυτό το σημείο παίρνουμε υπόψιν όλες τις κλίσεις των σερβοκινητήρων πλην ενός, ο οποίος περιστρέφει το βραχίονα

στο οριζόντιο επίπεδο.

```
if (d1==90)                                  %  Y = 0

    X = x;
    H = h;
    Y = 0;

elseif (d1<90)                              % positive Y

    a = x;
    B = d1;
    b = a * sind(B);
    c = a * cosd(b);

    X = b;
    H = h;
    Y = c;

else (d1>90)                                % negative Y

    a = x;
    B = 180 - d1;
    b = a * sind(B);
    c = a * cosd(B);

    X = b;
    H = h;
    Y = 0 - c;
end
```

Εύρεση τελικού σημείου απόληξης (x,y,z) μέσω τριγωνομετρικών συναρτήσεων.
Στην $1^{η}$ περίπτωση η βάση είναι κεντραρισμένη (90 μοίρες), στη $2^{η}$ περίπτωση η κλίση είναι μικρότερη από 90 μοίρες και στην $3^{η}$ περίπτωση η κλίση είναι μεταξύ των 90 και180 μοιρών. Όταν δεν είναι κεντραρισμένη η βάση, χρησιμοποιείται η ακτίνα, που δηλώνει την απόσταση κέντρου και τελικού σημείου, με σκοπό να βρεθεί η 3η συντεταγμένη που μας είναι άγνωστη. Αλλιώς η συντεταγμένη ισούται με μηδέν.

```
P = [d1 , d2 , D3 , D4];

arduino=serial('/dev/cu.usbmodemfa131','BaudRate',9600);

fopen(arduino);
pause(2);

fwrite (arduino , P(1) );
pause(1);

fwrite (arduino , P(2) );
pause(1);

fwrite (arduino , P(3) );
pause(1);

fwrite (arduino , P(4) );
pause(1);

fwrite (arduino , 0 );                % start signal
pause(1);

fclose(arduino);
```

Ως τελικό στάδιο του κώδικα, δημιουργείται ένας πίνακας, ο οποίος εμπεριέχει τις μεταβλητές που αντιστοιχούν στις κλίσεις των σερβοκινητήρων. Κατόπιν υλοποιείται η σύνδεση μεταξύ Matlab και Arduino και γίνεται η αποστολή των δεδομένων.
Ο ρόλος του Matlab πλέον έχει τελειώσει και από εδώ και πέρα ο μικροελεγκτής αναλαμβάνει τα επόμενα βήματα ελέγχου του βραχίονα.

INVERSE KINEMATICS

Σύμφωνα με την μέθοδο **inverse kinematics**, αντίθετα, η είσοδος που δίδεται στο Matlab είναι οι τρισδιάστατες συντεταγμένες που αντιστοιχούν στο σημείο του αντικειμένου και στο τελικό προορισμό του μέσα στο χώρο αλληλεπίδρασης του βραχίονα. Τη στιγμή που εισάγονται αυτές οι συντεταγμένες ακολουθεί μια διαδικασία υπολογισμού γωνιών κάθε σερβοκινητήρα μέσω κινηματικών αλγορίθμων που έχει ως αποτέλεσμα έναν πίνακα που περιλαμβάνει αυτά τα στοιχεία τιμών. Αναλυτικότερα η μαθηματική προσέγγιση που έχει χρησιμοποιηθεί, απλοποιεί το πρόβλημα αρχικά στις δύο διαστάσεις (κάτοψη) με σκοπό να ευθυγραμμιστεί ο βραχίονας με το σημείο επιλογής. Έπειτα το συνολικό προβλημα που απομένει λύνεται πάλι στις δύο διαστάσεις και έχει να κάνει με την πρισματική καθοδήγηση του βραχίονα στο επιθυμητό σημείο. Για την λύση αυτού, το δισδιάστατο πλευρικό επίπεδο χωρίζεται σε επιμέρους υποεπίπεδα που σε κάθε ένα απο αυτά αντιστοιχούν διαφορετικές κλίσεις ενός απο τους τρείς σερβοκινητήρες, συγκεκριμένα σε αυτόν που ανήκει στη βάση. Οι υπόλοιποι δύο που απομένουν δημιουργούν ένα τρίγωνο στο οποίο τριγωνομετρικά πλέον βρίσκονται οι σωστές κλίσεις για τη προσέγγιση του τελικού σημείου. Επόμενο βήμα είναι η αποστολή αυτών των αριθμών απο το Matlab στο μικροελεγκτή μέσω σειριακής επικοινωνίας που πραγματοποιούν μεταξύ τους.

