**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΣΥΣΤΗΜΑΤΑ ΑΥΤΟΜΑΤΙΣΜΟΥ»

Μεταπτυχιακή Εργασία

# Σχεδιασμός αυτόματου συστήματος τρισδιάστατης σάρωσης

Γιώργος Α. Τζαφίλκος

*Επιβλέπων Καθηγητής:Γ.Χ Βοσνιάκος*

ΑΘΗΝΑ 2021

# Περίληψη

Σκοπός της συγκεκριμένης εργασίας είναι ο σχεδιασμός και η κατασκευή συστήματος το οποίο θα προσφέρει τη μερική αυτοματοποίηση της τρισδιάστατης σάρωσης. Πιο συγκεκριμένα, στο εργαστήριο τεχνολογίας των κατεργασιών του ΕΜΠ υπάρχει ο τρισδιάστατος σαρωτής iScan M300 3D, ο οποίος χρησιμοποιείται για την τρισδιάστατη σάρωση αντικειμένων. Η τοποθέτηση και η μετακίνηση του σαρωτή σε κατάλληλες θέσεις για την σάρωση γίνεται χειροκίνητα. Η εργασία αυτή σκοπό έχει να αυτοματοποιήσει αυτή τη διαδικασία. Προτείνεται διάταξη, η οποία θα φέρει τον σαρωτή με 3 βαθμούς ελευθερίας, επιτρέποντας του κίνηση οριζόντια και κατακόρυφα ως προς το κατακόρυφο επίπεδο καθώς επίσης και περιστροφή γύρω από άξονα κάθετο σε αυτό. Η πρόταση του συστήματος ολοκληρώνεται με τραπέζι, στο οποίο θα στερεώνεται το αντικείμενο προς σάρωση και θα μπορεί να περιστρέφεται ελεγχόμενα αριστερόστροφα ή δεξιόστροφα ανάλογα με τις ανάγκες της σάρωσης. Τέλος, προτείνεται γραφικό περιβάλλον χρήστη για τον έλεγχο του συστήματος μέσω ηλεκτρονικού υπολογιστή.

Η εργασία ξεκινάει με το την αποσαφήνιση της έννοιας της τρισδιάστατης σάρωσης, την εξέλιξή της τα τελευταία χρόνια, τις διάφορες κατηγορίες σαρωτών, τη χρήση της στη βιομηχανία αλλά και εκτιμήσεις για την εξέλιξή της. Στη συνέχεια, γίνεται αναφορά στα βασικά στοιχεία δημιουργίας, μετάδοσης και ελέγχου της κίνησης.

Έπειτα, η εργασία επικεντρώνεται στον σχεδιασμό και τη δημιουργία του CAD μοντέλου, τις επιλογές που έγιναν κατά την διάρκεια της σχεδίασης, προσομοιώσεις συμπεριφοράς για κάποια εξαρτήματα του συστήματος καθώς επίσης και τη δημιουργία Gcode για την παραγωγή πρωτοτύπων εξαρτημάτων σε μηχανή CNC.

Τέλος, παρουσιάζεται ο προγραμματισμός του μικροελεγκτή μέσω του IDE Arduino με χρήση open source βιβλιοθηκών. Ακολουθείται η διαδικασία δημιουργίας γραφικού περιβάλλοντος χρήστη μέσω της πλατφόρμας Microsoft Visual Studio. Μέσα από αυτό ο χρήστης θα μπορεί να αλλάζει συγκεκριμένες μεταβλητές του κώδικα άρα και την κίνηση των κινητήρων ορίζοντας τη σχετική θέση του σαρωτή με το αντικείμενο.

Η εργασία ολοκληρώνεται με αποτελέσματα και συμπεράσματα που προέκυψαν από αυτήν καθώς και προτάσεις για βελτίωση.

# Abstract

The purpose of this thesis is the design and therefore the construction of a system that will semi automate the 3D scanning process. More specifically, the Manufacturing Technology Lab of NTUA uses iScan M300 3D Scanner which is used for 3D scanning objects. The scanner is placed and moved to the appropriate positions manually. This thesis aims to automate this process. Proposing a system that will carry the scanner offering it 3 degrees of freedom allowing it to move horizontally and vertically with respect to the vertical plane as well as rotate around an axis perpendicular to it. The proposal of the platform is completed with a rotating table to which the object will be fixed. The rotation of the table clockwise or anticlockwise will be controlled by the user. Finally, a graphical user interface is proposed for the control of the system through a computer.

The thesis begins with what exactly is 3D scanning, how it has evolved in recent years, the various categories of scanners, the use of 3D scanning in the industry and estimates for its development. Next, the basic elements of creating motion and motion control such as motors, gears, belts, linear actuators, microcontrollers, and motor drivers are referred to. The thesis continues with the design and creation of the CAD model, the choices made during the design, CAE studies for some system components as well as the production of Gcode to produce some original parts on a CNC machine.

Finally, the programming of the microcontroller is presented through the IDE Arduino and with the use of open-source libraries. The process of creating a graphical environment through the Microsoft Visual Studio platform is followed. Through this graphical interface user will be able to change specific script variables that control the motors. Thus, the user will be able to control the relevant position between scanner and object

The thesis is completed with results and conclusions that emerged from it and suggestions for continuation of the work.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# 1 Εισαγωγή

## 1.1 Σκοπός Εργασίας

Σκοπός τις συγκεκριμένης εργασίας είναι να αυτοματοποιήσει μερικώς τη διαδικασία της τρισδιάστατης σάρωσης. Το εργαστήριο τεχνολογίας των κατεργασίων του ΕΜΠ χρησιμοποιεί τον 3D σαρωτή iScan M300 για τη σάρωση αντικειμένων. Η τοποθέτηση του σαρωτή στις κατάλληλες θέσεις γίνεται χειροκίνητα. Η παρούσα εργασία θα προτείνει σύστημα, μέσω του οποίου θα αυτοματοποιηθεί η τοποθέτηση του σαρωτή στον χώρο και σε σχέση με το υπό σάρωση αντικείμενο. Η αυτοματοποίηση αυτή θα προσφέρει μεγαλύτερη ακρίβεια θέσης στον σαρωτή. Στόχος, επίσης, είναι να μειώσει δραστικά τον χρόνο ολόκληρης της διαδικασίας.

## 1.2 Η τρισδιάστατη σάρωση

Η τρισδιάστατη σάρωση (3d scanning) είναι μια μέθοδος μέτρησης διαστάσεων αντικειμένων και χώρων. Στη βιβλιογραφία αναφέρονται μέθοδοι τρισδιάστατης σάρωσης επαφής και μη επαφής. Στην παρούσα εργασία θα αναφερθούμε στις μεθόδους μη επαφής. Αυτό σημαίνει ότι μπορούμε να πάρουμε μέτρηση από το αντικείμενο ή τον χώρο που θέλουμε να μετρήσουμε χωρίς να έχουμε επαφή του μετρητικού οργάνου σε κάποια επιφάνεια. Σε κάποιες περιπτώσεις με τη μέτρηση των διαστάσεων είναι δυνατό να έχουμε και άλλες πληροφορίες για την επιφάνεια που μετράμε, όπως για παράδειγμα το χρώμα της. Η αρχή της τρισδιάστατης σάρωσης είναι η συγκέντρωση πολλών σημείων ή αλλιώς νέφους σημείων, τα οποία με την σειρά τους μέσω λογισμικού μπορούν χρησιμοποιηθούν για την κατασκευή τρισδιάστατων γραφικών μοντέλων. Με άλλα λόγια, έχουμε μια διαδικασία στην οποία ψηφιοποιούμε αντικείμενα και χώρους από τον πραγματικό κόσμο και τα αναπαριστούμε γραφικά με την βοήθεια υπολογιστή. Παρότι υπάρχουν διάφορες κατηγορίες σαρωτών, στην βιομηχανία έχουν επικρατήσει κυρίως δύο, οι τρισδιάστατοι σαρωτές δομημένου φωτός και οι τρισδιάστατοι σαρωτές λέιζερ.

Οι τρισδιάστατοι σαρωτές δομημένου φωτός προβάλουν εναλλασσόμενα φωτεινά μοτίβα πάνω στην επιφάνεια ενώ οι κάμερες του σαρωτή συλλέγουν εικόνες από διάφορες φάσεις των προβολών. Η διαδικασία αυτή επαναλαμβάνεται αρκετές φορές σταδιακά σε όλη την επιφάνεια του αντικειμένου με αποτέλεσμα να δημιουργηθεί ένα πυκνό νέφος σημείων. Στη συνέχεια το νέφος αυτό με την βοήθεια υπολογιστή και λογισμικού μετατρέπεται σε ένα στερεολιθικό αρχείο (stl) το οποίο με την σειρά του μπορεί να χρησιμοποιηθεί από CAD λογισμικά. Κύριο πλεονέκτημα της μεθόδου αυτής είναι η μεγάλη ακρίβεια της μέτρησης, ενώ ως μειονέκτημα έχουμε τον σχετικά μεγάλο χρόνο σάρωσης. Ένας περιορισμός της

μεθόδου είναι ότι το αντικείμενο της μέτρησης πρέπει να είναι απόλυτα σταθερό.

Κατά την σάρωση με λέιζερ το αντικείμενο σαρώνεται μόνο μια φορά, δημιουργώντας αντίστοιχα ένα νέφος σημείων. Η μέθοδος αυτή είναι αρκετά γρήγορη μεν, υπολείπεται όμως σε ακρίβεια σε σχέση με τη μέθοδο δομημένου φωτός. Η τρισδιάστατη σάρωση είναι ένα εργαλείο το οποίο υιοθετείται από όλο και περισσότερους κλάδους στην βιομηχανία. Μέχρι στιγμής έχει μεγάλη εφαρμογή σε τομείς όπως η αντίστροφη μηχανική, ο σχεδιασμός σύνθετων καμπύλων επιφανειών, η αεροδιαστημική, η υγεία, η τέχνη, η δημιουργία πρωτοτύπων κ.α.

## 1.3 Βιβλιογραφική Ανασκόπηση

Η αυτοματοποίηση της τρισδιάστατης σάρωσης και ο σχεδιασμός ενός μηχανικού συστήματος που θα την υποστηρίζει είναι ένα διεπιστημονικό αντικείμενο μελέτης. Συνδυάζει τομείς μηχανολογίας, ηλεκτρονικής, προγραμματισμού και υπολογιστικής όρασης. Υπάρχουν αρκετές μελέτες πάνω σε αυτό το θέμα και παρακάτω θα αναφερθούμε σε μερικές από αυτές.

Στα πλαίσια του αυτοματισμού της διαδικασίας για χρήση σε εφαρμογές αντίστροφης μηχανικής προτάθηκε ένα ρομποτικό σύστημα για τη σάρωση αντικειμένων οργανικών μορφών (Zhao et al. 2008). Στην προσέγγιση αυτή, ο σαρωτής λέιζερ είναι σταθερός σε μια βάση και το αντικείμενο προς σάρωση συγκρατείται από τον ρομποτικό βραχίονα IRB 4400 της ABB. Για την σάρωση του αντικειμένου προτάθηκαν δύο διαδρομές κίνησης του βραχίονα. Η μια ονομάζεται rotation mode και χρησιμοποιείται σε κομμάτια κυλινδρικής γεωμετρίας. Σε αυτήν ο βραχίονας περιστρέφει σταδιακά το κομμάτι μέχρι να γίνει μια πλήρης περιστροφή. Η άλλη ονομάζεται four view mode και όπως λέει το όνομά της σαρώνει το αντικείμενο από τέσσερις διαφορετικές γωνίες απόστασης 90 μοιρών μεταξύ τους. Τα αποτελέσματα ήταν αρκετά ικανοποιητικά με περιθώρια βελτίωσης στην αυτοματοποίηση της διαδικασίας και στην παραγωγή των CAD μοντέλων.

Οι Abd-Raheem et. al. (2018) προτείνουν μια διάταξη αυτοματοποίησης της τρισδιάστατης σάρωσης με τη χρήση καμερών. Η τεχνική αυτή που ονομάζεται φωτογραμμετρία μετατρέπει δισδιάστατες φωτογραφίες σε τρισδιάστατο αντικείμενο. Για την διαδικασία της σάρωσης χρησιμοποιήθηκε αλγόριθμος, ο οποίος ορίζει τη λήψη φωτογραφιών με μεγάλη αλληλοεπικάλυψη και στην συνέχεια με ειδικό λογισμικό γίνεται η κατασκευή του τρισδιάστατου μοντέλου. Το αντικείμενο βρίσκεται σε σταθερη θέση με την κάμερα να μπορεί να περιστρέφεται γύρω του σε σταθερή απόσταση. Επίσης, η κάμερα έχει δυνατότητα περιστροφής για να αλλάξει η γωνία με την οποία κοιτάζει το αντικείμενο. Δημιουργήθηκε επίσης και UI για τον έλεγχο του συστήματος. Η ακρίβεια της σάρωσης επιδέχεται βελτιώσεις ειδικά σε ημιδιαφανή και γυαλιστερά αντικείμενα.

Οι Linder et. al. (2015) προτείνουν την χρήση σερβοκινητήρων για αντικατάσταση των βηματικών σε υπάρχουσα δομή σάρωσης με την χρήση σαρωτή λέιζερ. Με τη χρήση σερβοκινητήρων η κίνηση του σαρωτή από διακριτές θέσεις γίνεται συνεχόμενη μειώνοντας τον συνολικό χρόνο της διαδικασίας. Τα αποτελέσματα και η ακρίβεια της σάρωσης εξαρτώνται από τα χαρακτηριστικά των κινητήρων. Θεωρείται ότι η χρήση hi-end κινητήρων θα βελτιώσει κατά πολύ το αποτέλεσμα της σάρωσης. Η θεώρηση αυτή μένει να εξεταστεί.

Οι Wagner et. al (2016) επιτυχώς δημιούργησαν μια διεργασία τρισδιάστατης σάρωσης με τη χρήση δύο συνεργαζόμενων ρομποτικών βραχιόνων. Ο ένας φέρει σαρωτή λέιζερ 2D και ο δεύτερος το κομμάτι προς σάρωση. Η χρήση δυο συνεργαζόμενων ρομπότ προσφέρει οπτική επαφή του λέιζερ με περισσότερα σημεία της επιφάνειας του κομματιού. Η κίνηση των δυο ρομπότ μπορεί να γίνει ταυτόχρονα. Η διαδικασία δεν είναι πλήρως αυτοματοποιημένη καθώς ο χειριστής θα πρέπει να επιλέξει τις θέσεις σάρωσης.

Οι Huidaib et al. (2018) προτείνουν μια έξυπνη και οικονομική λύση για την τρισδιάστατη σάρωση. Προσαρτούν τον σαρωτή σε ένα υφιστάμενο τηλεχειριζόμενο ρομπότ αμαξίδιο. Ο σαρωτής αποτελείται από μια κανονική κάμερα και μια Time of flight Camera για τον προσδιορισμό του βάθους. Το ρομπότ αμαξίδιο που επέλεξαν είναι το Festo robotino, το οποίο έχει ενσωματωμένη λειτουργία αποφυγής εμποδίων και είναι εύκολο στην χρήση και τον προγραμματισμό. Η πρόταση ανήκει στην κατηγορία της απομακρυσμένης σάρωσης, με την ανακατασκευή του τρισδιάστατου μοντέλου να γίνεται στον χώρο ελέγχου του ρομπότ.

Οι Reinhart et. al. (2009). προτείνουν ένα σύστημα αυτόματου προγραμματισμού ρομπότ, στο οποίο είναι προσαρτημένο τρισδιάστατος σαρωτής με σκοπό τον ποιοτικό έλεγχο κομματιών παραγωγής. Προσεγγίζουν το ζήτημα δημιουργώντας αλγορίθμους οι οποίοι με την σειρά τους υπολογίζουν collision free paths για τον σαρωτή βάση του 3D μοντέλου των κομματιών προς σάρωση.

Η στρατηγική της σάρωσης μπορεί να χωριστεί σε δύο βασικές κατηγορίες την model based και την non-model based.(Khalfaoui et al. 2012, Banta et al. 2000) Οι στρατηγικές βασισμένες στην model-based προσέγγιση προσπαθούν να ορίσουν τον ελάχιστο αριθμό απαιτούμενων όψεων για επιτυχημένη σάρωση. (Wagner et al . 2016)

Οι Tarbox et al. (1995) ανέπτυξαν μια διαδικασία για την επιλογή όψεων βασισμένη σε αισθητήρες με κίνηση πανω σε εικοσάεδρο. Η προσσέγιση αυτή βασιζεται σε έναν Πίνακα Μετρησιμότητας (Measurability Matrix) σε αντίθεση με τις non-model based πρακτικές που συνηθως στηρίζονται σε ογκομετρικές μεθόδους

Οι Connolly et al. (1985) πρότειναν δύο αλγόριθμους, οι οποίοι αναπαριστούν το αντικείμενο μέσω οκτάδων. Το κάθε στοιχείο από αυτές μπορεί να έχει τιμή ‘κενό’, ‘πλήρες’, ή ‘μή ορατό’. Ο συνδυασμός αυτών των δύο αλγορίθμων μπορεί να ορίσει το διάνυσμα κατεύθυνσης των όψεων.

Οι Banta et al. (2000) πρότειναν μια γεωμετρική μέθοδο, η οποία θα αναφέρεται σε ένα πλέγμα με τα στοιχεία του να παίρνουν τιμές ‘seen’ και ‘unseen’.

Οι Maver et al. (1993) δημιούργησαν μια μέθοδο, όπου γίνεται χρήση των ακμών του αντικειμένου, με τις επιφάνειες να αναγνωρίζονται ως πολύγωνα καθορίζοντας μη προσβάσιμες οπτικά περιοχές από κάθε όψη.

Οι Olague et al. (1998) προτείνουν στρατηγική σάρωσης με βάση ένα σύστημα καμέρων που έχει τοποθετηθεί περιμετρικά του αντικειμένου. Οι κάμερες αυτές λαμβάνουν υπ όψιν την κλίση της κάθε επιφάνειας και τις περιοχές μη ορατότητας ιδιαίτερα για πολύπλοκες γεωμετρίες.

Οι Borangiu et al. (2008) ανέπτυξαν μια διαδικασία, στην οποία το αντικείμενο σαρώνεται μια φορά από προδιαγεγραμμένη διαδρομή και μια δεύτερη από δεδομένα που προκύπτουν από την πρώτη. Η πρώτη σάρωση είναι μια πιο rough σάρωση που θα δώσει βασικά στοιχεία της γεωμετρίας αλλά και τις ΄κρυφές΄ περιοχές του αντικείμενου με την δέυτερη να δίνει περισσότερη λεπτομέρεια και πληροφορία.

Ένας τρόπος υπολογισμού της επόμενης όψης σάρωσης προτάθηκε από τους Pito et al. (1995), οι οποίοι χρησιμοποίησαν range images και διαχώρισαν την επιφάνεια του αντικειμένου σε ορατή και μη. Αξιοποιώντας αυτές τις πληροφορίες μπορούσαν να ορίσουν την επόμενη όψη σάρωσης.

## 1.4 Δομή Εργασίας

Στο παρόν πρώτο κεφάλαιο γίνεται μια παρουσίαση της τρισδιάστατης σάρωσης και του θέματος της εργασίας. Γίνεται επίσης αναφορά σε αντίστοιχες προσπάθειες που έχουν γίνει.
Στο δεύτερο κεφάλαιο γίνεται παρουσίαση των στοιχείων δημιουργίας, μεταφοράς και ελέγχου κίνησης. Θα αναφερθούμε σε όλα τα στοιχεία που έχουμε συμπεριλάβει στην πρόταση αυτοματοποίησης.
Το τρίτο κεφάλαιο της εργασίας παρουσίαζει τη δημιουργία του CAD μοντέλου μέσω του λογισμικού SOLIDWORKS. Παρουσιάζονται τα επιμέρους υποσυστήματα και οι επιλογές που έγιναν κατά τη διάρκεια της σχεδίασης.
Στο τέταρτο κεφάλαιο ασχολούμαστε με το software του συστήματος. Εκεί θα δούμε τον προγραμματισμό του μικροελεκγτή για τον έλεγχο των κινητήρων του συστήματος αλλα και τη δημιουργία GUI με τη βοήθεια του Microsoft Visual Studio.
Η εργασία ολοκληρώνεται με συμπεράσματα και προτάσεις για βελτίωση.

# 2 Στοιχεία Δημιουργίας ελέγχου και μετάδοσης κίνησης

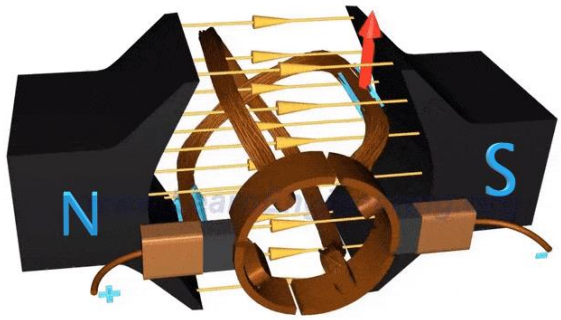
## 2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στους ηλεκτροκινητήρες και τον τρόπο λειτουργίας τους. Θα σταθούμε πιο πολύ στους βηματικούς κινητήρες καθώς αυτοί επιλέχθηκαν για τον σχεδιασμό του συστήματος. Θα αναφερθούμε επίσης στους τρόπους μετάδοσης της κίνησης. Τέλος θα γίνει μια εισαγωγή στους μικροελεγκτές και θα παρουσιαστεί ένα παράδειγμα ελέγχου βηματικού κινητήρα με μικροελεγκτή.

## 2.2 Κινητήρες

Οι κινητήρες είναι διατάξεις οι οποίες μετατρέπουν μια μορφή ενέργειας (χημική, θερμική, ηλεκτρική) σε μηχανική, συνήθως σε κυκλική-περιστροφική κίνηση. Στο κεφάλαιο αυτό θα ασχοληθούμε με τους ηλεκτρικούς κινητήρες, τις κατηγορίες τους, τον τρόπο λειτουργίας τους αλλά και τις προτεινόμενες χρήσεις για κάθε κατηγορία. Θα αναφερθούμε, επίσης, στο πρότυπο NEMA, το οποίο είναι ένα πρότυπο διαστασιολόγησης κυρίως βηματικών κινητήρων.

Οι ηλεκτρικοί κινητήρες χωρίζονται σε δύο βασικές κατηγορίες, στους κινητήρες συνεχούς και εναλλασσόμενου ρεύματος. Οι κινητήρες συνεχούς χωρίζονται σε δύο κατηγορίες, τους σύγχρονους και τους ασύγχρονους ανάλογα με το αν ο ρότορας περιστρέφεται σύγχρονα με την αλλαγή του μαγνητικού πεδίου μέσα στον κινητήρα η όχι. Οι κινητήρες συνεχούς επίσης κατηγοριοποιούνται σε αυτούς που έχουν ψήκτρες (καρβουνάκια), σε αυτούς χωρίς και στους βηματικούς κινητήρες. Μια ακόμα κατηγορία είναι οι σερβοκινητήρες, οι οποίοι χρησιμοποιούν σύστημα ανάδρασης για τον έλεγχο της θέσης του ρότορα. Σε όλες τις παραπάνω κατηγορίες θα αναφερθούμε στη συνέχεια δίνοντας μεγαλύτερη έμφαση στους βηματικούς κινητήρες.




Εικόνα 2.1 Σχηματική απεικόνιση ηλεκτρικού κινητήρα
Πηγή:electronics-tutorial.ws

### 2.2.1 Κινητήρες συνεχούς ρεύματος (DC motors) με ψήκτρες.

Οι κινητήρες συνεχούς ρεύματος είναι μηχανισμοί, οι οποίοι μετατρέπουν το συνεχές ηλεκτρικό ρεύμα σε μηχανική κίνηση. Αποτελούνται από δύο κύρια μέρη: τον στάτορα που είναι το σταθερό μέρος και τον ρότορα που είναι το κινητό.

Βασική αρχή λειτουργίας του κινητήρα συνεχούς είναι η παραγωγή περιστρεφόμενου μαγνητικού πεδίου από τον στάτορα, το οποίο οδηγεί τον ρότορα σε περιστροφή. Ο στάτορας αποτελείται από ένα σύστημα μαγνητών ενώ ο ρότορας από περιελίξεις συρμάτων που δημιουργούν πηνία. Λόγο της ροής ηλεκτρικού ρεύματος από τα πηνία δημιουργείται ηλεκτρομαγνητική δύναμη (βλ. Νόμο Lorenz) η οποία σε συνδυασμό με τους μαγνήτες του στάτορα οδηγεί τον ρότορα σε περιστροφική κίνηση. Οι στάτορες συνήθως έχουν περιελίξεις χαλκού σε διάφορες ακτινικές θέσεις προκειμένου να υπάρχει μια σχεδόν σταθερή ροπή καθ' όλη την διάρκεια της περιστροφής παράγοντας μια –σχεδόν- ομαλή και σταθερή κίνηση.

Εικόνα 2.2 Ηλεκτρικός κινητήρας συνεχούς ρεύματος
Πηγή:Cablemotors.gr

### 2.2.2 Κινητήρες συνεχούς χωρίς ψήκτρες (brushless)

Οι κινητήρες χωρίς ψήκτρες χρησιμοποιούν αισθητήρες και ένα ψηφιακό σύστημα ή μια πλακέτα κυκλώματος για την εναλλαγή της πολικότητας. Τα σήματα από τους αισθητήρες επεξεργάζονται στην πλακέτα η αλλιώς driver και έτσι ελέγχεται με ακρίβεια η σωστή θέση αλλαγής της πολικότητας, καθώς ο ρότορας περιστρέφεται. Στη βιβλιογραφία οι κινητήρες χωρίς ψήκτρες αναφέρονται ως η εξέλιξη των πρώτον DC κινητήρων με ψήκτρες. Σε αυτούς, πέρα από τον έλεγχο της ταχύτητας και της επιτάχυνσης, μπορούμε να έχουμε και έλεγχο θέσης κάτι που στους κινητήρες με ψήκτρες είναι δυνατό μόνο με την χρήση encoder. Επιπλέον, οι ψήκτρες είναι ένα εξάρτημα του κινητήρα, το οποίο με τη χρήση φθείρεται και θέλει αντικατάσταση ενώ με τους κινητήρες χωρίς ψήκτρες η συντήρηση είναι λιγότερο απαιτητική.

### 2.2.3 Βηματικοί κινητήρες (stepper motors)

Οι βηματικοί κινητήρες ή κινητήρες σταδιακών βημάτων (stepper motors) ανήκουν στην κατηγορία των DC κινητήρων. Χαρακτηριστικό των κινητήρων αυτών είναι ότι περιστρέφονται σε διακριτά βήματα (steps), η γωνία των οποίων είναι σταθερή και το μέγεθος της εξαρτάται από τα κατασκευαστικά στοιχεία του κινητήρα. Οι κινητήρες αυτοί

δέχονται σαν είσοδο ψηφιακούς παλμούς τάσης. Το πλήθος και η συχνότητα των παλμών αυτών ελέγχουν και οδηγούν την περιστροφή του κινητήρα. Ο έλεγχος της κίνησης, της ταχύτητας και της θέσης μπορεί να γίνει με σχετικά υψηλή ακρίβεια, γι' αυτό και τέτοιου είδος κινητήρες είναι η συνήθης επιλογή για εφαρμογές ρομποτικής. Άλλο προτέρημα τους είναι και το σχετικά χαμηλό κόστος.

Ένα ακόμα χαρακτηριστικό που κάνει τους βηματικούς κινητήρες συνήθη επιλογή για ρομποτικές διατάξεις είναι η ύπαρξη ροπής συγκράτησης. Η ροπή αυτή κρατάει σταθερό τον ρότορα σε συγκεκριμένη θέση. Έτσι, πέρα από μετατόπιση σε ένα σημείο ο κινητήρας αναπτύσσει και ροπή για τη διατήρηση της θέσης αυτής.

Η ακρίβεια κίνησης των κινητήρων εξαρτάται από το μέγεθος του βήματος, δηλαδή την ελάχιστη μετακίνηση του ρότορα. Συνήθως, στο εμπόριο οι βηματικοί κινητήρες έχουν βήμα από 1 έως 7.5 μοίρες.

Επειδή ακριβώς τα ηλεκτρομηχανικά στοιχεία αυτά δέχονται ψηφιακούς παλμούς σαν είσοδο η χρήση ψηφιακών κυκλωμάτων είναι απαραίτητη για τον έλεγχό τους.

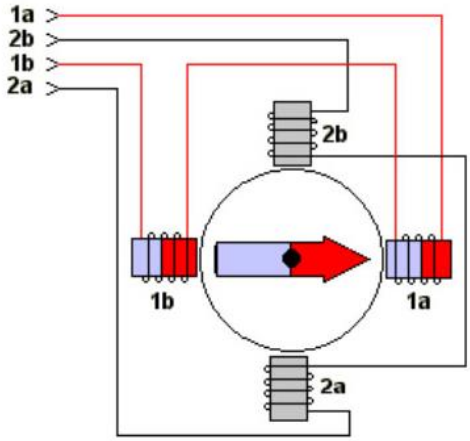
Εικόνα 2.3 Βηματικός κινητήρας
Πηγή: TME.eu

***Είδη βηματικών κινητήρων***

Οι βηματικοί κινητήρες μπορούμε να πούμε ότι έχουν δομή παρόμοια με αυτή των κινητήρων συνεχούς ρεύματος χωρίς ψήκτρες. Αποτελούνται από το σταθερό μέρος, τον στάτορα στον οποίο υπάρχουν ηλεκτρομαγνήτες και έναν ρότορα ο οποίος φέρει και αυτός μαγνήτη. Ανάλογα με το αν ο μαγνήτης στον ρότορα είναι μόνιμος η μεταβλητός χωρίζουμε τους κινητήρες σε βηματικούς κινητήρες μόνιμου μαγνήτη (permanent magnet stepping motor (PM)) και σε βηματικούς κινητήρες μεταβλητής μαγνητικής αντίδρασης (variable reluctance stepper motor (VR)). Τέλος, υπάρχει και ο υβριδικός βηματικός κινητήρας (Hybrid stepper motor (HB) ) ο οποίος συνδυάζει τα κύρια χαρακτηριστικά των δύο προηγούμενων.

Οι βηματικοί κινητήρες μπορούν να είναι είτε διπολικοί, που σημαίνει ότι χρειάζονται δύο πηγές ενέργειας ή μια πηγή με δυνατότητα αλλαγής πολικότητας, είτε μονοπολικοί που σημαίνει αντίστοιχα ότι χρειάζονται μια πηγή.

***Αρχή λειτουργίας διπολικού κινητήρα.***




Εικόνα 2.4 Σχηματικό διάγραμμα διπολικού κινητήρα

Ας υποθέσουμε ότι σε ένα βηματικό κινητήρα υπάρχουν 4 ανεξάρτητα πηνία στερεωμένα στον στάτορα με γωνία 90 μοιρών μεταξύ τους όπως φαίνεται και από το παραπάνω σχήμα(εικ 2.4). Αν τα πηνία ενεργοποιούνται με κυκλική σειρά π.χ. 1Α-2Α-1Β-2Β-1Α τότε ό ρότορας θα περιστραφεί κατά 360 μοίρες με ωρολογιακή φορά. Αν η σειρά είναι αντίστροφη τότε ο ρότορας θα ακολουθήσει ανθωρολογιακή φορά. Η λειτουργία της ενεργοποίησης μόνο του ενός πηνίου κάθε φορά συνήθως αποφεύγεται επειδή παράγει τη μισή ροπή από την ονομαστική ισχύ του κινητήρα και προτιμάται μόνο σε περιπτώσεις όπου θέλουμε να έχουμε εξοικονόμηση ενέργειας.

Πιο συνηθισμένη λειτουργία είναι αυτή της ενεργοποίησης αντιδιαμετρικών πηνίων. Για παράδειγμα, αν έχουμε τον κινητήρα (εικ 2.4) και θέλουμε ο ρότορας να εκτελέσει μια πλήρη περιστροφή τα πηνία θα ενεργοποιηθούν με την ακόλουθη σειρά : 1Α-1Β, 2Α-2Β, 1Α-1Β, 2Α-2Β, 1Α-1Β.

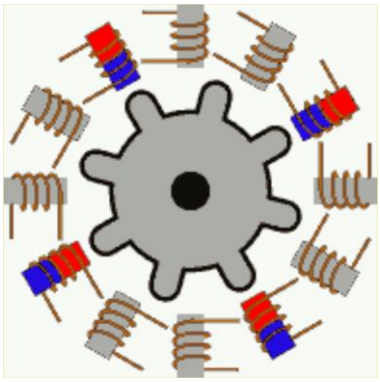
***Λειτουργία μισού βήματος.***

Στην λειτουργία μισού βήματος έχουμε διπλάσια ακρίβεια στον έλεγχο της θέσης του ρότορα απ’ ο,τι στη λειτουργία ολόκληρου βήματος. Για τη δημιουργία μισού βήματος (βλ. εικ.3.4) όταν ο ρότορας είναι στην οριζόντια θέση 1Β-1Α η επόμενη φάση είναι η 1Α-2Α. Έτσι, ο ρότορας θα δείχνει ακριβώς ανάμεσα στα δύο πηνία σχηματίζοντας γωνία 45 μοιρών από το καθένα. Θα έχει διανύσει δηλαδή 45 μοίρες ή μισό βήμα σε σχέση με την προηγούμενη συμπεριφορά. Για την πλήρη περιστροφή θα έχουμε την ακολουθία :2Α-2Β, 2Α-1Β,1Β-1Α,1Β-2Β, 2Β-2Α,2Β-1Α, 1Α-1Β.

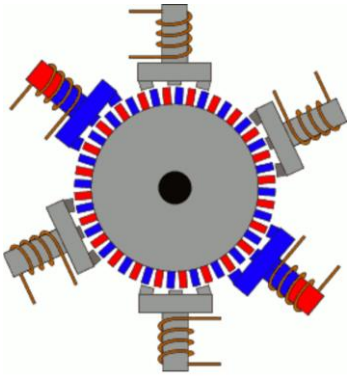
Τέλος, υπάρχει και η λειτουργία micro stepping. Στη λειτουργία αυτή στέλνονται στα πηνία κυμματομορφές, παρόμοιες με την ημιτονοειδή αντί για ορθογωνικούς παλμούς. Με αυτόν τον τρόπο επιτυγχάνεται ομαλότερη και ακριβέστερη περιστροφή του ρότορα, ο οποίος μπορεί να περιστρέφεται συνεχώς.

***Σχηματικές απεικονίσεις μεταβλητού μαγνήτη και υβριδικού βηματικού κινητήρα.***

Ο βηματικός κινητήρας μεταβλητής μαγνητικής αντίδρασης δεν έχει μόνιμο μαγνήτη στο ρότορα. Ο ρότορας αποτελείται από μαλακό σίδηρο σε σχήμα δίσκου με δόντια (σαν γρανάζι). Ο στάτορας έχει περισσότερα από 4 πηνία, τα οποία ενεργοποιούνται σε αντιδιαμετρικά ζευγάρια. Η έλλειψη μόνιμου μαγνήτη έχει ως επίπτωση την μείωση της ονομαστικής ροπής, αλλά δημιουργεί το πλεονέκτημα της ελεύθερης κίνησης του ρότορα (χωρίς θέσεις) όταν τα πηνία δεν διαρρέονται από ρεύμα



Εικόνα 2.5 Σχηματική απεικόνιση κινητήρα μεταβλητού μαγνήτη



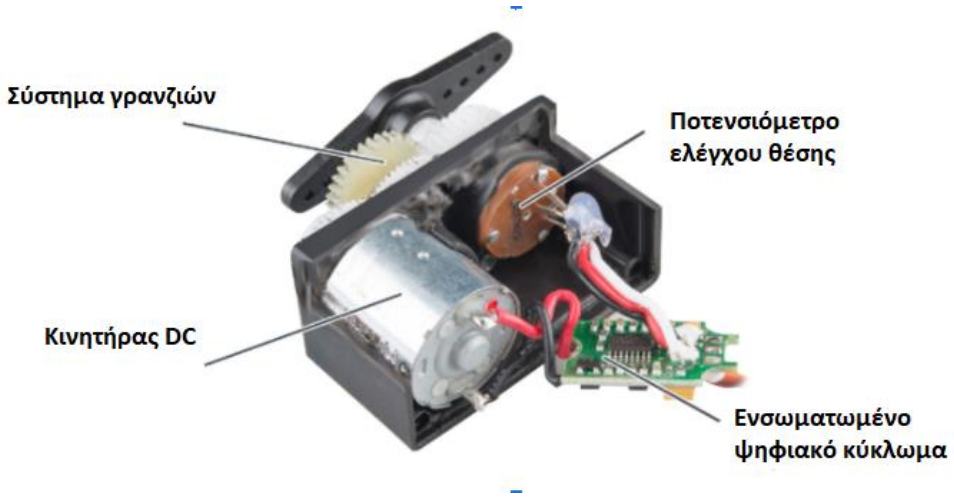
Εικόνα 2.6 Σχηματική απεικόνιση υβριδικού βηματικού κινητήρα

Ο υβριδικός βηματικός κινητήρας συνδυάζει τα χαρακτηριστικά των βηματικών κινητήρων μόνιμου μαγνήτη και μεταβλητής μαγνητικής αντίδρασης με αποτέλεσμα την παραγωγή περισσότερης ροπής με πολύ μικρά βήματα περιστροφής. Εκτός από μεγάλη ακρίβεια στην κίνηση οι κινητήρες αυτοί αναπτύσσουν υψηλές ταχύτητες με μειονέκτημα, όμως, το μεγάλο κόστος κατασκευής.

### 2.2.4 Σερβοκινητήρες (Servo motor)

Οι σερβοκινητήρες παρέχουν μεγάλο έλεγχο θέσης και μεγάλη ακρίβεια λόγω του κλειστού βρόχου ανάδρασης που χρησιμοποιούν για τον έλεγχο της ταχύτητας και της θέσης. Οι σερβοκινητήρες χρησιμοποιούνται ευρέως σε ρομποτικές διατάξεις που θέλουμε να έχουμε μεγάλη ακρίβεια θέσης. Είναι, επίσης, συνηθισμένη η εφαρμογή τους σε CNC συστήματα. Μειονεκτούν σε σχέση με τους βηματικούς κινητήρες κυρίως στο θέμα κόστους, καθώς ένας σερβοκινητήρας κοστίζει αρκετά παραπάνω από έναν βηματικό κινητήρα. Παρουσιάζουν όμως καλύτερη συμπεριφορά και αυξημένη ροπή σε υψηλές ταχύτητες σε αντίθεση με τους βηματικούς κινητήρες που προτείνονται για εφαρμογές χαμηλών ταχυτήτων.