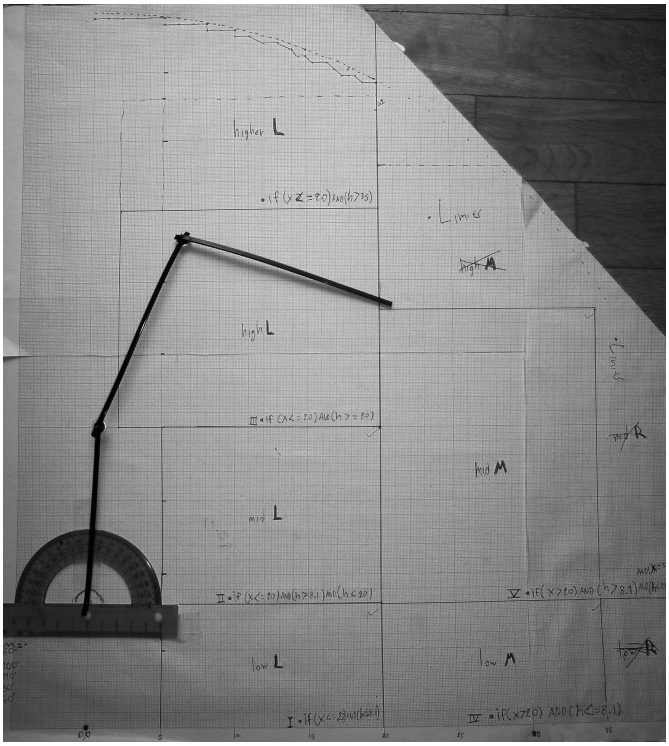
Πρόγραμμα Inverse Kinematics (Matlab)

```
clear all                                          % reset
clc

arm1 = 12;                                      % arms length
arm2 = 13.9;
arm3 = 15;
GA = 8.1;                                    % ground distance

in_x = 'type x: ';
in_h = 'type h: ';
in_y = 'type y: ';

x = input(in_x)
h = input(in_h)
y = input(in_y)
```

Ορισμός των μεγεθών άκρων του βραχίονα, το ύψος της βάσης και εισαγωγή επιθυμητών κλίσεων που επιλέγουμε για την αρχή της επεξεργασίας αυτών των τιμών στον παρακάτω κώδικα.

```
AB = arm1;
BC = arm2;
CD = arm3;


if (y>0)&&(y<=38)                          % 360 calculation
Y = '+';
ser1 = atand(x/y);
X = sqrt( x^2 + y^2 );
elseif (y<0)&&(y>=-38)
Y = '-';
ser11 = atand(x/y);
ser1 = 180 + ser11;
X = sqrt( x^2 + y^2 );
else  (y==0)
Y = '0';
ser1 = 90;
X = x;
end
```

Εύρεση κλίσης του σερβοκινητήρα βάσης, μέσω τριγωμετρίας. Η συντεταγμένη του ύψους δεν εμπεριέχεται σε αυτό το κομμάτι επειδή η ανάλυση γίνεται σε δύο διαστάσεις (κάτοψη).



Εικόνα 4

Η εικόνα 4 μας δείχνει διακεκριμένες περιοχές :

από LOW μέχρι HIGH οπου αντίστοιχα διακρίνονται σε LEFT - MID – RIGHT, οι οποίες στην πραγματικότητα αντιστοιχούν σε κάθε περίπτωση διαφορετικών συντεταγμένων μέσα στις περιοχές αυτές.
Βάσει των παραπάνω, ο κώδικας υπολογίζει τις συντεταγμένες κάθε επιλεγέντος σημείου των επί μέρους περιοχών.

```
DE = X;                          % transparent triangle
de = DE;

if (X<=20)&&(X>=5)&&(h<=8.1)                % low L
LOOP = 'low L';
ser2 = 60;
AE = GA - h;
AD = sqrt(AE^2 + DE^2);
ADE_a = asind(DE/AD);
ADE_d = 90 - ADE_a;
ABD_a =ser2 + ADE_d;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
```

```
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3_1 = ABCD_b - 90;

if SERVO3_1 < 0
   SERVO3_1 = 0;
end

SERVO3 = SERVO3_1;
SERVO4 = ABCD_c - 20;

elseif (X<=20)&&(X>=5)&&(h>8.1)&&(h<20)    % mid L
LOOP = 'mid L';
ser2 = 100;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a + 10;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X<=20)&&(X>=2)&&(h>=20)&&(h<43)   % high L
LOOP = 'high L';
ser2 = 110;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a + 20;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
```

```
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X>20)&&(X<=35)&&(h<=8.1)           % low M
LOOP = 'low M';
ser2 = 30;
AE = GA - h;
AD = sqrt(AE^2 + DE^2);
ADE_a = asind(DE/AD);
ADE_d = 90 - ADE_a;
ABD_a =ser2 + ADE_d;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 =  ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X>20)&&(X<=35)&&(h>8.1)&&(h<=28)   % mid M
LOOP = 'mid M';
ser2 = 40;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a - 50;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
```

```
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X>=2)&&(X<=20)&&(h>35)&&(h<43)    % higher L
LOOP = 'higher L';
AC = AB + BC;
AK = h - GA;
DK = X;
AD = sqrt(AK^2 + DK^2);
ABCD_a = acosd( (AC^2 + AD^2 - CD^2)/(2*AC*AD) );
ABCD_b = 90;
ABCD_c = acosd( (AC^2 + CD^2 - AD^2)/(2*AC*CD) );
ABCD_d = 180 - ABCD_a - ABCD_c;
ADK_a = acosd( (AD^2 + DK^2 - AK^2)/(2*AD*DK) );

if (AD<41)                                 % on limits
SERVO1 = ser1;
SERVO2 = ABCD_a + ADK_a;
SERVO3 = ABCD_b;
SERVO4 = ABCD_c - 20;

else                                      % off limits
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end
```

Αναλύοντας ένα απο το κομάτια μπλοκ κωδικα, διακρίνουμε μια περίπτωση εύρεση συντεταγμένων στην περιοχή (higher L).
Πριν το πρόγραμμα σαρώσει αυτό το μπλοκ κώδικα, έχει γίνει υπολογισμός, γνωρίζοντας την κλίση του ενός σερβοκινητήρα (κάτοψη) και το πρόβλημα ανάγεται πλέον σε πρόβλημα δύο διαστάσεων (όπως φαίνεται και στην εικόνα 4), προκειμένου εντός συγκεκριμένων ορίων να βρέθούν και οι υπόλοιπες κλίσεις των σερβοκινητήρων. Ειδάλως, όταν είναι εκτός ορίων, το πρόγραμμα αντιδρά, ορίζοντας τις κλίσεις του βραχίονα στην αρχική του θέση.

```
elseif (X>20)&&(X<=35)&&(h>28)          %  M - R
LOOP = 'M - R';
AC = AB + BC;
AK = h - GA;
```

```
DK = X;
AD = sqrt(AK^2 + DK^2);
ABCD_a = acosd( (AC^2 + AD^2 - CD^2)/(2*AC*AD) );
ABCD_b = 90;
ABCD_c = acosd( (AC^2 + CD^2 - AD^2)/(2*AC*CD) );
ABCD_d = 180 - ABCD_a - ABCD_c;
ADK_a = acosd( (AD^2 + DK^2 - AK^2)/(2*AD*DK) );

if (AD<41)                                  % on limits
SERVO1 = ser1;
SERVO2 = ABCD_a + ADK_a;
SERVO3 = ABCD_b;
SERVO4 = ABCD_c - 20;

else                                       % off limits
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end

else
LOOP = 'No_Limits';
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end
p = [SERVO1 , SERVO2 , SERVO3 , SERVO4];
```

Ο πίνακας του πρώτου σημείου τρισδιάστατων συντεταγμένων συμπεριλαμβάνεται έπειτα σε ένα πίνακα εννιά διαστάσεων λόγο οτι εμπεριέχονται και οι συντεταγμένες του τελικου σημείου κατάληξης.