Ο σερβοκινητήρας αποτελείται απο έναν κινητήρα συνεχούς ρεύματος (DC) με μόνιμο μαγνήτη στον στάτη, ένα σύστημα γραναζιών για τη μείωση των στροφών και την αύξηση της ροπής (μειωτήρα), ποτενσιόμετρο το οποίο λειτουργεί και σαν αισθητήρας θέσης και από ένα ηλεκτρικό κλειστό κύκλωμα ελέγχου το οποίο χρησιμοποιείται για τον έλεγχο και την διόρθωση της κίνησης και θέσης του ρότορα.




Εικόνα 2.7 Επιμέρους στοιχεία σερβοκινητήρα

Οι σερβοκινητήρες συνήθως έχουν τρία καλώδια για την σύνδεση τους. Ένα που συνδέεται με την πηγή τάσης, ένα με τη γείωση και ένα καλώδιο εισόδου σήματος.

### 2.2.5 Πρότυπο διαστασιολόγησης NEMA

Για τη χρήση κινητήρων πέραν από την επιλογή χαρακτηριστικών λειτουργίας όπως ροπή, ακρίβεια, μέγιστη ταχύτητα κ.α. σημαντική είναι και η διαστασιολόγηση. Αν η διαστασιολόγηση ενός κινητήρα ακολουθεί κάποια συγκεκριμένα πρότυπα τότε είναι εύκολη η επιλογή εξαρτημάτων που θα συνεργάζονται με τον κινητήρα, όπως για παράδειγμα βάσεις στήριξης. Ένα από τα πιο καθιερωμένα πρότυπα διαστασιολόγησης κινητήρων είναι το πρότυπο NEMA (National Electrical Manufacturers Association), το οποίο καθιερώθηκε από

την Ένωση ηλεκτρονικών κατασκευαστών των Η.Π.Α. Το πρότυπο αυτό περιλαμβάνει διαστάσεις πλαισίου συμπεριλαμβανομένου των οπών, διαστασιολόγηση βάσεων κινητήρα, διάμετρο και μήκος αξόνων. Οι κατηγορίες NEMA όσον αφορά στη διαστασιολόγηση του πλαισίου του κινητήρα έχουν την μορφή NEMA + αριθμός, όπου ο αριθμός δηλώνει την επιφάνεια του προσώπου του πλαισίου σε τετραγωνικές ίντσες. Για παράδειγμα ένας κινητήρας NEMA 14 έχει επιφάνεια προσώπου 1,4 τετραγωνικές ίντσες. Το πρότυπο αυτό αφορά κυρίως κινητήρες από 8 έως 42 τετραγωνικές ίντσες. Πάνω από τις 42 ίντσες δεν υπάρχει κάποιο πρότυπο NEMA σχετικά με τη διαστασιολόγηση. Αυτό συμβαίνει γιατί οι εφαρμογές που χρειάζονται κινητήρες τέτοιου μεγέθους είναι συγκεκριμένες και συνήθως σε τέτοιες περιπτώσεις γίνεται μια εξειδικευμένη διαστασιολόγηση.


Εικόνα 2.8 Βηματικοί κινητήρες διαφόρων μεγεθών NEMA
Πηγή: Polulu.com

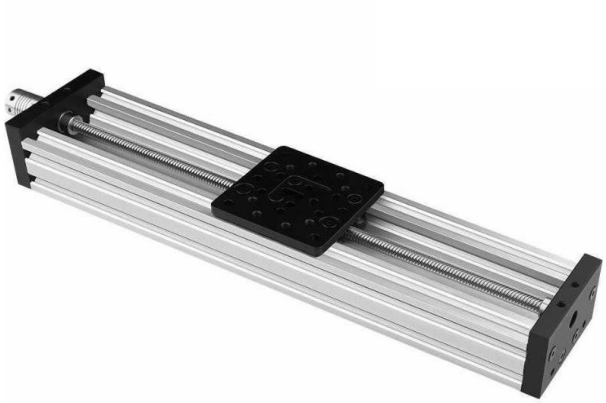
## 2.3 Μετάδοση Κίνησης

Στο προηγούμενο κεφάλαιο είδαμε πως γίνεται η μετατροπή της ηλεκτρικής ενέργειας σε κινητική μέσα από την χρήση ηλεκτροκινητήρων. Σε αυτό το κεφάλαιο θα παρουσιάσουμε

τρόπους μετάδοσης αυτής της κίνησης και τη μετατροπή της σε ευθύγραμμη και περιστροφική. Οι ακόλουθοι τρόποι χρησιμοποιήθηκαν και στον σχεδιασμό του τελικού συστήματος της παρούσας εργασίας.

### 2.3.1 Γραμμικοί οδηγοί (linear actuators)

Για τη μετατροπή της περιστροφικής κίνησης εξόδου ενός κινητήρα σε ευθύγραμμη πολύ συχνά χρησιμοποιούνται γραμμικοί οδηγοί. Οι γραμμικοί οδηγοί (linear actuators) αποτελούνται από ένα σταθερό συνήθως κομμάτι στο οποίο εδράζονται οι ράγες, οι οποίες ορίζουν την γραμμική κίνηση, έναν κοχλία ανάμεσα από τις δυο ράγες, ο οποίος περιστρέφεται με την χρήση ενός ηλεκτροκινητήρα και ένα βαγονέτο, το οποίο διαπερνάται από τον κοχλία. Με την περιστροφή του κοχλία το βαγονέτο κινείται ευθύγραμμα με κατεύθυνση που ορίζεται από τη φορά περιστροφής του κοχλία. Κατ' επέκταση η ταχύτητα και η επιτάχυνση του βαγονέτου εξαρτάται από την περιστροφική ταχύτητα και επιτάχυνση του κοχλία.



Εικόνα 2.9 Γραμμικός οδηγός

Υπάρχουν δύο κύριες κατηγοριοποιήσεις των γραμμικών οδηγών που έχουν να κάνουν με το είδος του βαγονέτου:

***Lead screw - βαγονέτο με σπείρωμα.***

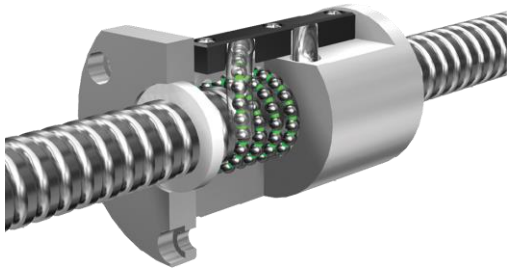
Στην κατηγορία αυτή το βαγονέτο μέσα από το οποίο περνάει ο κοχλίας, έχει εσωτερικό σπείρωμα ίδιου βήματος με τον κοχλία. Το βαγονέτο στηρίζεται σε ράγες έτσι ώστε να μην επιτρέπεται η περιστροφή του. Με αυτόν τον τρόπο, όταν περιστρέφεται ο κοχλίας το βαγονέτο "ακολουθεί" το σπείρωμα, κινούμενο ευθύγραμμα ανάλογα με τη φορά περιστροφής του κοχλία .

***Ball Screw - σφαιρικής βίδας***

Αντίστοιχο του προηγούμενου είναι και το βαγονέτο με σφαιρική βίδα (ball screw). Σε αυτό το βαγονέτο υπάρχουν μπίλιες πάνω στο σπείρωμα, οι οποίες κινούνται με παρόμοιο τρόπο με τα ρουλεμάν. Έτσι, όταν περιστρέφεται ο κοχλίας κινούνται και οι μπίλιες εντός του βαγονέτου μετακινώντας το ευθύγραμμα προς τη διεύθυνση που ορίζει η φορά περιστροφής του κοχλία. Τα βαγονέτα με ρουλεμάν σε σύγκριση με αυτά με σπείρωμα παρουσιάζουν

πολύ μικρότερες τριβές και ελάχιστη αβεβαιότητα θέσης. Το κόστος τους συγκριτικά με τα βαγονέτα σπειρώματος είναι μεγαλύτερο.

Εικόνα 2.10 Lead screw



Εικόνα 2.11 γραμμικός οδηγός σφαιρικής βίδας

### 2.3.2 Ιμάντες

Ένας άλλος αρκετά διαδεδομένος τρόπος μετάδοσης κίνησης, τόσο ευθύγραμμης όσο και περιστροφικής, είναι η χρήση ιμάντων. Για την ευθύγραμμη κίνηση συνήθως χρησιμοποιούνται δυο τροχαλίες στα άκρα της διαδρομής. Ο ένας τροχαλίας συνδέεται με τον κινητήρα παρέχοντας την περιστροφική κίνηση στο σύστημα ενώ ο δεύτερος υπάρχει για να βοηθήσει την κίνηση του ιμάντα. Το βαγονέτο είναι προσαρτημένο σταθερά σε κάποιο σημείο του ιμάντα έτσι ώστε να μετακινείται μαζί με αυτόν. Για τη χρήση ιμάντα απαιτούνται και τεντωτήρες, οι οποίοι διατηρούν την απαιτούμενη τάση στα άκρα του ιμάντα προκειμένου η κίνηση να πραγματοποιείται ομαλά. Υπάρχουν διάφορα είδη ιμάντα ανάλογα με τη γεωμετρία της επιφάνειας επαφής, το καθένα εκ των οποίων με τα δικά του χαρακτηριστικά. Υπάρχουν λείοι ιμάντες που κινούνται λόγω της στατικής τριβής, βαθμωτοί ιμάντες οι οποίοι έχουν δόντια τα οποία εφαρμόζουν ακριβώς στον τροχαλία και ιμάντες με τραπεζοειδή διατομή. Η γεωμετρία των ιμάντων έχει γίνει αντικείμενο μελέτης ως προς το ποιες γεωμετρίες είναι κατάλληλες για διάφορες εφαρμογές και με ποιες έχουν μεγαλύτερη μετάδοση ισχύος.

Οι ιμάντες πολύ συχνά χρησιμοποιούνται και για να μεταφέρουν κίνηση από ένα γρανάζι σε άλλο. Η σχέση μετάδοσης καθορίζεται από την γεωμετρία των γραναζιών.

Εικόνα 2.12 Μετάδοση κίνησης με ιμάντα

### 2.3.3 Γρανάζια (Οδοντωτοί τροχοί)

Τα γρανάζια αποτελούν έναν συνηθισμένο και αρκετά παλιό τρόπο μετάδοσης κίνησης. Υπάρχουν αρκετοί τύποι γραναζιών: οι μετωπικοί οδοντωτοί τροχοί, οι ελικοειδής, οι κωνικοί κ.α. Τα γρανάζια χρησιμοποιούνται για την μετάδοση κίνησης και κυρίως για την δημιουργία σχέσης μετάδοσης μεταξύ κινητήρα και κινητού. Η σχέση μετάδοσης εξαρτάται από τον αριθμό δοντιών που έχουν τα εμπλεκόμενα γρανάζια. Τα γραναζια έχουν συνήθως μεγάλη διάρκεια ζωής και απαιτούν μικρή συντήρηση. Εμφανίζουν σχετικά θορυβώδη λειτουργία, έχουν υψηλό κόστος κατασκευής και η μετάδοση κίνησης γίνεται μη ελαστικά.


Εικόνα 2.13 Είδη γραναζιών

.

## 2.4 Έλεγχος κίνησης – μικροελεγκτές, οδηγοί κινητήρων

### 2.4.1 Εισαγωγή

Στον σύγχρονο κόσμο η χρήση ηλεκτρονικών συσκευών βρίσκεται παντού, από την καθημερινή ζωή μέχρι τις σύγχρονες βιομηχανικές εγκαταστάσεις. Οι ηλεκτρονικές συσκευές ποικίλουν σε πολυπλοκότητα και λειτουργία. Συναντάμε από τις πιο απλές οικιακές συσκευές μέχρι τις πιο σύνθετες σε αεροσκάφη και διαστημόπλοια. Σε όλες αυτές δομικό κομμάτι της λειτουργίας τους είναι ο έλεγχος. Τον σημαντικό αυτό ρόλο έρχονται να καλύψουν οι microcontrollers ή MCUs ή μικροελεγκτές.

Οι microcontrollers ουσιαστικά είναι μικροί υπολογιστές, οι οποίοι έχουν σχεδιαστεί και προγραμματιστεί για να εκτελούν μια συγκεκριμένη λειτουργία. Βασικά χαρακτηριστικά τους είναι το μικρό μέγεθος, η ανθεκτικότητα και το χαμηλό τους κόστος. Τα συστήματα, τα οποία χρησιμοποιούν microcontrollers συχνά αναφέρονται και ως ενσωματωμένα συστήματα ή embedded systems. Ένας ορισμός για τα ενσωματωμένα συστήματα θα μπορούσε να είναι ο εξής: “ Ένα σύστημα του οποίου η κύρια λειτουργία δεν είναι υπολογιστική, αλλά ελέγχεται από έναν υπολογιστή ενσωματωμένο σε αυτήν ”M.Jimenez et al .

### 2.4.2 Μικροελεγκτές

Οι μικροελεγκτές όπως αναφέραμε είναι ουσιαστικά μικροί υπολογιστές και αντίστοιχα αποτελούνται από επιμέρους υποσυστήματα. Τα βασικά υποσυστήματα τους είναι τα εξής:

***Μικροεπεξεργαστής*** - Επεξεργαστής ο οποίος είναι σχεδιασμένος για μικροελεγκτές

***Μνήμη*** - Σε κάποιους μικροελεγκτές η μνήμη είναι χωρισμένη σε μνήμη δεδομένων και μνήμη προγράμματος

***Ρολόι/μετρητής*** - Πολλοί μικροελεγκτές έχουν περισσότερα από ένα εσωτερικά ρολόγια ή μετρητές- timers/counters

***Digital I/O*** - Θύρες για την είσοδο και έξοδο ψηφιακών σημάτων

***Analog I/0*** - Οι περισσότεροι μικροελεγκτές έχουν ενσωματωμένο έναν μετατροπέα σήματος από αναλογικού σε ψηφιακό. Κάποιοι έχουν και μετατροπέα από ψηφιακό σε αναλογικό.

***Interface*** - Πρωτόκολλο επικοινωνίας για την σύνδεση με PC ώστε να γίνει ο απαραίτητος προγραμματισμός τους. Πολλές φορές συναντάμε θύρα USB ή σύνδεση ethernet, συνήθως σε μεγαλύτερους μικροελεγκτές.

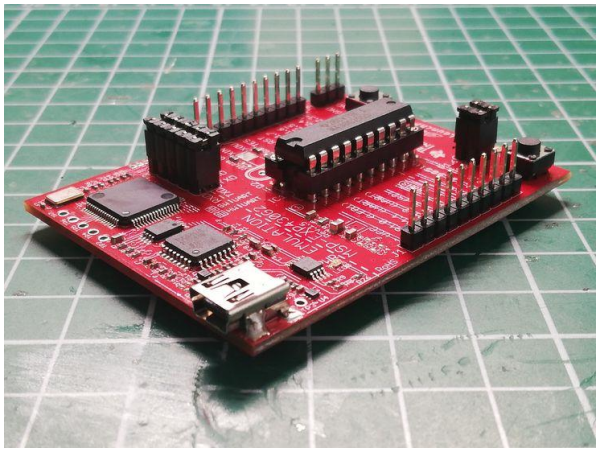
***Watchdog Timer*** - Ένα “χρονόμετρο’’ το οποίο ελέγχει αν ο μικροελεγκτής λειτουργεί και αν λειτουργεί σωστά.

Ο πρώτος μικροεπεξεργαστής παράχθηκε στα 1971 από την εταιρεία Intel και ήταν ο i4004. Στην αρχή όλα τα υποσυστήματα, όπως οι μνήμες και οι είσοδοι/έξοδοι, συνδέονταν εξωτερικά με τον επεξεργαστή. Η ανάγκη όμως για ολοκληρωμένα συστήματα τα οποία θα μπορούσαν να σταθούν αυτόνομα έφερε την δημιουργία των μικροελεγκτών. Έτσι, το 1974 η εταιρία Texas instruments έδωσε στην κυκλοφορία τον πρώτο μικροελεγκτή, τον TMS 1000.


Εικόνα 2.14 Ο πρώτος μικροεπεξεργαστής από την Intel i4004


Εικόνα 2.15 Ο πρώτος μικροελεγκτής TMS 1000 από την Texas Instruments

Ο μικροελεγκτής TMS 1000 είχε ενσωματωμένη μνήμη, λειτουργία read only, read and write memory και ρολόι. Από τότε οι μικροελεγκτές έχουν βελτιωθεί με την προσθήκη πολλών υποσυστημάτων και αυξημένης ταχύτητας. Μερικές από τις πιο γνωστές εταιρείες σήμερα στον χώρο τον μικροελεγκτών είναι οι: Texas Instruments, Silicon Labs, Toshiba, STMelectronics και Zilog μεταξύ άλλων.

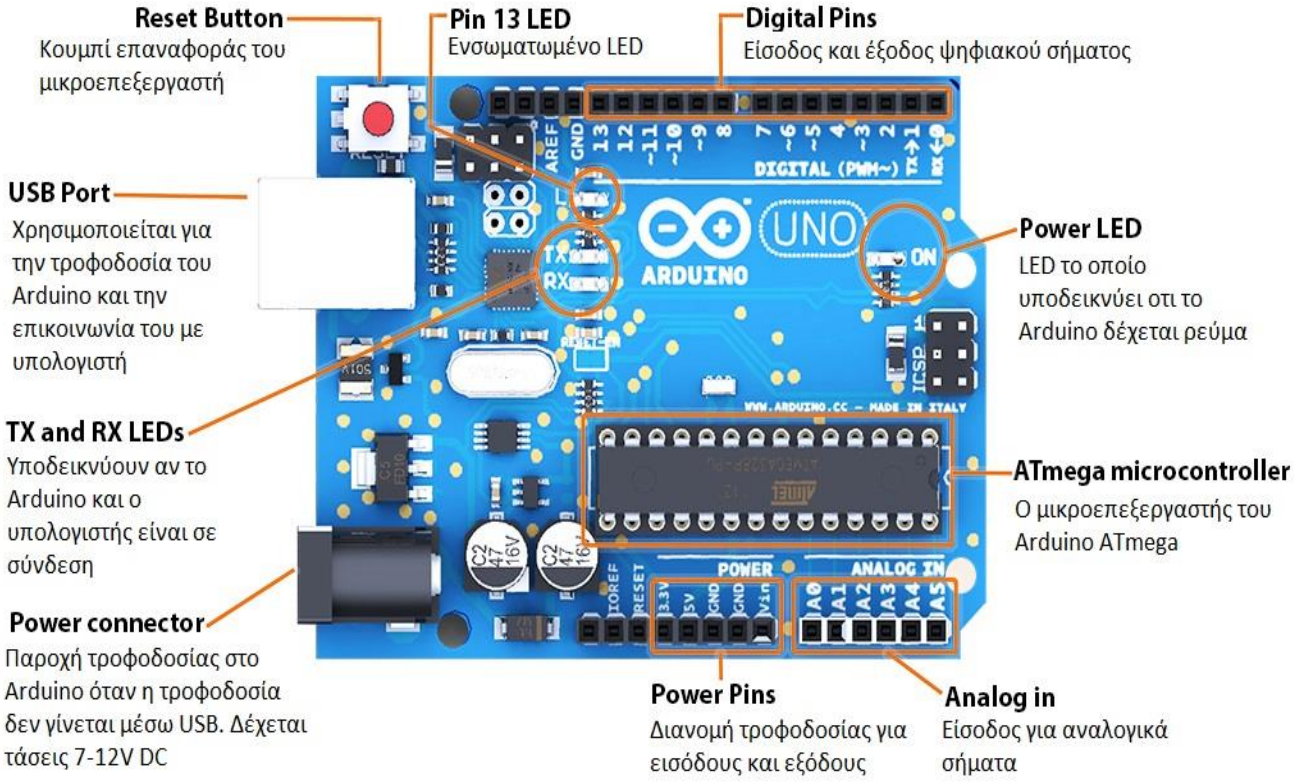


Εικόνα 2.16 Σύγχρονος μικροεπεξεργαστής από την Texas Instruments ο **MSP 430.**

### 2.4.3 Arduino

Στις αρχές του 2005 μια ομάδα από το Interaction Design Institute της Ivrea στην Ιταλία κατάφερε να δημιουργήσει μια πλατφόρμα που θα έκανε τον προγραμματισμό μικροελεγκτών πιο προσιτό στο ευρύ κοινό μειώνοντας το κόστος και την πολυπλοκότητα. Η νέα πλατφόρμα αποτελούνταν από ένα IDE, διάφορες βιβλιοθήκες για την ευκολία στον προγραμματισμό, μια πλακέτα τυπωμένου κυκλώματος (PCB) και έναν μικροεπεξεργαστή, τον ATmega168. Ονομάστηκε Arduino από την καφετέρια που έκανε η ομάδα τις συναντήσεις της.

Το Arduino είναι μια ανοικτού κώδικα ηλεκτρονική πλατφόρμα μικροελεγκτή hardware και Software. Η πλατφόρμα αυτή υλοποιεί την Processing Language χρησιμοποιώντας ως γλώσσα προγραμματισμού την C++. Η φυσική πλακέτα του Arduino είναι ένα πλήρες σύστημα μικροελεγκτή με όλα τα υποσυστήματα που χρειάζονται για να έχουμε έναν ολοκληρωμένο έλεγχο.




Εικόνα 2.17 Δομικά στοιχεία του Arduino Uno

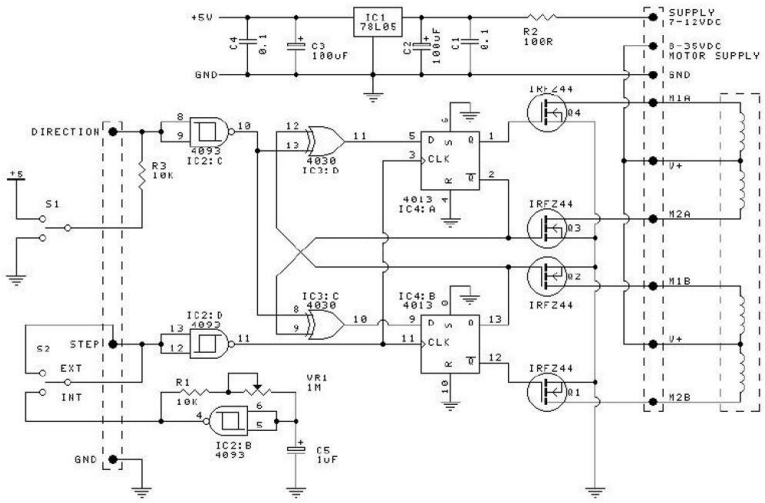
Το Arduino είναι μια πλατφόρμα, η οποία έχει υιοθετηθεί από πλήθος δημιουργών, από φοιτητές, ερευνητές, προγραμματιστές και επαγγελματίες μέχρι καλλιτέχνες και ερασιτέχνες. Με την χρήση του Arduino έχουν δημιουργηθεί αμέτρητα projects και πρωτότυπα από έλεγχο μηχανολογικών κατασκευών, 3d printing projects μέχρι projects IoT και wearables.

***Έλεγχος βηματικού κινητήρα με μικροελεγκτή.***

Για τον έλεγχο ενός βηματικού κινητήρα με μικροελεγκτή απαραίτητη είναι η ενδιάμεση σύνδεση του με έναν βηματικό οδηγό (stepper driver). Ο οδηγός αυτός θα μεταφράσει το σήμα εξόδου από την ψηφιακή έξοδο του microcontroller σε σήμα κατάλληλο για τον κινητήρα έτσι ώστε να πραγματοποιήσει την κατάλληλη κίνηση.

### 2.4.5 Οδηγός βηματικού κινητήρα -Stepper Driver

Οι βηματικοί οδηγοί είναι σχετικά απλά κυκλώματα, τα οποία μεταφράζουν το ψηφιακό σήμα σε παλμό για τον έλεγχο και την δημιουργία κίνησης στους βηματικούς κινητήρες. Όπως είδαμε σε προηγούμενο κεφάλαιο (2.2.3) οι βηματικοί κινητήρες κινούνται σε βήματα. Για να γίνει αυτό πρέπει να ενεργοποιούνται διαδοχικά οι πόλοι του κινητήρα ούτως ώστε να πραγματοποιηθεί η περιστροφή. Η λειτουργία ενός βηματικού οδηγού είναι με απλά λόγια η διαδοχική ενεργοποίηση αυτών των πόλων προκειμένου να γίνει η περιστροφή. Η φορά και η ταχύτητα του κινητήρα ορίζεται από τον χρήστη.




Εικόνα 2.18 Διάγραμμα κυκλώματος απλού βηματικού οδηγού
Πηγή : electronics-diy.com

Υπάρχει πλήθος βηματικών οδηγών ανάλογα με την πολυπλοκότητα του κυκλώματος, τις τάσεις και τα ρεύματα που μπορούν να διαχειριστούν, αλλά και τις λειτουργίες που υποστηρίζουν όπως το microstepping. Εδώ να σημειώσουμε ότι η απόδοση ενός κινητήρα εξαρτάται σε μεγάλο βαθμό από τον βηματικό οδηγό που χρησιμοποιούμε.

### 2.4.6 Παράδειγμα ελέγχου με βηματικού κινητήρα

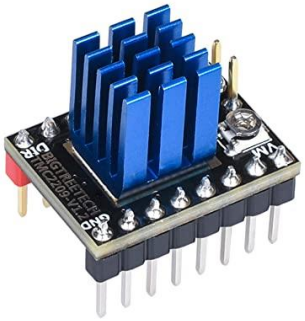
Στο παράδειγμα αυτό θα δούμε πως γίνεται στην πράξη ο έλεγχος ενός βηματικού κινητήρα με τη χρήση μικροελεγκτή και βηματικού οδηγού. Η παρουσίαση του παραδείγματος έχει σκοπό να εξοικειώσει τον αναγνώστη με τον έλεγχο των βηματικών κινητήρων μέσω της πλατφόρμας Arduino και να τον προετοιμάσει για το κεφάλαιο 4 στο οποίο παρουσιάζεται λεπτομερώς ο προγραμματισμός του μικροελεγκτή.

Για το παράδειγμα αυτό χρησιμοποιήθηκαν:

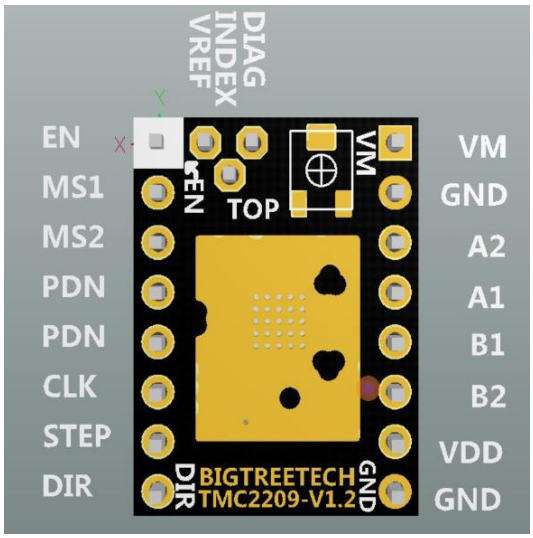
1) **Arduino Mega** – Παρόμοια πλακέτα με το Arduino Uno. Διαθέτει περισσότερα pins εισόδου και εξόδου.

2) **Βηματικός κινητήρας** - NEMA 17 , βήματος $1.8^0$, ονομαστικής ροπής 0.4Nm, 1.5A

3)**Βηματικός οδηγός TMC 2209** - Λειτουργίες Step/Dir/ microstepping



Εικόνα 2.19 Βηματικός οδηγός TMC2209




Εικόνα 2.20 Pin definition TMC 2209
Πηγή: Shenzhen Biqu Technologies Co. Ltd

Η σύνδεση των εξαρτημάτων έγινε ως εξής:

* όπου step. dir τα pins του βηματικού οδηγού
  όπου ARD τα pins του Arduino
  όπου mtr οι ακροδέκτες του κινητήρα

***Τροφοδοσία οδηγού***

VDD (step.dir) - 3.5 to 5 V(ARD)
GND (step.dir) - GND (ARD)

***Σύνδεση κινητήρα με οδηγό***

Οι τέσσερις ακροδέκτες του κινητήρα με:
A1(step.dir)
A2(step.dir)
B1(step.dir)
B2(step.dir)

***Τροφοδοσία κινητήρα****

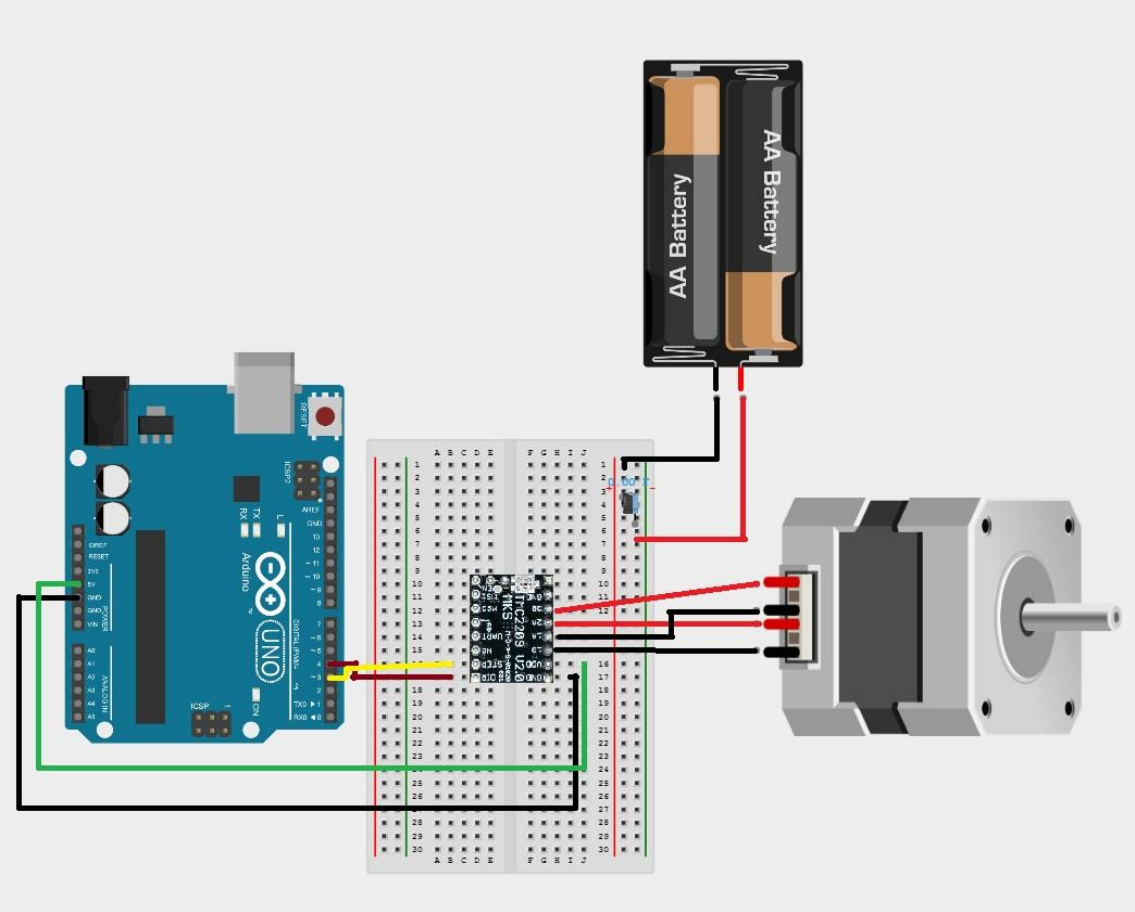
VM (step.dir) - εξωτερική τροφοδοσία 8-12V
GND(step.dir) - εξωτερική τροφοδοσία 8 -12V
*έγινε χρήση πυκνωτή χωρητικότητας 47 mF για προστασία από spikes ρεύματος

***Σύνδεση οδηγού με Arduino***

DIR(step.dir) - Pin4 (ARD) / έλεγχος φοράς κίνησης
STEP (step.dir) - Pin5(ARD) / έλεγχος βημάτων του κινητήρα

***Επιλογή Microstepping***

Ο συγκεκριμένος οδηγός έχει δύο λειτουργίες για microstepping την MS1 και την MS2. Για να ενεργοποιήσουμε κάποια από αυτές συνδέουμε το pin EN(step.dir) με το αντίστοιχο Pin της microstep λειτουργίας.




Εικόνα 2.21 Συνδεσμολογία εξαρτημάτων

***Προγραμματισμός του Arduino***

Ο παρακάτω κώδικας είναι γραμμένος στο IDE Arduino. Ο κινητήρας θα κινηθεί για 200 steps που ισοδυναμούν με μια πλήρη περιστροφή.

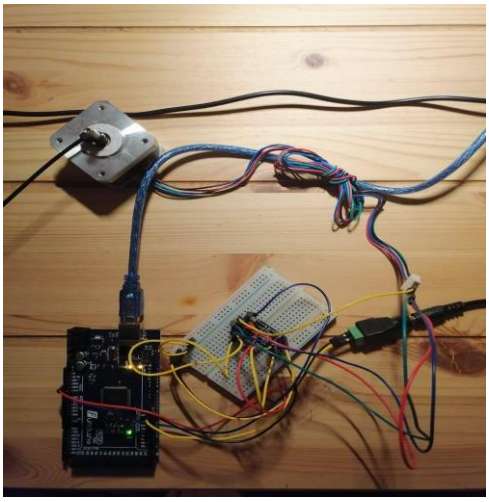
```
// Απλή περιστροφή 360 μοιρών
//
//ορισμός σταθερών//
const int stepPin = 3; //ορισμός Pin ελέγχου βημάτων
const int dirPin = 4; // ορισμός Pin ελέγχου κατεύθυνσης

void setup() {
  pinMode (3, OUTPUT); //Pin3 ώς έξοδος
  pinMode (4, OUTPUT); //Pin4 ως έξοδος
}

void loop() {
  digitalWrite (dirPin, HIGH); // επιλογή φοράς

  for (int x= 0; x<200; x++){   // έχουμε ανάλυση 1.8 μοίρες άρα πλήρης περιστροφή σε 200 steps
    digitalWrite(stepPin,HIGH); // στέλνουμε παλμό στον κινητήρα
    delayMicroseconds(500);     // από εδώ έλεγχος ταχύτητας
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);}
delay(1000);
    }                           //Τέλος προγράμματος
```

Εικόνα 2.22 Απλή περιστροφή κινητήρα



Εικόνα 2.23 Φυσικό σύστημα

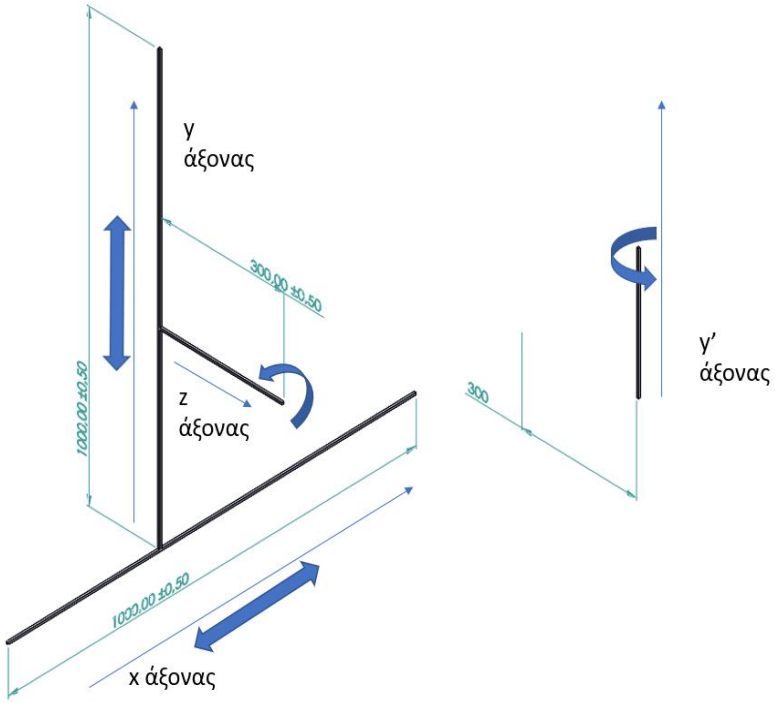
# 3 Σχεδιασμός Συστήματος

## 3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε αναλυτικά τον σχεδιασμό του συστήματος και τις επιλογές που έγιναν κατά τη διάρκεια του. Θα ξεκινήσουμε με τον ορισμό της κινηματικής του συστήματος. Έπειτα, θα συνεχίσουμε με τη δημιουργία του CAD μοντέλου τόσο για το σύστημα που φέρει τον σαρωτή όσο και για την περιστρεφόμενη τράπεζα. Στην συνέχεια θα δούμε μέσω προσομοίωσης την παραμόρφωση του εξαρτήματος που θα φέρει τον σαρωτή και αν αυτό επηρεάζει και σε ποιο βαθμό τη γεωμετρία της κατασκευής. Επίσης, θα δημιουργήσουμε Gcode για να παράξουμε τα απαραίτητα εξαρτήματα σε μηχανή CNC που διαθέτει το εργαστήριο.

Για την δημιουργία των CAD μοντέλων χρησιμοποιήθηκε το λογισμικό Solidworks, για την προσομοίωση της συμπεριφοράς του εξαρτήματος το Solidworks Simulation. Για τον σχεδιασμό των CAM parts και του G κώδικα χρησιμοποιήθηκε το λογισμικό SolidCAM ως add in του Solidworks. Τέλος, θα αναφερθούμε στα απαραίτητα χαρακτηριστικά που θα πρέπει να έχουν οι κινητήρες του συστήματος σε σχέση με την ακρίβεια τους αλλά και την απαραίτητη ροπή που χρειαζόμαστε.