Η συνέχεια του κώδικα είναι μια επανάληψη του παραπάνω κώδικα για την εισαγωγή και εύρεση του σημείου κατάληξης του αντικειμένου που μεταφέρει ο βραχίονας, οπότε επίσης ακολουθεί η ίδια μεθοδολογία εξαιρώντας το τμήμα του κώδικα που υλοποιεί τη σύνδεση με τον μικροελεγκτή.

```
in_x = 'type x: ';
in_h = 'type h: ';
in_y = 'type y: ';
```

```
x = input(in_x)
h = input(in_h)
y = input(in_y)

AB = arm1;
BC = arm2;
CD = arm3+2;

if (y>0)&&(y<=38)                    % 360 calculation  !
Y = '+';
ser1 = atand(x/y);
X = sqrt( x^2 + y^2 );

elseif (y<0)&&(y>=-38)
Y = '-';
ser11 = atand(x/y);
ser1 = 180 + ser11;
X = sqrt( x^2 + y^2 );

else  (y==0)
Y = '0';
ser1 = 90;
X = x;
end

DE = X;                        % transparent triangle


if (X<=20)&&(X>=5)&&(h<=8.1)                % low L
LOOP = 'low L';
ser2 = 60;
AE = GA - h;
AD = sqrt(AE^2 + DE^2);
ADE_a = asind(DE/AD);
ADE_d = 90 - ADE_a;
ABD_a =ser2 + ADE_d;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3_1 = ABCD_b - 90;
```

```
if SERVO3_1 < 0
   SERVO3_1 = 0;
end

SERVO3 = SERVO3_1;
SERVO4 = ABCD_c - 20;

elseif (X<=20)&&(X>=5)&&(h>8.1)&&(h<20)    % mid L
LOOP = 'mid L';
ser2 = 100;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a + 10;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;


SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X<=20)&&(X>=2)&&(h>=20)&&(h<43)   % high L
LOOP = 'high L';
ser2 = 110;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a + 20;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;
```

```
SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
SERVO4 = ABCD_c - 20;

elseif (X>20)&&(X<=35)&&(h<=8.1)           % low M
LOOP = 'low M';
ser2 = 30;
AE = GA - h;
AD = sqrt(AE^2 + DE^2);
ADE_a = asind(DE/AD);
ADE_d = 90 - ADE_a;
ABD_a =ser2 + ADE_d;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 =  ABCD_b - 90;
SERVO4 = ABCD_c - 20;
elseif (X>20)&&(X<=35)&&(h>8.1)&&(h<=28)   % mid M
LOOP = 'mid M';
ser2 = 40;
AE = h - GA;
AD = sqrt(AE^2 + DE^2);
ADE_a = acosd( (AE^2 + AD^2 - DE^2)/(2*AE*AD));
ADE_d = 90 - ADE_a;
ABD_a = ADE_a - 50;
BD = sqrt(AB^2 + AD^2 - 2*AB*AD*cosd(ABD_a));
ABD_d = acosd( (AD^2 + BD^2 - AB^2)/(2*AD*BD) );
ABD_b = 180 - ABD_a - ABD_d;
BCD_b = acosd( (BC^2 + BD^2 - CD^2)/(2*BC*BD) );
BCD_c = acosd( (BC^2 + CD^2 - BD^2)/(2*BC*CD) );
BCD_d = 180 - BCD_b - BCD_c;
ABCD_a = ABD_a;
ABCD_b = ABD_b + BCD_b;
ABCD_c = BCD_c;
ABCD_d = ABD_d + BCD_d;

SERVO1 = ser1;
SERVO2 = ser2;
SERVO3 = ABCD_b - 90;
```

```
SERVO4 = ABCD_c - 20;

elseif (X>=2)&&(X<=20)&&(h>35)&&(h<43)    % higher L
LOOP = 'higher L';
AC = AB + BC;
AK = h - GA;
DK = X;
AD = sqrt(AK^2 + DK^2);
ABCD_a = acosd( (AC^2 + AD^2 - CD^2)/(2*AC*AD) );
ABCD_b = 90;
ABCD_c = acosd( (AC^2 + CD^2 - AD^2)/(2*AC*CD) );
ABCD_d = 180 - ABCD_a - ABCD_c;
ADK_a = acosd( (AD^2 + DK^2 - AK^2)/(2*AD*DK) );

if (AD<41)                                 % on limits
SERVO1 = ser1;
SERVO2 = ABCD_a + ADK_a;
SERVO3 = ABCD_b;
SERVO4 = ABCD_c - 20;

else                                      % off limits
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end

elseif (X>20)&&(X<=35)&&(h>28)            %  M - R
LOOP = 'M - R';
AC = AB + BC;
AK = h - GA;
DK = X;
AD = sqrt(AK^2 + DK^2);
ABCD_a = acosd( (AC^2 + AD^2 - CD^2)/(2*AC*AD) );
ABCD_b = 90;
ABCD_c = acosd( (AC^2 + CD^2 - AD^2)/(2*AC*CD) );
ABCD_d = 180 - ABCD_a - ABCD_c;
ADK_a = acosd( (AD^2 + DK^2 - AK^2)/(2*AD*DK) );

if (AD<41)                                 % on limits
SERVO1 = ser1;
SERVO2 = ABCD_a + ADK_a;
SERVO3 = ABCD_b;
SERVO4 = ABCD_c - 20;

else                                      % off limits
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end
```

```
else
LOOP = 'No_Limits';
SERVO1 = 90;
SERVO2 = 90;
SERVO3 = 0;
SERVO4 = 110;
end
```

Πρόκειται για το τέλος του κωδικα που αφορά τον υπολογισμό των δύο διαφορετικών τρισδιάστατων συντεταγμένων, όπου κινείται ο βραχίονας με σκοπό τη μεταφορά ενός αντικειμένου απο μια θέση, σε μια άλλη.

```
P=[p(1),p(2),p(3),p(4),SERVO1,SERVO2,SERVO3,SERVO4,DE];
```

Ο τελικός πίνακας που συμπεριλαμβάνει τις επιθυμιτές συντεταγμένες που θα κινηθεί ο βραχίονας και μία τιμή που χρησιμοποιείται για την αρχική απόσταση μεταξύ βραχίονα και αντικειμένου.

```
arduino=serial('/dev/cu.usbmodemfa131','BaudRate',9600);

fopen(arduino);
pause(2);
fwrite (arduino , P(1) );
pause(1);
fwrite (arduino , P(2) );
pause(1);
fwrite (arduino , P(3) );
pause(1);
fwrite (arduino , P(4) );
pause(1);
fwrite (arduino , P(5) );
pause(1);
fwrite (arduino , P(6) );
pause(1);
fwrite (arduino , P(7) );
pause(1);
fwrite (arduino , P(8) );
pause(1);
fwrite (arduino , de );              % 1st x
pause(1);
fwrite (arduino , DE );              % 2nd x
pause(1);
fwrite (arduino , 0 );
pause(1);
fclose(arduino);
```

Όπως και στην περίπτωση Forward Kinematics (σελ. 17), ο κώδικας κάνει αντίστοιχες ενέργειες στέλνοντας τα δεδομένα στον μικροελεγκτή.

Συνδεσιμότητα μεταξύ MATLAB / ARDUINO

Ο τρόπος σύνδεσης μεταξύ υπολογιστή και μικροελεγκτή υλοποιείται μέσω σειριακής επικοινωνίας. Αναλυτικότερα, η ταχύτητα συγχρονισμού κάθε φορά που γίνεται η μετάδοση είναι στα 9600 baud rate δηλαδή 9.6 kbps. Επειδή τα δεδομένα που μεταδίδονται είναι λίγα, δεν υπάρχει ανάγκη μεγάλης ταχύτητας. Ουσιαστικά τα δεδομένα που μεταδίδονται είναι έντεκα integer μεταβλητές. Απο τη μεριά του Matlab αυτά τα στοιχεία ανοίκουν σε έναν πίνακα. Οι οχτώ από αυτές μεταβλήτες αντιστοιχούν στις κλίσεις των σερβοκινητήρων. Οι δύο είναι οι τιμές που υποδεικνύουν την απόσταση απο τη βάση σε κάθε σημείο (δύο για την συγκεκριμένη περίπτωση), και η τελευταία μεταβλητή ορίζει το τέλος της μετάδοσης και ταυτόχρονα την έναρξη της ρουτίνας του ελεγκτή. Αυτή η σύνδεση υλοποιείται στο τελευταίο τμήμα του κώδικα στο Matlab, εφόσον εχουν υπολογιστεί οι κλίσεις των συντεταγμένων.

Συγκεκριμένα οι εντολές:

```
arduino=serial('/dev/cu.usbmodemfa131','BaudRate',9600);
fopen(arduino);
```

ανοίγουν τη σύνδεση μεταξύ υπολογιστή και μικροελεγκτή, Matlab και Arduino αντίστοιχα. Η φυσική τους σύνδεση γίνεται με ένα κοινό USB καλώδιο μεταξύ υπολογιστή και μικροελεγκτή, το οποίο χρησιμοποιείται επίσης απο το μικροελεγκτή για τον προγραμματισμό του.

Πρόγραμμα Μικροελεγκτή (Arduino)

```
//------------------------------------------- variables..

#include <Servo.h>
#include <NewPing.h>

#define trigger  3
#define echo     2
#define MAX 200

int cm = 57;
int cal = 285;
NewPing sonar(trigger, echo, MAX);

int pos1 = 0;
int pos2 = 0;

int go = 0;
int re = 0;

int X = 0;

int ma3 = 0;
int d = 0;
int dd = 0;
int ddd= 0;
int d1 = 0;
int d2 = 0;
int d3 = 0;
int d4 = 0;

int ch = 0;

int check  = 0;
```

```
int check1 = 0;
int check2 = 0;
int de = 0;
int DE = 0;

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;

int p[]= {0,0,0,0,0,0,0,0,0,0,1};    // data array
int P[]= {0,0,0,0,0,0,0,0,0,0,1};   // calibrated array
int i=0;
int data;
int s1 = 90;
int s2 = 40;
int s3 = 0;
int s4 = 110;
int s5;
int q5;
int S2;
int S3;
int S4;
int q1 = 90;
int q2 = 40;
int q3 = 0;
int q4 = 110;
//---------------------------------------------- ..variables
```

Δήλωση βιβλιοθηκών sonar, servo και μεταβλητών του προγράμματος.

```
void setup() {                              //  SETUP

servo1.attach(8);
servo2.attach(9);
servo3.attach(10);
servo4.attach(11);
servo5.attach(12);
servo6.attach(13);

servo1.write(90);
servo2.write(40);
servo3.write(0);
servo4.write(110);
servo5.write(78);
servo6.write(50);

Serial.begin(9600);
}
```

Οι πρώτες εντολές `(servo1.attach)`, με τα νούμερα εντος παρενθέσεως, δηλώνουν την αντίστοιχη έξοδο του μικροελεγκτή που συνδέεται με κάθε σερβοκινητήρα. Οι επόμενες εντολές `(servo6.write)` αρχικοποιούν τη θέση των σερβοκινητήρων με τις κλίσεις από 0 μέχρι 180, που φαίνεται σε κάθε ένα απο αυτούς.

```
void loop() {                                   //   LOOP

//--------------------------------------------- importing

if(Serial.available()>0) {

while (Serial.available()>0)  {
data=Serial.read();
p[i]=data;
i++;

if (i==11)
i = 0;      } }
```

Άνοιγμα σειριακής επικοινωνίας μέσο εντολής και είσοδος δεδομένων με σκοπό την εισαγωγή τους σε έναν πίνακα

```
//--------------------------------------------- calibrating

P[0] = p[0];
P[1] = abs (p[1] - 170 );
P[2] = p[2];
P[3] = abs (p[3] - 180 );
P[4] = p[4];
P[5] = abs (p[5] - 170 );
P[6] = p[6];
P[7] = abs (p[7] - 180 );
P[8] = p[8]; // 1st X
de = P[8];
P[9] = p[9]; // 2nd X
DE = P[9];
P[10] = p[10];
```

Αντιστοίχηση δεδομένων εισόδου με σκοπό τη διαμέτρηση τους. Κύριος λόγος αυτού, η αντίστροφη τοποθέτηση των σερβοκινητήρων πάνω στο βραχίονα αλλά και η ευθυγράμμιση που χρειάζεται κάθε ένας απο αυτούς.