## 3.2 Κινηματική του συστήματος




Εικόνα 3.1 Κινηματική του συστήματος

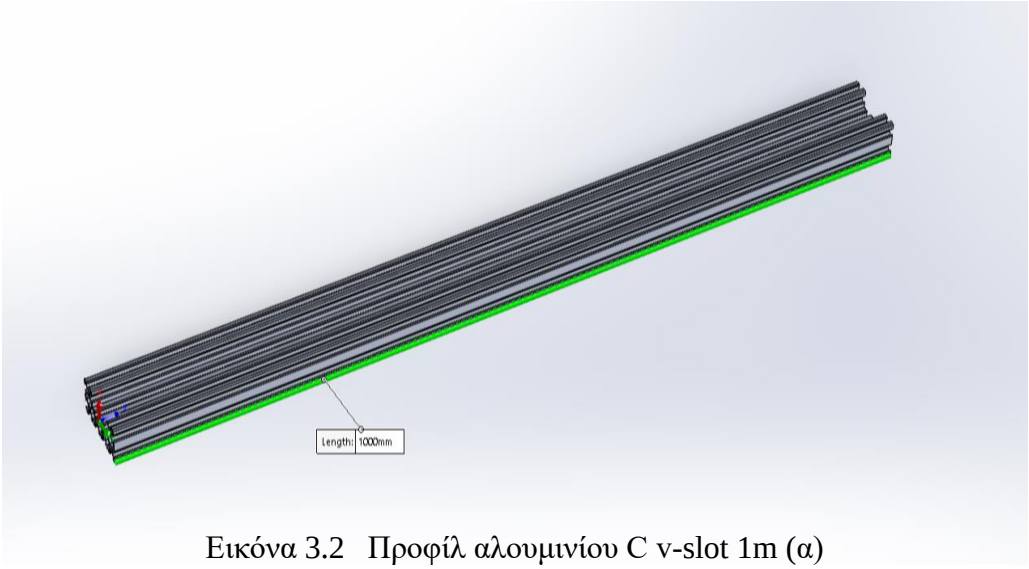
Στον οριζόντιο άξονα x θα επιτρέπεται μετατόπιση 1000mm. Στον κατακόρυφο αξονα y θα επιτρέπται επίσης μετατόπιση 1000mm. Κάθετα στην επιφάνεια x y έχει οριστεί ο άξονας z ο οποίος ορίζει την περιστροφή του σαρωτή. Τέλος, παράλληλα στον άξονα y και σε σταθερή απόσταση 300mm από τον άξονα x ως προς z έχουμε τον άξονα y'. Αυτός είναι ο άξονας περιστροφής του τραπεζιού.

## 3.3 Σχεδιασμός συστήματος μετακίνησης σαρωτή

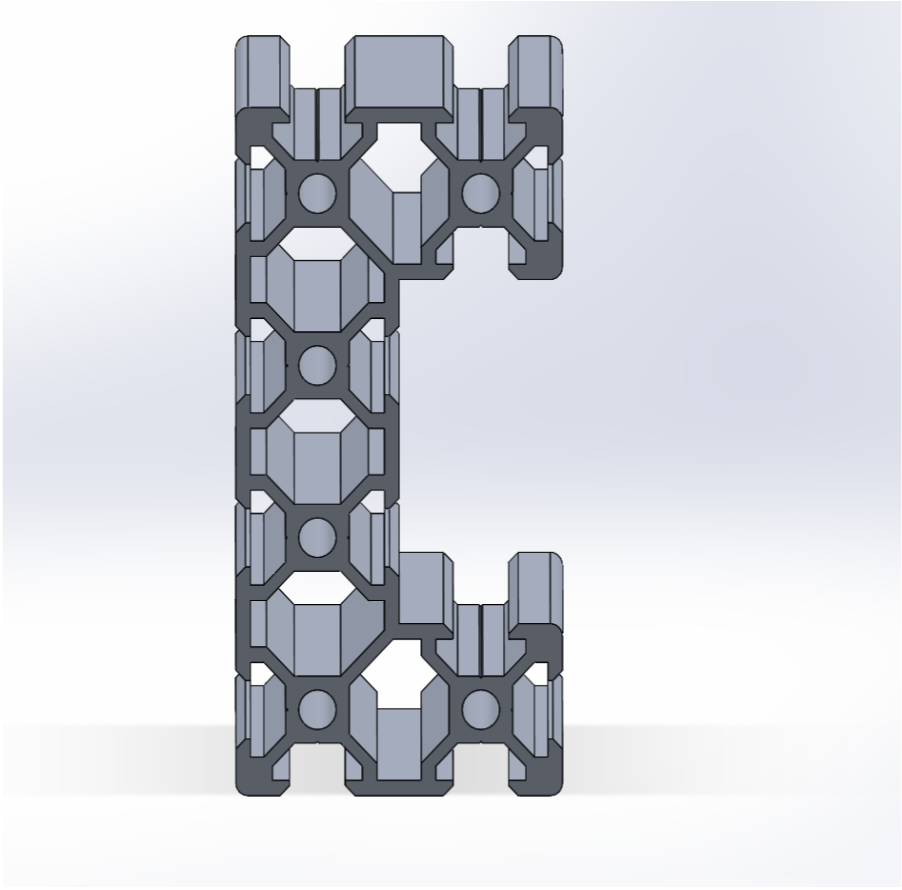
Πριν ξεκινήσει ο σχεδιασμός του συστήματος έγινε μια σχετική έρευνα αγοράς για να δούμε τα διαθέσιμα εξαρτήματα που υπάρχουν. Κατά τον σχεδιασμό προτιμήθηκε, όπου αυτό ήταν εφικτό, να χρησιμοποιηθούν εξαρτήματα που υπάρχουν στην αγόρα. Αυτό θα εξασφάλιζε την σωστή σύνδεση των κομματιών και θα μείωνε το κόστος του συστήματος. Επίσης θα έκανε ευκολότερη μια μελλοντική συντήρηση του συστήματος, καθώς τα διάφορα εξαρτήματα και ανταλλακτικά θα υπάρχουν στην αγορά και δεν θα είναι αναγκαία η παραγωγή τους.

### 3.3.1 Οριζόντια κίνηση

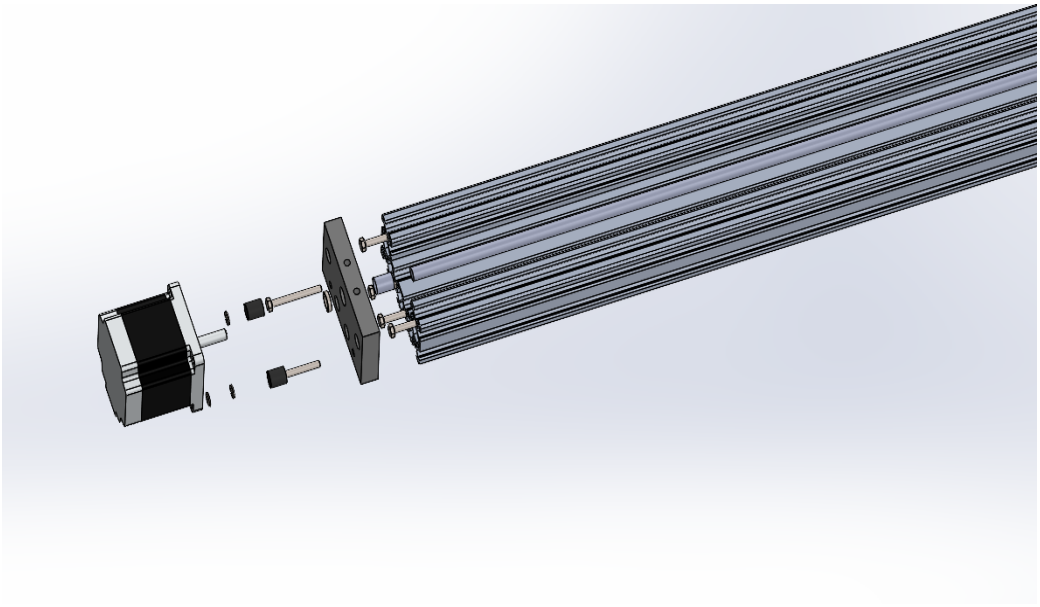
Για την οριζόντια κίνηση θεωρήθηκε αρκετή μια διαδρομή του ενός μέτρου. (*βλ. μεταπτυχιακή εργάσια Δ. Τύρη).* Ως μηχανισμός κίνησης επιλέχθηκε ο γραμμικός οδηγός με σφαιρική βίδα. Ο συγκεκριμένος οδηγός επιλέχθηκε λόγω της ακρίβειας κίνησης που παρέχει και των μειωμένων τριβών που αναπτύσσει κατά την κίνηση. Για τον γραμμικό επενεργητή επιλέχθηκε προφίλ αλουμινίου τύπου v-slot γεωμετρίας C.



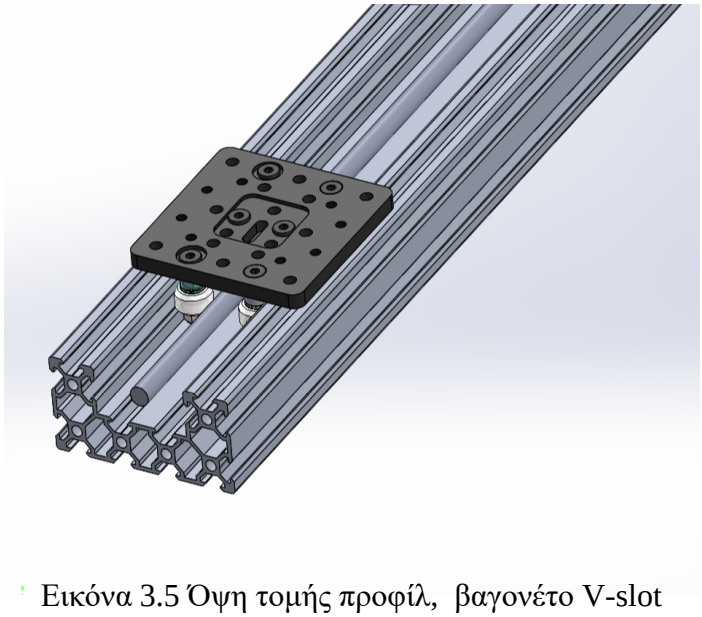

Εικόνα 3.2 Προφίλ αλουμινίου C v-slot 1m (α)



Εικόνα 3.3 Προφίλ αλουμινίου C v-slot (b)

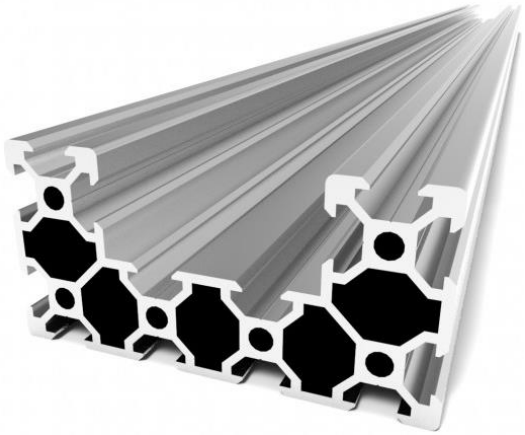


Εικόνα 3.4 Exploited view σύνδεσης κινητήρα με προφίλ αλουμινίου και κοχλία



Εικόνα 3.5 Όψη τομής προφίλ,  βαγονέτο V-slot

Εικόνα 3.6 Βαγονέτο όπως εμφανίζεται στο site του προμηθευτή



Εικόνα 3.7 Προφίλ αλουμινίου C όπως εμφανίζεται στο site του προμηθευτή

Εικόνα 3.8 C beam end mount όπως φαίνεται από το site του προμηθευτή

### 3.3.2 Κατακόρυφη κίνηση

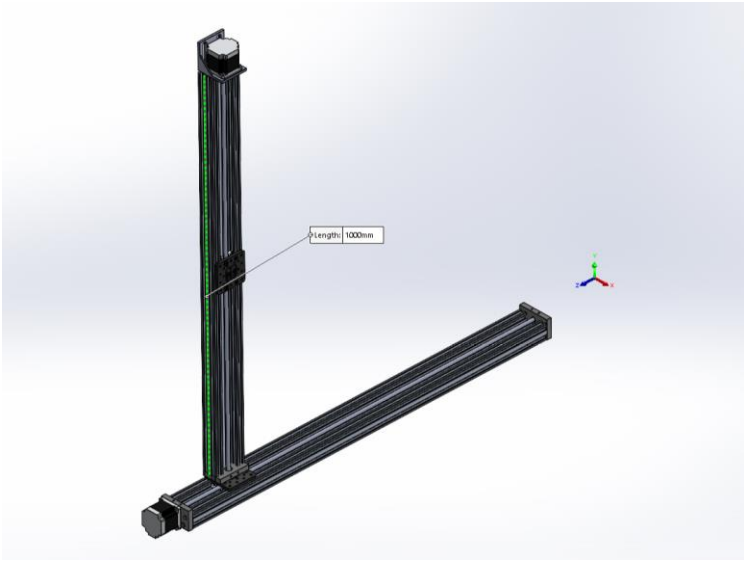
Για την κατακόρυφη κίνηση του σαρωτή επιλέχθηκε γραμμικός επενεργητής τύπου lead screw. Η κίνηση είναι κατακόρυφη οπότε για μηχανισμό σφαιρικής βίδας θα χρειαζόμασταν κάποιο φρένο σε περίπτωση αστοχίας του κινητήρα ή όταν αυτός δεν θα ήταν σε λειτουργία. Ο μηχανισμός κοχλία περικόχλιου δεν απαιτεί τη χρήση φρένου και αυτός ήταν ο λόγος που επιλέχθηκε.

Για την κατακόρυφη κίνηση ή ανύψωση του σαρωτή μελετήθηκαν και άλλες εναλλακτικές. Μια από αυτές ήταν η δημιουργία ενός τραπεζιού, το οποίο θα ανυψώνονταν με τη βοήθεια ενός συστήματος ψαλιδιών. Τεχνική παρόμοια με αυτήν που εφαρμόζεται σε ανυψώσεις προσωπικού, βιομηχανικού τύπου ανελκυστήρες και ανελκυστήρες ανύψωσης οχημάτων σε συνεργεία. Αυτή η μέθοδος όμως απορρίφθηκε εξαιτίας των αρκετών εξαρτημάτων που θα έπρεπε να κατασκευαστούν συγκεκριμένα για τον σκοπό της εργασίας. Επιπλέον, για λόγους απλότητας της κατασκευής προτιμήθηκε να χρησιμοποιηθεί παρόμοιος μηχανισμός με οριζόντια κίνηση.

Ως μήκος διαδρομής θεωρήθηκε αρκετό το ένα μέτρο όπως ακριβώς και για την οριζόντια κίνηση. Στην ουσία για τις δύο αυτές κινήσεις, οριζόντια και κατακόρυφη, έχουμε δύο παρόμοιους γραμμικούς οδηγούς. Ο ένας, που δίνει την οριζόντια κίνηση, παραμένει σταθερός ως προς ένα ακίνητο σημείο αναφοράς με μόνο κινητό μέρος, κατά τον οριζόντιο άξονα, το βαγονέτο. Ο δεύτερος επενεργητής είναι τοποθετημένος κάθετα πάνω στο βαγονέτο του πρώτου. Μ’ αυτόν τον τρόπο, το βαγονέτο του 2$^{ου}$ επενεργητή, αυτού που δίνει την κατακόρυφη κίνηση, έχει 2 βαθμούς ελευθερίας και μπορεί να βρίσκεται οπουδήποτε στο επίπεδο που ορίζουν οι δύο άξονες σε μια επιφάνεια ενός τετραγωνικού μέτρου.

### 3.3.3 Περιστροφή σαρωτή

Η κίνηση του σαρωτή ολοκληρώνεται με έναν ακόμα βαθμό ελευθερίας: αυτόν της περιστροφής γύρω από άξονα κάθετο στο επίπεδο που ορίζουν οι δυο προηγούμενοι οδηγοί. Η περιστροφή πραγματοποιείται από κινητήρα stepper motor με μειωτήρα. Η επιλογή κινητήρα με μειωτήρα, όπως θα αναφερθεί και στην συνέχεια, έγινε για δύο λόγους. Ο πρώτος ήταν για την αύξηση της ροπής και ο δεύτερος για την αύξηση της ακρίβειας περιστροφής.




Εικόνα 3.9 Οριζόντια και κατακόρυφη κίνηση

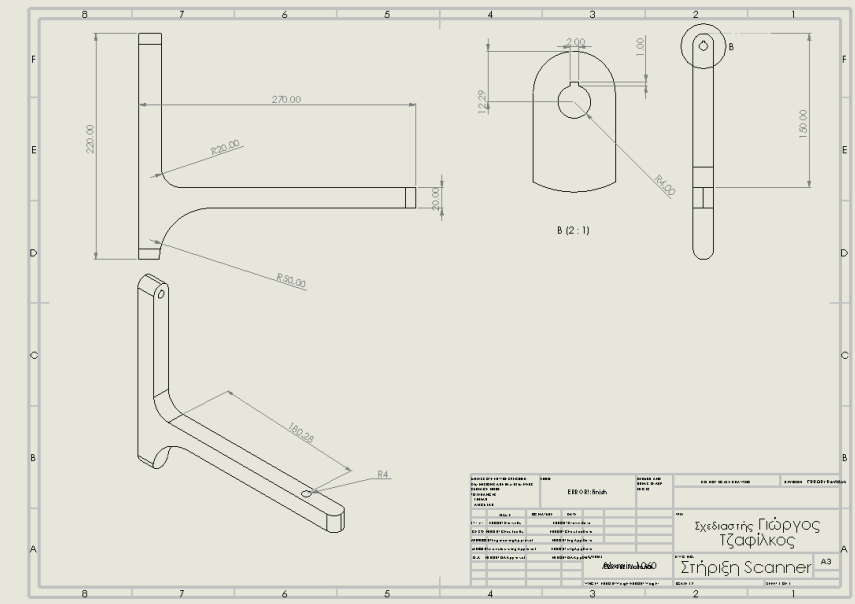
### 3.3.4 Στήριξη σαρωτή

Για τη στήριξη του σαρωτή έπρεπε να σχεδιαστεί και να κατασκευαστεί εξάρτημα κατάλληλης γεωμετρίας. Η στήριξη του σαρωτή θα πρέπει να τοποθετεί τον προβολέα και τις κάμερες του σαρωτή στο ίδιο επίπεδο και στην ίδια ευθεία με τον άξονα περιστροφής. Έτσι, σχεδιάστηκε το παρακάτω εξάρτημα.