Ο παρακάτω κώδικας αφορά την κίνηση του βραχίονα σε αυτά τα δύο σημεία που έχουν οριστεί και επίσης ελέγχει ανά περιόδους το χώρο μπροστά του με σκοπό την αποφυγή συγκρούσεων με άλλα αντικείμενα στατικά ή και κινούμενα.

```
//------------------------------------------ GO

if (P[10]==0) {

do {

if      (s1>P[0]) { s1--; }
else if (s1<=P[0]) { s1++; }
servo1.write(s1);              // 1st slow
delay(20);

}while (s1 != P[0]);
```

Εκμεταλευόμενος την ρουτίνα do.. while η κίνηση του κάθε σερβοκινητήρα γίνεται ομαλά, χωρίς ανεπιθύμητες ταλαντώσεις. Σε αυτό το κοματι του κώδικα ελέγχεται ένας σερβοκινητήρας.

```
q1 = s1;

blockX();

if ( (P[0]>170) || ( P[0]<10 ) ) { servo5.write(78); } // 5th
else  { servo5.write(P[0]-12); }

do{

if      (s2>P[1]) { s2--; }
else if (s2<P[1]) { s2++; }

if      (s3>P[2]) { s3--; }
else if (s3<P[2]) { s3++; }

if      (s4>P[3]) { s4--; }
else if (s4<P[3]) { s4++; }

servo2.write(s2);  // 2nd slow
servo3.write(s3);  // 3rd slow
servo4.write(s4);  // 4th slow

if ( (s2 == P[1]) && (s3 == P[2]) && (s4 == P[3]) )
{ go = 1; }

else
{ go = 0; }

delay(30);

}while (go == 0);
```

Αντίστοιχη ρουτίνα do.. while για την ομαλή κίνηση τριών σερβοκινητήρων που κινούνται ταυτόχρονα.

```
delay(500);
servo6.write(100);            // 6th claw
delay(1000);
//------------------------------------------- starting point

S2 = s2 - 30;   // fast lift
S3 = s3;
S4 = s4;
```

Η ρουτίνα do.. while αρχικοποιεί τη θέση του βραχίονα εφόσον έχει αρπάξει το αντικείμενο μεταφοράς.

```
  do{
if      (S2>40) { S2--; }
else if (S2<40) { S2++; }

if      (S3>2) { S3--; }
else if (S3<2) { S3++; }

if      (S4>100) { S4--; }
else if (S4<100) { S4++; }

servo2.write(S2);  // 2nd slow
servo3.write(S3);  // 3rd slow
servo4.write(S4);  // 4th slow

if ( (S2 == 40) && (S3 == 2) && (S4 == 100) )
{ pos1 = 1; }
else
{ pos1 = 0; }

delay(30);
}while (pos1 == 0);
servo5.write(78);
```

```
blockP();    //    Pause


//------------------------------------------- pointed

do {

if (q1>P[4]) { q1--; }

else if (q1<P[4]) { q1++; }

servo1.write(q1);            // 1st slow
delay(20);

}while (q1 != P[4]);

blockD();


//------------------------------------------- GO


if ( (P[4]>170) || ( P[4]<10 ) ) { servo5.write(78); }  // 5th
else  { servo5.write(P[4]-12); }    // servo

do{

blockP();    //    Pause

if      (q2>P[5]) { q2--; }
else if (q2<P[5]) { q2++; }
if      (q3>P[6]) { q3--; }
else if (q3<P[6]) { q3++; }

if      (q4>P[7]) { q4--; }
```

```
else if (q4<P[7]) { q4++; }

servo2.write(q2);  // 2nd slow
servo3.write(q3);  // 3rd slow
servo4.write(q4);  // 4th slow

if ( (q2 == P[5]) && (q3 == P[6]) && (q4 == P[7]) )
{ re = 1; }

else { re = 0; }

delay(30);

}while (re == 0);

delay(500);
servo6.write(40);                    // 6th claw
delay(1000);

//---------------------------------------- starting position

q2 = q2 -30; // fast lift

do{

if      (q2>40) { q2--; }
else if (q2<40) { q2++; }

if      (q3>0) { q3--; }
else if (q3<0) { q3++; }
if      (q4>100) { q4--; }
else if (q4<100) { q4++; }

servo2.write(q2);  // 2nd slow
servo3.write(q3);  // 3rd slow
```

```
servo4.write(q4);  // 4th slow
if ( (q2 == 40) && (q3 == 0) && (q4 == 100) ) { pos2 = 1; }
else { pos2 = 0; }
delay(30);

}while (pos2 == 0);

do {

if      (q1>90) { q1--; }
else if (q1<=90) { q1++; }

servo1.write(q1);            // 1st slow
delay(20);

}while (q1 != 90);
servo5.write(78);

p[10]=1;

}  }          //    LOOP
```

Σε αυτό το σημείο ο βασικός κώδικας τελειώνει και ακολουθούν οι κλάσεις, οι οποίες εμπεριέχονται μέσα στον παραπάνω κώδικα. Λόγος δημιουργίας των κλάσεων είναι ο ευκολότερος και γρηγορότερος χειρισμός αλλαγών του κώδικα.

```
//------------------------------------------ classes
```

Η κλάση `blockP()` πραγματοποιεί στιγμιαία παύση όταν καλείται να ‘τρέξει’, σε περίπτωση εμποδίου αλλά και άμεση συνέχεια όταν ελευθερωθεί ο χώρος όπου κινείται ο βραχίονας.

```
void blockP() { // pause

do{

check = sonar.ping();
if(check > 2000) { check = 0; }
if ( check == 0 ) { ch = 0; }
else { ch = ( ( ma3 +cal ) / cm); }

}while (ch != 0);

check = 0;
}
```

Η κλάση `blockD()` ελέγχει το χώρο τη στιγμή που είναι κεντραρισμένος ο βραχίονας στο σημείο, έχοντας πιάσει το αντικείμενο. Σε περίπτωση ελεύθερου χώρου το αντικείμενο τοποθετείται στο έδαφος, ειδάλως ο βραχίονας περιμένει την απελευθέρωση του χώρου.

```
void blockD() { // drop

do{

servo1.write(q1+2);
delay(200);

d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();

if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

ddd = max( max(d1 , d2) , max(d3, d4) ); // 1o

servo1.write(q1-2);
delay(200);

d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();

if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
```

```
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

dd = max( max(d1 , d2) , max(d3, d4) ); // 2o

servo1.write(q1);
delay(200);

d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();

if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

d = max( max(d1 , d2) , max(d3, d4) ); // 3o
ma3 = max( d , max( dd , ddd ) ); // MAX measurment

if ( ma3 == 0 ) { check1 = 0; }
else { check1 = ( ( ma3 +cal ) / cm); }
delay(500);

}while (check1 != 0);
ma3 = 0;
d  = 0;
dd  = 0;
ddd  = 0;
d1 = 0;
d2 = 0;
```

```
d3 = 0;
d4 = 0; }
```

Η κλάση `blockX()` καλείται σε συγκεκριμένο τμήμα του κώδικα και έχει ως σκοπό να σκανάρει την περιοχή που βρίσκεται το αντικείμενο πριν το πιάσει ο βραχίονας. Σε περίπτωση που δεν υπάρχει το αντικείμενο στη θέση αυτη περιμένει μέχρι να εντοπιστεί σε συγκεκριμενο σημείο. Αλιώς βρίσκεται σε αναμονή και σαρώσει το πεδίο ξανά και ξανά.

```
void blockX() { // 1st lift

do{

servo1.write(s1+2);
delay(200);
d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();

if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

ddd = max( max(d1 , d2) , max(d3, d4) ); // 1o

servo1.write(s1-2);
delay(200);
```

```
d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();
if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

dd = max( max(d1 , d2) , max(d3, d4) ); // 2o
servo1.write(s1);
delay(200);
d1 = sonar.ping();
delay(100);
d2 = sonar.ping();
delay(100);
d3 = sonar.ping();
delay(100);
d4 = sonar.ping();

if(d1 > 2000) { d1 = 0; }
if(d2 > 2000) { d2 = 0; }
if(d3 > 2000) { d3 = 0; }
if(d4 > 2000) { d4 = 0; }

d = max( max(d1 , d2) , max(d3, d4) ); // 3o

ma3 = max( d , max( dd , ddd ) ); // MAX measurment

if ( ma3 == 0 ) { check1 = 0; }
else { check1 = ( ( ma3 +cal ) / cm); }
```

```
delay(500);

if( (check1 <= de+3) && (check1 >= de-3) ) { X=1; }
else { X=0;  }
}while ( X==0 );
ma3 = 0;
d  = 0;
dd  = 0;
ddd  = 0;
d1 = 0;
d2 = 0;
d3 = 0;
d4 = 0;  }
```

ΑΠΟΤΕΛΕΣΜΑΤΑ

Εφαρμόζοντας την παραπάνω Μεθοδολογία στο συγκεκριμένο Βραχίονα, παρατηρούμε τα εξής:

- Ακρίβεια κινήσεων:

  Η ακρίβεια των κινήσεων είναι απολύτως εξαρτημένη απο το υλικό, από το οποίο είναι φτιαγμένος ο βραχίονας, σε συνδυασμό με την ακρίβεια κλίσης που έχουν οι σερβοκινητήρες.
  Επίσης είναι εξαρτημένη και από το λογισμικό. Δηλαδή μια υψηλή ‘ποιότητα’ κώδικα συνδιαστικά με κατάλληλα εξωτερικά αισθητήρια μπορούν να οδηγήσουν στο επιθυμητό αποτέλεσμα.
  Στη συγκεκριμένη περίπτωση τα αποτελέσματα ήταν ικανοποιητικά πλην περιπτώσεων, κατά τις οποίες ο βραχίονας αποκτούσε το μέγιστο άνοιγμα κι αυτό, λόγω της ελαστικότητας του υλικού κατασκευής του.