Εικόνα 3.10 Εξάρτημα στήριξης σαρωτή

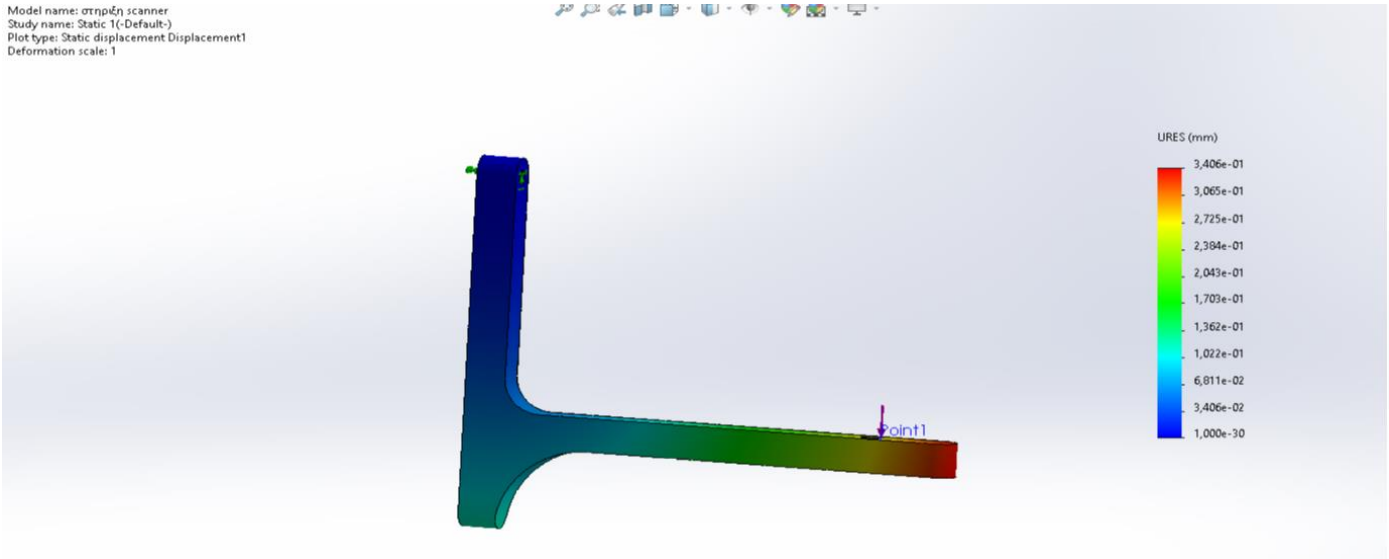
Ως υλικό του εξαρτήματος συγκράτησης επιλέχθηκε το αλουμίνιο για τις μηχανικές του ιδιότητες καθώς και για την ευκολία, με την οποία κατεργάζεται.

Αυτή η ευκολία στην κατεργασία μας ενδιαφέρει στην προκειμένη περίπτωση καθώς το εξάρτημα πρόκειται να κατασκευαστεί σε φρέζα CNC που διαθέτει το εργαστήριο.




Εικόνα 3.11 Κατασκευαστικό σχέδιο εξαρτήματος στήριξης scanner

Για το συγκεκριμένο εξάρτημα έγινε προσομοίωση της καταπόνησης με την μέθοδο των πεπερασμένων στοιχείων. Η προσομοίωση θα έδειχνε την παραμόρφωση του εξαρτήματος υπό την επίδραση του βάρους του σαρωτή και αν αυτή επηρεάζει και σε ποιο βαθμό τη γεωμετρία τους συστήματος. Για την προσομοίωση χρησιμοποιήθηκε το add in του Solidoworks, Solidworks Simulation. Ως υλικό επιλέξαμε το αλουμίνιο 1060, ως στήριξη επιλέχθηκε η πάκτωση στην ένωση του εξαρτήματος με τον κοχλία του κινητήρα και ως δύναμη ορίσαμε τα 35N, όσο και το βαρος του σαρωτή, σε απόσταση 200mm από τον άξονα περιστροφής.

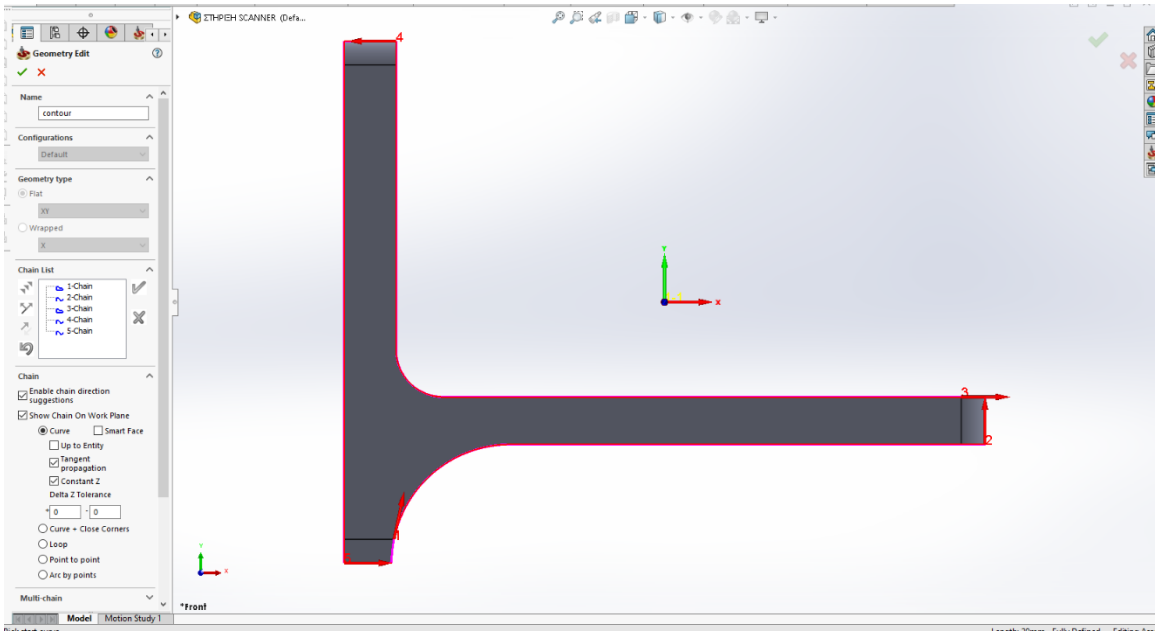



Εικόνα 3.12 Προσομοίωση στατικής φόρτισης εξαρτήματος συγκράτησης σαρωτή

Η προσομοίωση μας δείχνει ότι η μετατόπιση στη θέση στήριξης του άξονα θα είναι περίπου 0.27mm. Μετατρέποντας την μετατόπιση αυτή σε γωνία έχουμε

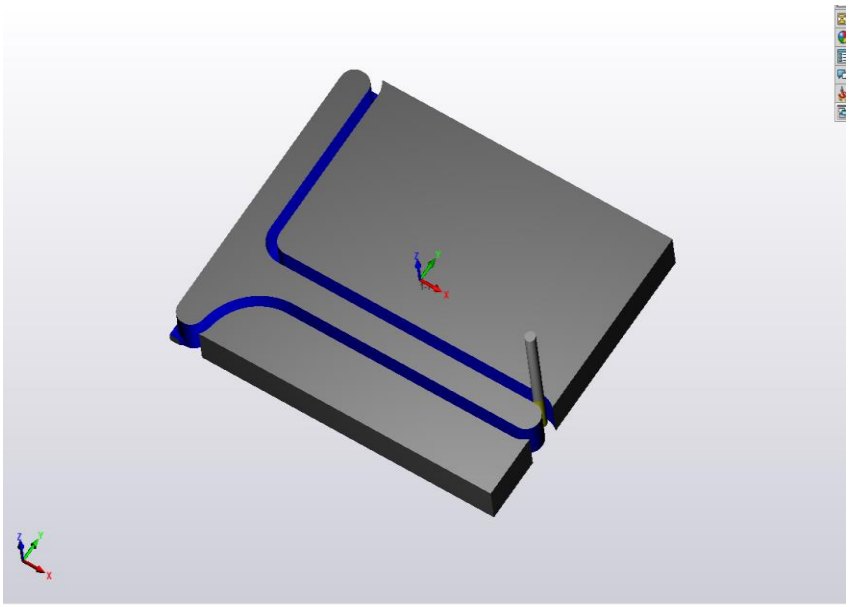
arctan (0.29/200) ≈ $0.08^0$ η οποία είναι πολύ μικρή για να επηρεάσει σε υπολογίσιμο βαθμό την κατασκευή μας. Εδώ θα πρέπει να σημειώσουμε ότι η πραγματική γωνία εξαιτίας λυγισμού θα είναι μεγαλύτερη καθώς θα έχουμε λυγισμό και από τον ίδιο τον κινητήρα αλλά και από το προφίλ αλουμινίου, ο οποίος λυγισμός θα εξαρτάται και από την θέση του βαγονέτου στον κατακόρυφο άξονα. Η μελέτη όμως του φαινομένου του λυγισμού για ολόκληρη την κατασκευή ξεφεύγει από το πλαίσιο αυτής της εργασίας.
Εφ ΄όσον είδαμε ότι το εξάρτημα στήριξης είναι ικανό να φέρει τον σαρωτή συνεχίζουμε με τη δημιουργία G κώδικα για την παραγωγή του εξαρτήματος σε φρέζα CNC. Για τον σχεδιασμό τον κατεργασιών και την παραγωγή G κώδικα χρησιμοποιήθηκε το λογισμικό SolidCAM ως add in του Solidworks.

Ως αρχικό στοκ επιλέχθηκε πλάκα αλουμινίου πάχους 2mm, πλάτους και ύψους 300mm. Εδώ να σημειώσουμε ότι το μέγεθος του στοκ στην πραγματική κατεργασία μπορεί να διαφέρει από αυτό που έχουμε ορίσει. Η τελική γεωμετρία του στοκ θα εξαρτηθεί από τα διαθέσιμα τεμάχια του προμηθευτή αλλά και τον τρόπο συγκράτησης του κομματιού πάνω στην μηχανή, π.χ. συγκράτηση σε κενό, σε μέγγενη κ.ο.κ. Αφού λοιπόν έχουμε ορίσει το στοκ και τις υπόλοιπες μεταβλητές (σύστημα συντεταγμένων, τελική γεωμετρία κομματιού) δημιουργούμε μια διαδικασία κοπής τύπου προφίλ. Επιλέγουμε ως γεωμετρία κοπής την περίμετρο του εξαρτήματος και ορίζουμε εργαλείο και συνθήκες κοπής.




Εικόνα 3.13 Επιλογή γεωμετρίας για την παραγωγή τροχιάς εργαλείου

Ως εργαλείο επιλέχθηκε ένα απλό κονδύλι flat nose διαμέτρου 8mm με συνθήκες κοπής S3500 F1000 και βάθος κοπής τα 0.4mm. Τέλος, επιλέχθηκε G κώδικας με τη χρήση post processor κατάλληλο για τον τύπο μηχανής που θα χρησιμοποιήσουμε. Συγκεκριμένα, χρησιμοποιήθηκε ο g_milling_3x που βρίσκεται στην βιβλιοθήκη της solidCAM.



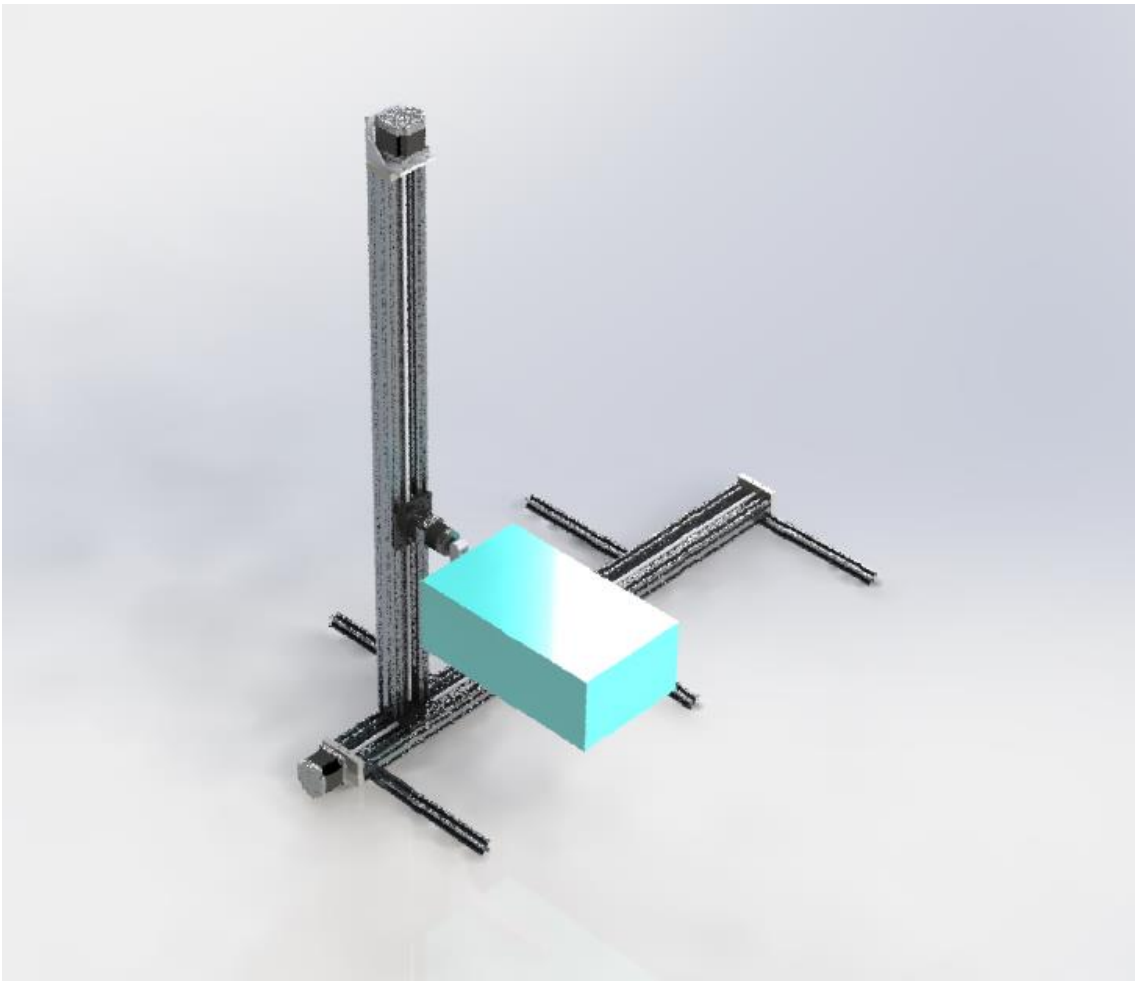
Εικόνα 3.14 Προεπισκόπηση κοπής από SolidCAM

```
O1000 (Stiriksi SCANNER)
(COMPENSATION-WEAR)
(REV-0.70)
(DEC-11-2021-8:10:06PM)
(TOOL 1 - DIA 8.)
N1 G90 G17 G40 G80 G00
M06 T1 ()
(F-contour2)
G00 G54 G90 X-124.161 Y-119.3 S3500 M03
G43 H1 Z120.
S3500
Z25.
Z2.
G01 Z-5. F300.
X-128.961 F1000.
X-128.926 Y-118.721
X-128.822 Y-118.151
X-128.649 Y-117.598
X-128.411 Y-117.069
X-128.111 Y-116.573
X-127.754 Y-116.117
X-127.344 Y-115.707
X-126.888 Y-115.35
```

Εικόνα 3.15 Μέρος Gcode της κατεργασίας**.**

Ο σχεδιασμός ολοκληρώνεται με την προσθήκη πέντε προφίλ αλουμινίου μήκους 250mm εκατέρωθεν του x άξονα που θα λειτουργήσουν ως πλαϊνή στήριξη της κατασκευής.

Ο σαρωτής iscan M300 έχει διαστάσεις 300x200x130 και αναπαρίσταται ως το μπλε κουτί στην παρακάτω φωτορεαλιστική απεικόνιση.



Εικόνα 3.16 Φωτορεαλιστική απεικόνιση με σαρωτή

Εικόνα 3.17 Φωτορεαλιστική απεικόνιση

## 3.4 Σχεδιασμός περιστρεφόμενου τραπεζιού

Το περιστρεφόμενο τραπέζι είναι το δεύτερο υποσύστημα της κατασκευής. Σε αυτό θα στερεώνεται το αντικείμενο προς σάρωση και θα περιστρέφεται ελεγχόμενα. Για τον σχεδιασμό του περιστρεφόμενου τραπεζιού λήφθηκαν υπόψιν τα παρακάτω χαρακτηριστικά:

- Να μπορεί να περιστρέφεται τουλάχιστον 360$^{o}$ δεξιόστροφα και αριστερόστροφα.
- Να μπορεί να δεχτεί βάρος έως 100 kg.
- Η κατασκευή να γίνει με χαμηλό κόστος

Το περιστρεφόμενο τραπέζι είναι μια σχετικά απλή διάταξη. Κύριο εξάρτημα μιας τέτοιας διάταξης είναι το κεντρικό ρουλεμάν, το οποίο είναι σχεδιασμένο για να δέχεται μεγάλες αξονικές δυνάμεις. Συνήθως το περιστρεφόμενο μέρος τοποθετείται στο πάνω μέρος ενός τέτοιου ρουλεμάν και το κάτω μέρος του ρουλεμάν συνδέεται σταθερά με την υπόλοιπη κατασκευή η οποία είναι πακτωμένη. Για το συγκεκριμένο τραπέζι επιλέχθηκε το ρουλεμάν με βάσεις που φαίνεται στην φωτογραφία *εικ 3.18*.
Το συγκεκριμένο ρουλεμάν έχει διαστάσεις 150x150mm και σύμφωνα με τον κατασκευαστή αντέχει σε αξονικά φορτία έως 220kg.

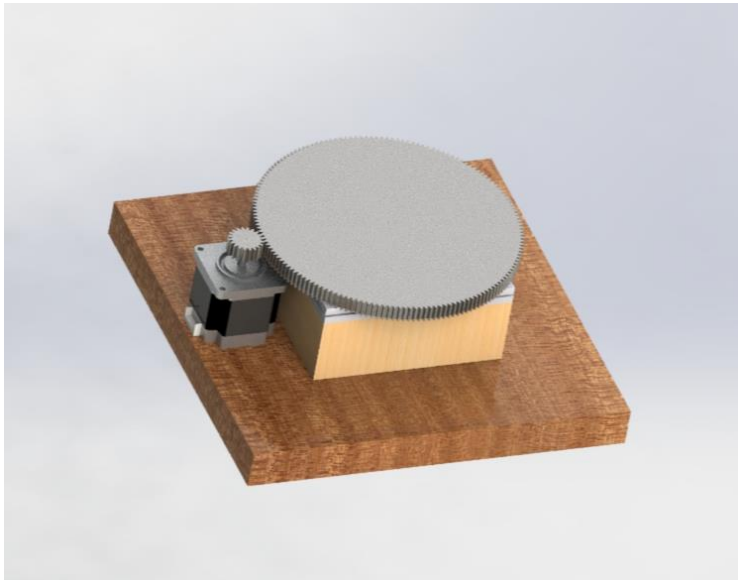
Εικόνα 3.18 Ρουλεμάν για περιστρεφόμενο τραπέζι (swivel plate)
Πηγη: www.richelieu.com

Όπως είπαμε το περιστρεφόμενο τραπέζι είναι μια απλή διάταξη έτσι για το συγκεκριμένο τραπέζι η λογική της σχεδίασης είναι η εξής: Το ρουλεμάν θα τοποθετηθεί σε μια επιφάνεια, με το πάνω μέρος του ρουλεμάν να έχει σταθερή σύνδεση με γρανάζι συνολικής διαμέτρου περίπου ίσης με τις ακμές του ρουλεμάν(150mm). Το γρανάζι θα κινείται από κινητήρα που

θα φέρει γρανάζι ίδιου module, αλλά μικρότερης διατομής, για την δημιουργία μεγαλύτερης ροπής αλλά και υψηλότερης ακρίβειας μετατόπισης. Τέλος, τόσο ο κινητήρας όσο και η σταθερή επιφάνεια που θα φέρει το ρουλεμάν θα είναι πακτωμένα σε μια κοινή επιφάνεια. Επίσης, πάνω στο γρανάζι θα στερεώνεται η επιφάνεια στην οποία θα τοποθετούνται τα αντικείμενα προς σάρωση.