- Ταχύτητα κινήσεων:

  Εξαρτάται από τους σερβοκινητήρες συνδιασμένων με το λογισμικό, το οποίο τα οδηγεί στην επιθυμητή θέση.
  Στη συγκεκριμένη περίπτωση, αν και ο Βραχίονας είχε τη δυνατότητα σχετικά μεγάλης ταχύτητας, εν τούτοις, λόγω της ελαστικότητας του υλικού, στις μεγάλες ταχύτητες παραρηρήθηκαν ταλαντώσεις, οι οποίες δημιουργούσαν αστοχίες.
  Για το λόγο αυτόν, προγραμματίστηκαν πιο μικρές ταχύτητες.

- Μεταφορά μικροαντικειμένων:

  Ανάλογα με τη δύναμη και τη γεωμετρία εκάστοτε βραχίονα καθορίζεται το βάρος και ο όγκος των αντικειμένων προς μεταφορά.
  Ο εν λόγω βραχίονας έχει τη δυνατότητα να αποσπάσει και μετακινήσει αντικείμενο όγκου της τάξης 9 κυβικών εκκατοστών περίπου και βάρους 100 γραμμαρίων περίπου.

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Με την παρούσα εργασία επιτεύχθηκε η λειτουργία ενός ρομποτικού βραχίονα πολύ μικρού μεγέθους για τη μεταφορά αντικειμένων σε συγκεκριμένα σημεία μικρών διαστάσεων και μικρού βάρους.
Εν τούτοις όλες οι κινήσεις έγιναν επιτυχώς και με ασφάλεια δεδομένου, ότι υπήρξε ορθός προγραμματισμός, προκειμένου οι μετακινήσεις να λαμβάνουν υπόψη τα εμπόδια και να υπάρχει η δυνατότητα αποφυγής τους.
Από την άλλη διαπιστώθηκαν παράγοντες που προκαλούσαν σχετικές αστοχίες, όπως π.χ. το υλικό, που προκαλούσε ταλαντώσεις στα μεγαλύτερα επιθυμητά ανοίγματα.
Όμως τέτοια προβλήματα σε περιπτώσεις λειτουργίας βραχιόνων σε μεγάλες κλίμακες μπορούν να αποφευχθούν.
Σε κάθε περίπτωση ο βραχίονας της παρούσας εργασίας θεωρούμε, ότι αποτελεί ένα καλό παράδειγμα εφαρμογής της μεθόδου και σε μεγαλύτερη κλίμακα. Δηλαδή υπό συνθήκες εργοταξίου, όπου η μεταφορά των αντικειμένων μπορεί να γίνει με ασφάλεια και ταχύτητα. Εκεί οι λεπτομέρειες της κατασκευής είναι δυνατόν να επιλεγούν προς όφελος της λειτουργίας.

Σε γενικές γραμμές η κατασκευή και ο προγραμματισμός ενός ρομποτικού βραχίονα είναι ένα θέμα, το οποίο, πέρα από τις κατασκευαστικές και τεχνικές προκλήσεις, προσφέρει ένα πεδίο προγραμματιστικών προκλήσεων στον αυτοματιστή, που καλείται να αναλύσει το πρόβλημα, να σχεδιάσει και να υλοποιήσει αλγοριθμικές λύσεις συνδυάζοντας γνώσεις από διάφορα επιστημονικά πεδία, προκειμένου να πετύχει το σκοπό του.
Οι σκοποί δε εφαρμογής μπορούν να ποικίλουν. Πέραν της μεταφοράς αντικειμένων, μπορεί να χρησιμοποιηθεί σε συγκολλήσεις εξαρτημάτων, σε βαφές αντικειμένων, στην τοποθέτηση αντικειμένων του ενός πάνω στο άλλο ή πλάϊ στο άλλο, στη συσκευασία προϊόντων αλλά και στον έλεγχο προϊόντων.
Σε κάθε περίπτωση αποτελεί ένα κεφάλαιο του Αυτοματισμού με εξαιρετικό ενδιαφέρον από ερευνητική άποψη.

ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ

1. Industrial Rbotics: Theory, Modeling and Control (2006)
   SERDAR KUCUK and ZAFER BINGUL

2. MATLAB tutorial online
   http://www.tutorialspoint.com/matlab/

3. ARDUINO tutorial and examples
   https://www.arduino.cc

4. ΡΟΜΠΟΤΙΚΗ – ΑΥΤΟΜΑΤΑ (2006), Σύγχρονη Εκδοτική
   ΔΗΜ. Χ. ΒΟΥΚΑΛΗΣ και ΕΙΡΗΝΗ Δ. ΒΟΥΚΑΛΗ

5. ΡΟΜΠΟΤΙΚΗ (2003)
   ΣΠΥΡΟΥ Γ. ΤΖΑΦΕΣΤΑ