Για την κοινή επιφάνεια, στην οποία θα πακτωθούν ο κινητήρας και η επιφάνεια που θα εδράζεται το ρουλεμάν, επιλέχθηκε το ξύλο σαν υλικό. Η διαστάσεις της τάβλας αυτής θα είναι 300x300x20 mm. Ξύλινος θα είναι και ο κύβος πάνω στον οποίο θα εδράζεται το ρουλεμάν. Η επιλογή του ξύλου έγινε με γνώμονα την ευκολία επεξεργασίας του αλλά και την εύκολη προσάρτηση διαφόρων εξαρτημάτων σε αυτό με κόλλα ή καρφιά. Επίσης το χαμηλό κόστος του ξύλου σαν υλικό, συγκριτικά με κάποιο μέταλλο όπως το αλουμίνιο, θα κρατήσει χαμηλά το κόστος της κατασκευής.

Η σχέση περιστροφής του κινητήρα με αυτή του τραπεζιού είναι 6:1. Αυτό θα αυξήσει την ακρίβεια του συστήματος και θα μειώσει τις ανάγκες ροπής του κινητήρα.



Εικόνα 3.19 Φωτορεαλιστική απεικόνιση περιστρεφόμενου τραπεζιού

## 3.5 Διαστασιολόγηση Κινητήρων

Σημαντικό στάδιο στον σχεδιασμό του συστήματος είναι η επιλογή των κατάλληλων κινητήρων. Οι κατάλληλοι κινητήρες θα πρέπει να τηρούν τις προδιαγραφές σχετικά με την ακρίβεια θέσης που έχουμε ορίσει και να παρέχουν την απαιτούμενη ροπή στο σύστημα.

***Ακρίβεια θέσης – Γραμμική Κίνηση***

Ως απαιτούμενη ακρίβεια στις γραμμικές κινήσεις ορίσαμε το 1mm. Πάμε να εξετάσουμε το ελάχιστο βήμα κινητήρα που χρειαζόμαστε για να έχουμε αυτήν την ακρίβεια. Για τους γραμμικούς επενεργητές επιλέχθηκε κοχλίας με τα εξής χαρακτηριστικά:

- Μήκος 1000mm
- Lead 8mm
- Lead pitch 2mm
- Number of starts 4

$$ελάχιστο\ βήμα\ κινητήρα = (απαιτούμενη\ ακρίβεια\ θέσης)/lead * 360 \quad (3.1)$$

Έτσι, σύμφωνα με τα χαρακτηριστικά που έχουμε χρειαζόμαστε κινητήρα με ελάχιστο βήμα $45^0$. Οι περισσότεροι κινητήρες του εμπορίου έχουν βήμα αρκετά μικρότερο από $45^0$.

***Ακρίβεια γωνίας – Περιστροφική κίνηση***

Ως απαιτούμενη ακρίβεια στις περιστροφικές κινήσεις ορίσαμε την $1^0$ .

Για την περιστροφή του σαρωτή καθώς η περιστροφή γίνεται απευθείας από τον κινητήρα χρειαζόμαστε βήμα μικρότερο ή ίσο με $1^0$ .

Για την περιστροφή του τραπεζιού έχουμε σχέση μετάδοσης 6:1 με το απαιτούμενο βήμα κινητήρα να προκύπτει από την ακόλουθη σχέση:

$$ελάχιστο\ βήμα\ κινητήρα = απαιτούμενη\ ακρίβεια\ γωνίας * σχέση\ μετάδοσης \quad (3.2)$$

Έτσι προκύπτει ότι για την περιστροφή του τραπεζίου χρειαζόμαστε κινητήρα με βήμα μικρότερο ή ίσο με $6^0$ .

***Υπολογισμός απαιτούμενης ροπής κινητήρων***

Για τον υπολογισμό της ελάχιστης ροπής που χρειάζεται να έχει ο κινητήρας θα υπολογίσουμε την ελάχιστη ροπή που χρειάζεται κάθε κίνηση του συστήματος.

***Απαιτούμενη ροπή ευθύγραμμης κίνησης στον Χ άξονα.***

Για την ευθύγραμμη κίνηση στον Χ άξονα χρησιμοποιήθηκε κοχλίας με τα χαρακτηριστικά που αναφέρθηκαν προηγουμένως και ένα φορείο σφαιρικής βίδας για να κινεί το φορείο.

Σύμφωνα με την βιβλιογραφία, η απαραίτητη ροπή που χρειάζεται για να κινηθεί ένα φορείο πάνω σε κοχλία ορίζεται από την ακόλουθη σχέση:

$$T=FL*2ΠE \quad (3.3)$$

Όπου

- $T$ η απαιτούμενη ροπή.
- $F$ οι αξονικές δυνάμεις
- $E$ συντελεστής αποτελεσματικότητας

Ως αξονικές δυνάμεις λαμβάνονται οι δυνάμεις, οι οποίες αντιστέκονται στην ευθύγραμμη κίνηση.
Για την οριζόντια κίνηση θεωρήθηκε μια αξονική δύναμη η οποία αντιστέκεται στην κίνηση ίση με περίπου 50N και έχει να κάνει κυρίως με την επαφή του φορείου με το προφίλ αλουμινίου.
Ο συντελεστής αποτελεσματικότητας $E$ από τους κατασκευαστές για τις σφαιρικές βίδες ορίζεται περίπου στο 0.9.
Έτσι, η απαιτούμενη ροπή κινητήρα για την μετατόπιση στον άξονα Χ από την εξίσωση 3.3 υπολογίζεται σε: 0,7 kg.cm

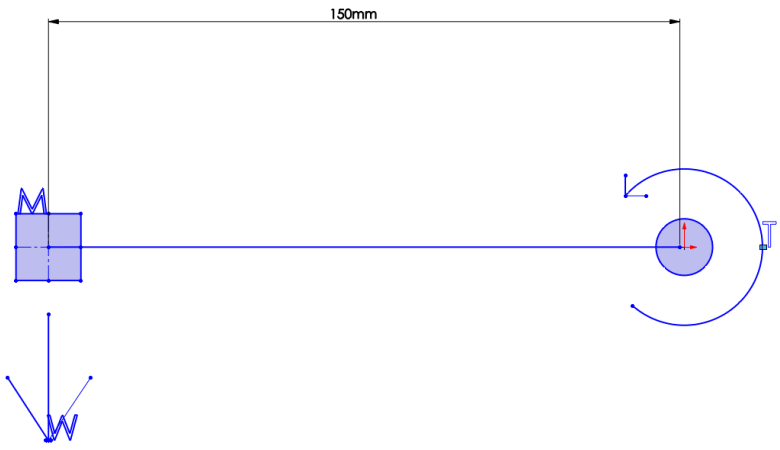
***Απαιτούμενη ροπή ευθύγραμμης κίνησης στον Y άξονα***
Για τον υπολογισμό της απαιτούμενης ροπής θα χρησιμοποιήσουμε τον ίδιο τύπο (3.3)
Ως αξονική δύναμη θεωρούμε το βάρος W του υποσυστήματος (σαρωτής – κινητήρας περιστροφής - φορείο) καθώς και μια δύναμη Ft η οποία προκαλείται από την τριβή του φορείου με το προφίλ αλουμινίου. Η δύναμη αυτή (Ft) ορίζεται περίπου στα 30N ενώ το βάρος του υποσυστήματος (W) στα 50N.
Για την κατακόρυφη κίνηση χρησιμοποιήσαμε απλό περικόχλιο για το φορείο όπως ενδείκνυται, καθώς δεν χρειάζεται η χρήση συστήματος φρένων σε περίπτωση αστοχίας. Οι κατασκευαστές δίνουν έναν βαθμό αποτελεσματικότητας για τα περικόχλια που κυμαίνεται από 0,2 έως 0,8 και εξαρτάται από την γεωμετρία του κοχλία, τα υλικά καθώς και την λίπανση. Για τους κοχλίες που πρόκειται να χρησιμοποιήσουμε δεν υπήρχαν διαθέσιμα στοιχεία για τον συντελεστή αποτελεσματικότητας, όποτε επιλέχθηκε ένας μικρός συντελεστής τιμής 0,3. Έτσι η γενική εξίσωση 4.3 για τον υπολογισμό της ροπής γίνεται:

$$Ty = ((W + Ft) * L)/2\pi E \quad (3.4)$$

Άρα, η απαιτούμενη ροπή για την κίνηση στον κατακόρυφο άξονα είναι 3,4 kg*cm.

***Υπολογισμός απαιτούμενης ροπής για την περιστροφή του σαρωτή***
Ο σαρωτής είναι τοποθετημένος σε εξάρτημα με τη θέση στήριξης του σαρωτή να απέχει 150mm από τον άξονα περιστροφής. Το βάρος του σαρωτή είναι περίπου 35N. Μέγιστη

ροπή συγκράτησης θα χρειάζεται όταν η απόσταση της δύναμης του βάρους θα είναι μέγιστη. Αυτό γίνεται όταν θα έχουμε περιστροφή $90^0$ από την αρχική θέση ισορροπίας.




Εικόνα 3.20 Υπολογισμός ροπής στην περιστροφή του σαρωτή

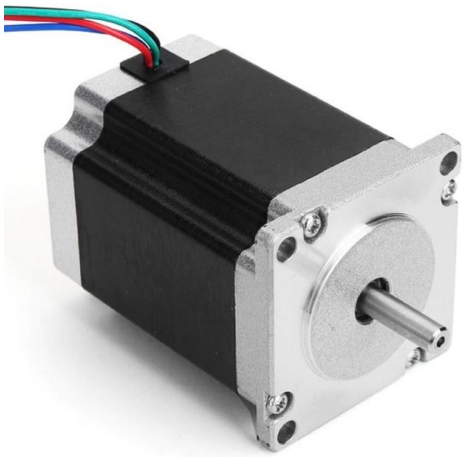
Η μέγιστη ροπή αναπτύσσεται στην οριζόντια θέση με τιμή 3,5*15 =52,5kg*cm.
Έτσι, από τις τιμές που υπολογίσαμε προκύπτει ο παρακάτω συγκεντρωτικός πίνακας:

| ***Χαρακτηριστικά*** | ***Οριακές τιμές κινητήρα*** |
|---|---|
| Ακρίβεια στην ευθύγραμμη κίνηση 1mm | Μέγιστο βήμα $45^0$ |
| Ακρίβεια στην περιστροφική κίνηση σαρωτή | Μέγιστο βήμα $1^0$ |
| Ακρίβεια στην περιστροφική κίνηση του τραπεζιού | Μέγιστο βήμα $6^0$ |
| Ροπή ευθύγραμμης κίνησης στο οριζόντια άξονα | Ελάχιστη τιμή ροπής 0.7kg*cm |
| Ροπή ευθύγραμμης κίνησης στο κατακόρυφο άξονα | Ελάχιστη τιμή ροπής 3,4 kg*cm |
| Ροπή περιστροφικής κίνησης σαρωτή | Ελάχιστη τιμή ροπής 52.5kg*cm |

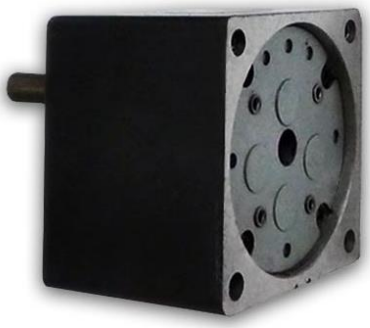
*Πίνακας 3.1 Οριακές τιμές χαρακτηριστικών κινητήρων*

Με βάση τις τιμές του πίνακα 3.1 και μετά από ερεύνα αγοράς επιλέχθηκαν οι παρακάτω κινητήρες:

- Για την ευθύγραμμη κίνηση στο οριζόντιο άξονα επιλέχθηκε ο κινητήρας NEMA23 9kg*cm $1,8^0$
- Για την ευθύγραμμη κίνηση στον κατακόρυφο άξονα επιλέχθηκε ο κινητήρας NEMA23 19kg*cm $1,8^0$
- Για την περιστροφική κίνηση του σαρωτή κινητήρας επιλέχθηκε ο κινητήρας NEMA23 9kg*cm $1,8^0$ με μειωτήρα 1:10 τελικής ροπής 90kg*cm $0.18^0$



Εικόνα 3.21 Κινητήρας Nema 23 19kg*cm όπως εμφανίζεται στο site του προμηθευτή



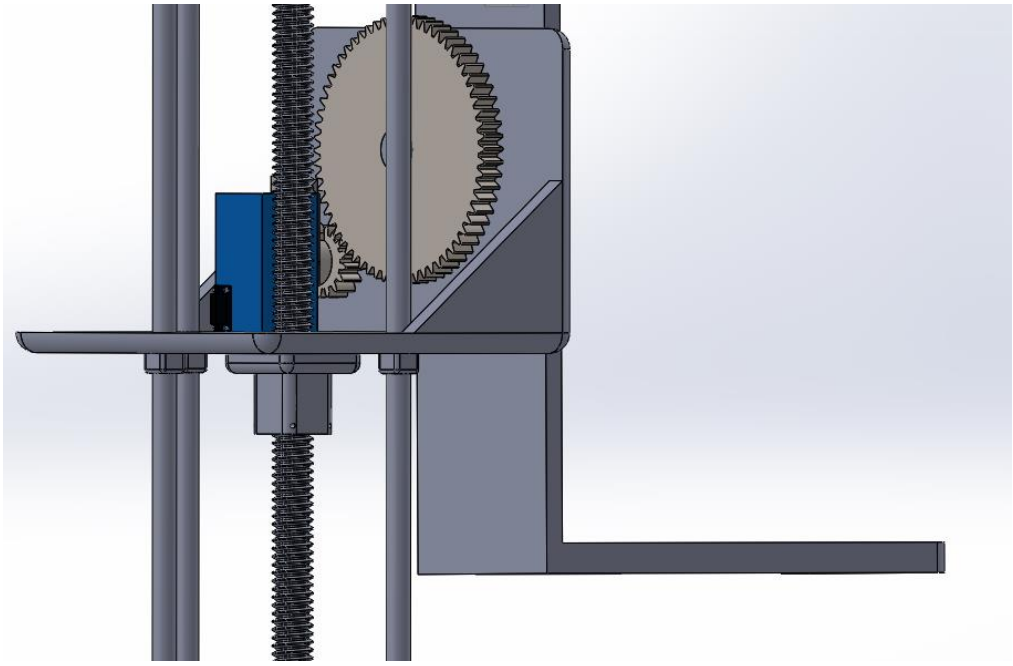
Εικόνα 3.22 Μειωτήρας στροφών 1:10 Nema 23 όπως εμφανίζεται στο site του προμηθευτή

## 3.6 Άλλες σχεδιαστικές προσεγγίσεις

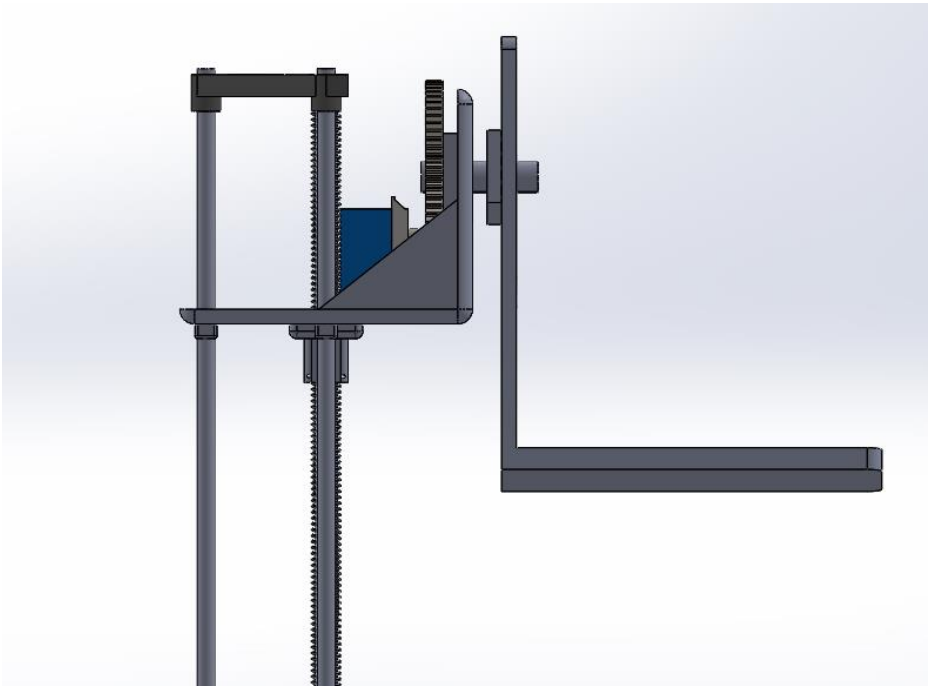
Κατά την διάρκεια του σχεδιασμού μελετήθηκαν και άλλοι μηχανισμοί κίνησης κυρίως για την περιστροφή του σαρωτή και για την κατακόρυφη μετακίνηση. Οι παρακάτω προσεγγίσεις απορρίφθηκαν κυρίως για την απαίτηση που είχαν σε ειδικά εξαρτήματα όπως ειδικά φορεία, βασεις στήριξης, ψαλίδια κ.α. Τα εξαρτήματα αυτά θα έπρεπε να

κατασκευαστούν αυξάνοντας το συνολικό κόστος και τις κατασκευαστικές απαιτήσεις του μηχανισμού.

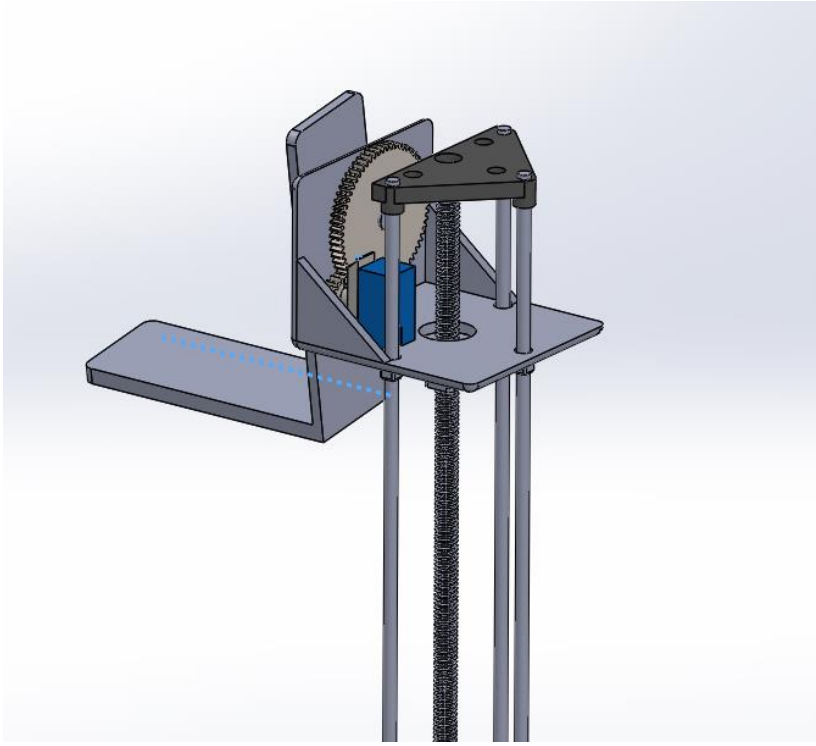
### 3.6.1 Περιστροφή με σερβοκινητήρα



Εικόνα 3.23 Περιστροφή με σερβοκινητήρα (α)



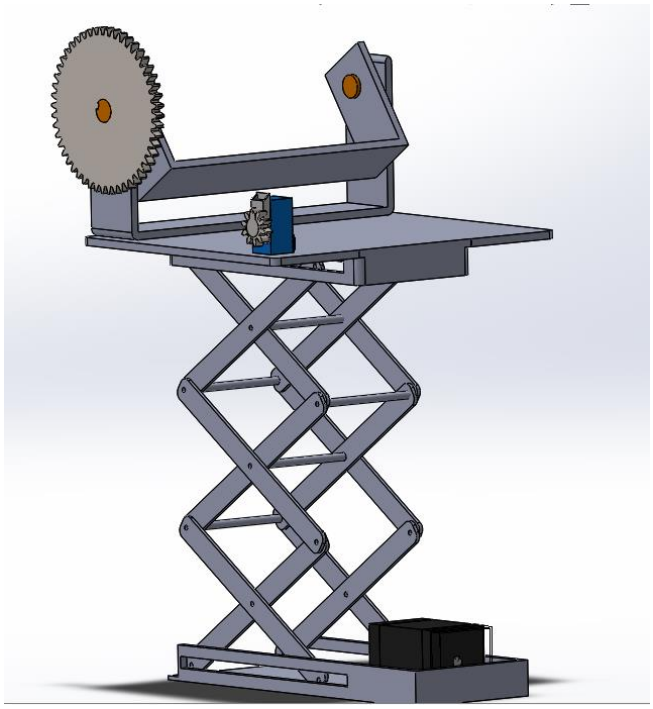
Εικόνα 3.24 Περιστροφή με σερβοκινητήρα (β)



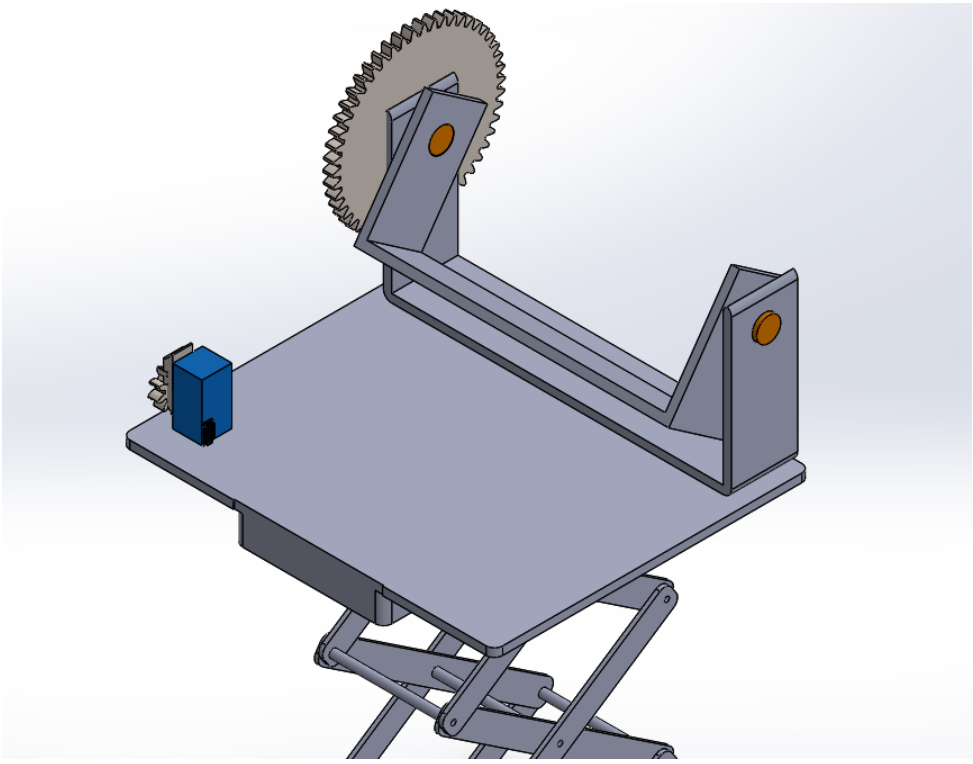
Εικόνα 3.25 Περιστροφή με σερβοκινητήρα (γ)

Με αυτόν τον μηχανισμό η περιστροφή θα γινόταν με χρήση σερβοκινητήρα και γραναζιού. Ο κινητήρας θα έπρεπε να τοποθετηθεί σε φορείο το οποίο θα κινούνταν κατακόρυφα.

### 3.6.2 Ανύψωση με σύστημα ψαλιδιών



Εικόνα 3.26 Ανύψωση με ψαλίδια



Εικόνα 3.27 Περιστροφή σαρωτή στον μηχανισμό με ψαλίδια

Στον παραπάνω μηχανισμό η κατακόρυφη κίνηση θα πραγματοποιούνταν με την χρήση ψαλιδιών. Ένας γραμμικός οδηγός θα μετακινούσε το κάτω μέρος των ψαλιδιών αναγκάζοντας τα να ΄ανοίγουν΄ και να ΄κλείνουν΄. Με αυτό τον τρόπο θα είχαμε κατακόρυφη κίνηση. Για την περιστροφή του σαρωτή ένας σερβοκινητήρας μέσω ιμάντα θα έδινε κίνηση σε γρανάζι το οποίο με την σειρά του θα περιέστρεφε τον σαρωτή με τον μηχανισμό που φαίνεται στις εικόνες (3.26, 3.27).

## 3.7 Κατασκευή συστήματος

Για την κατασκευή του υποσυστήματος κίνησης του σαρωτή η διαδικασία είναι αρκετά απλή. Όλα τα εξαρτήματα, εκτός της στήριξης του σαρωτή, υπάρχουν στο εμπόριο και έχουν επιλεχθεί με τις κατάλληλες διαστάσεις για την εύκολη συνεργασία τους. Η κατασκευή του υποσυστήματος αυτού απαιτεί κυρίως την συναρμολόγηση των κομματιών. Για τη διαχείριση των καλωδίων του υποσυστήματος θα πρέπει να δώσουμε ιδιαίτερη προσοχή σε αυτά των κινητήρων του κατακόρυφου άξονα και της περιστροφής. Σε αντίθεση με τον κινητήρα του οριζόντιου άξονα που είναι σταθερός οι υπόλοιποι κινητήρες του σαρωτή θα πρέπει να κινούνται ελεύθερα οριζόντια και κατακόρυφα. Έτσι, θα πρέπει να γίνει μια αντίστοιχη διαχείριση καλωδίων που θα επιτρέπει τις κινήσεις αυτές.

Για την κατασκευή του περιστρεφόμενου τραπεζιού οι κατασκευαστικές ανάγκες είναι μεγαλύτερες καθώς εδώ έχουμε εξαρτήματα που είναι ειδικά σχεδιασμένα για τον σκοπό της εργασίας. Κατά την κατασκευή θα πρέπει να δοθεί ιδιαίτερη προσοχή ώστε η βάση πάνω

στην οποία θα τοποθετηθεί το ρουλεμάν να είναι παράλληλη με τον οριζόντιο άξονα. Η βάση αυτή, όπως προαναφέρθηκε, θα κατασκευαστεί με ξύλο με τις συνδέσεις των επιμέρους κομματιών να γίνονται με καρφιά και κόλλα. Με αντίστοιχο τρόπο θα γίνει και η σύνδεση με την ξύλινη βάση του υποσυστήματος. Επόμενο βήμα στην κατασκευή του τραπεζιού είναι η τοποθέτηση του ρουλεμάν και του γραναζιού πάνω σε αυτό. Κρίσιμο για τη σωστή λειτουργία του υποσυστήματος είναι η έκκεντρη τοποθέτηση του γραναζιού με τον άξονα περιστροφής του ρουλεμάν. Σε περίπτωση μη σωστής τοποθέτησης, το σύστημα δεν θα μπορεί να λειτουργήσει όπως σχεδιάστηκε οπότε χρειάζεται ιδιαίτερη προσοχή.

Τέλος, αφού τα δύο υποσυστήματα έχουν κατασκευαστεί θα πρέπει να γίνει η σχετική τους τοποθέτηση για την ολοκλήρωση του συστήματος. Εδώ ο προτζέκτορας του σαρωτή θα πρέπει να είναι προσανατολισμένος με τρόπο που να κοιτάει τον άξονα περιστροφής του τραπεζιού. Η απόσταση μεταξύ των δύο υποσυστημάτων θα πρέπει να είναι σταθερή και όσο γίνεται μικρότερη.

# 4 Προγραμματισμός μικροελεγκτή – Σχεδιασμός UI

## 4.1 Εισαγωγή

Για τον έλεγχο των κινητήρων του συστήματος δημιουργήθηκε κώδικας, c++ στο IDE Arduino. Ο κώδικας αυτός περιέχει μεταβλητές, οι οποίες καθορίζουν την κίνηση των τεσσάρων κινητήρων και κατ' επέκταση τη σχετική θέση του σαρωτή ως προς το αντικείμενο. Δημιουργήθηκε επίσης UI, το οποίο λειτουργεί σε περιβάλλον Windows και μέσα από αυτό ο χρήστης μπορεί να ορίζει τις μεταβλητές θέσης των κινητήρων. Για τη δημιουργία του UI χρησιμοποιήθηκε το Visual Studio 2019 και για τον προγραμματισμό του η γλώσσα C#.

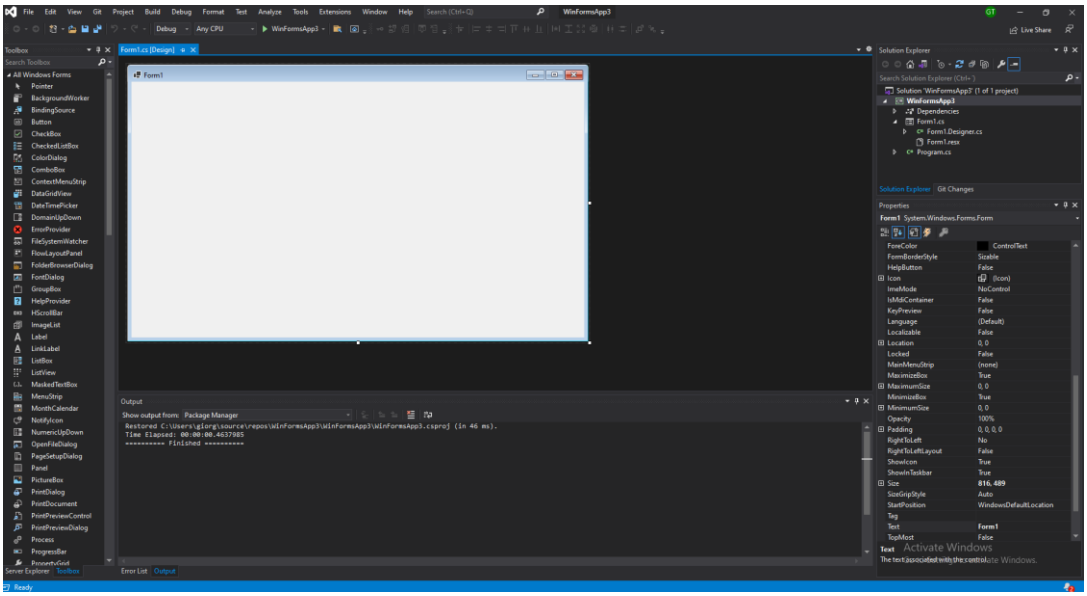
## 4.2 Δημιουργία UI

Το Visual Studio είναι μια ολοκληρωμένη πλατφόρμα δημιουργίας κώδικα και εφαρμογών που προσφέρεται από την microsoft. Το Visual Studio υποστηρίζει τις περισσότερες γλώσσες προγραμματισμού και είναι μια δημοφιλής πλατφόρμα για την δημιουργία εφαρμογών σε όλα τα λειτουργικά συστήματα, όπως για παράδειγμα: windows, mac os, linux και android. Χρησιμοποιείται επίσης για τη δημιουργία ιστοσελίδων και web εφαρμογών.

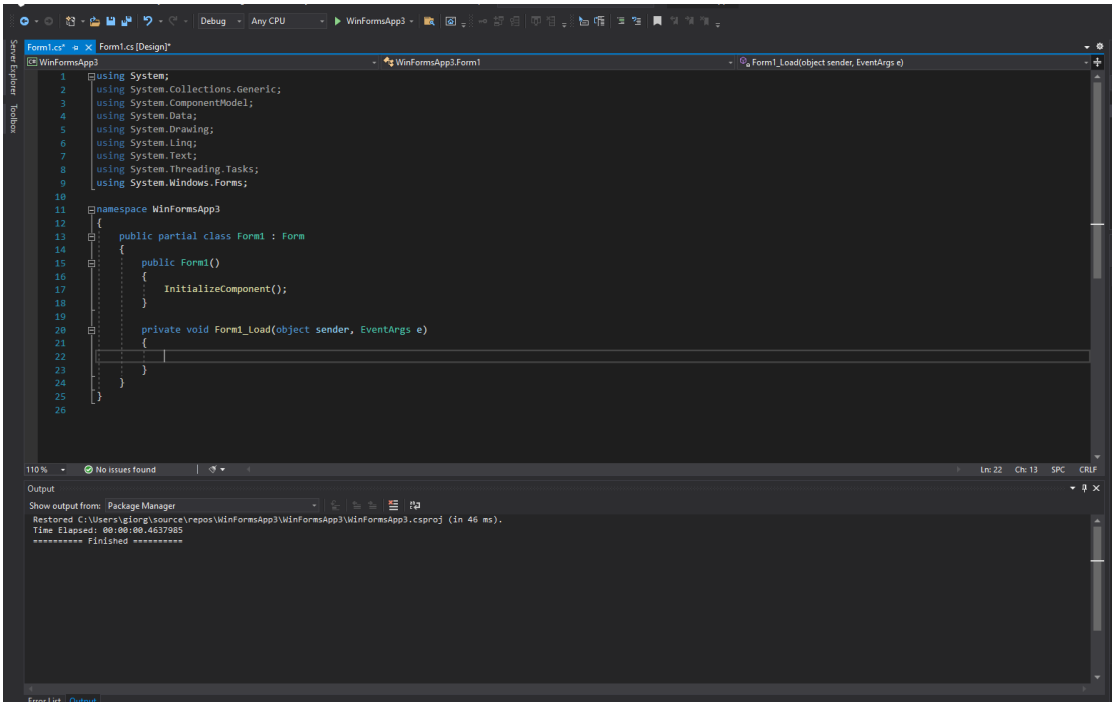
Χρησιμοποιήσαμε την έκδοση 2019 community, η οποία διατίθεται δωρεάν από την επίσημη ιστοσελίδα του Microsoft Visual Studio.

Για τη δημιουργία του UI από την αρχική καρτέλα του Visual Studio, επιλέξαμε δημιουργία νέου project, τύπου windows form app με χρήση της C# ως γλώσσα προγραμματισμού. Το περιβάλλον προγραμματισμού του Visual Studio για την συγκεκριμένη περίπτωση έχει 2 κύρια παράθυρα προγραμματισμού. Το ένα, το οποίο είναι γραφικό, χρησιμοποιείται για την διαμόρφωση του γραφικού περιβάλλοντος. Περιέχει λειτουργίες drag n drop για την τοποθέτηση στοιχείων στο παράθυρο όπως κουμπιά, text boxes μπάρες κ.α., και είναι μια οπτική απεικόνιση του τελικού προγράμματος. Οι λειτουργίες των στοιχείων μπορούν να προγραμματιστούν μέχρι ένα βαθμό από την γραφική καρτέλα του προγράμματος. Μερικά από τα χαρακτηριστικά που μπορούμε να ορίσουμε από αυτό το περιβάλλον είναι τα ονόματα των στοιχείων ως μεταβλητές, τις διαστάσεις τους , τον τρόπο που θα εμφανίζουν τα δεδομένα, να τα ομαδοποιήσουμε κ.α.

Από την άλλη το προγραμματιστικό, μη γραφικό περιβάλλον, μας δίνει τον πλήρη έλεγχο πάνω στον προγραμματισμό τόσο της ίδιας της εφαρμογής όσο και των επιμέρους στοιχείων της.



Εικόνα 4.1 Γραφικό περιβάλλον προγραμματισμού VS

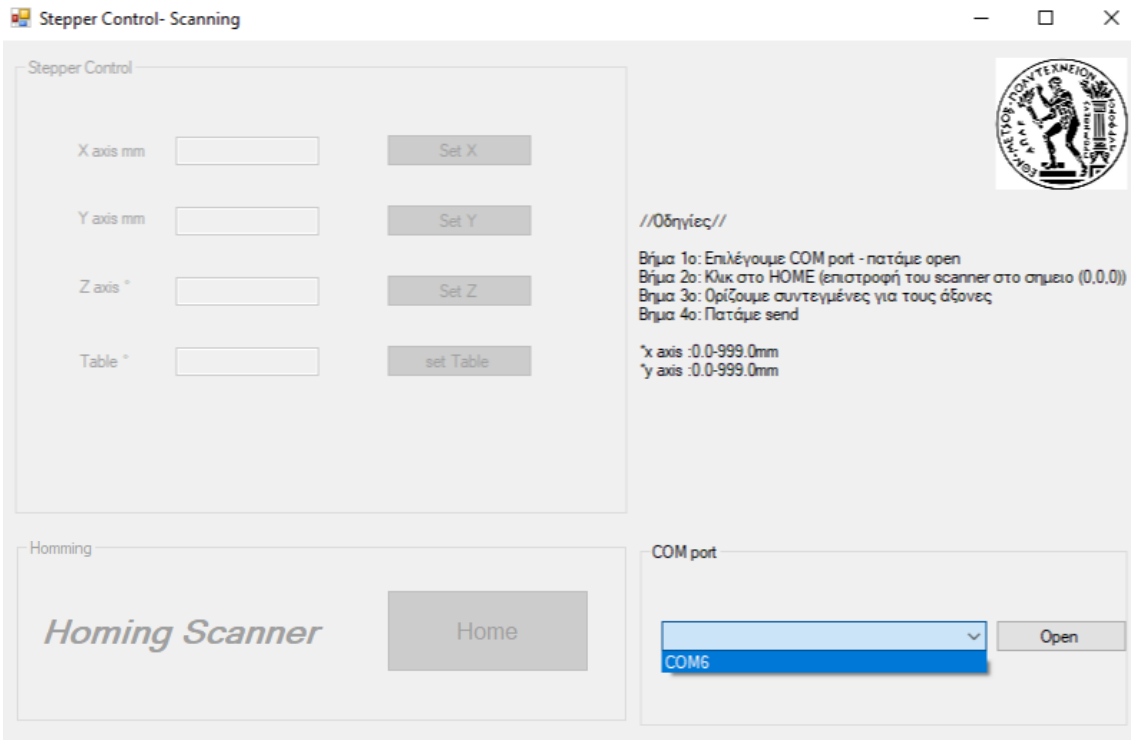



Εικόνα 4.2 Μη γραφικό περιβάλλον προγραμματισμού VS

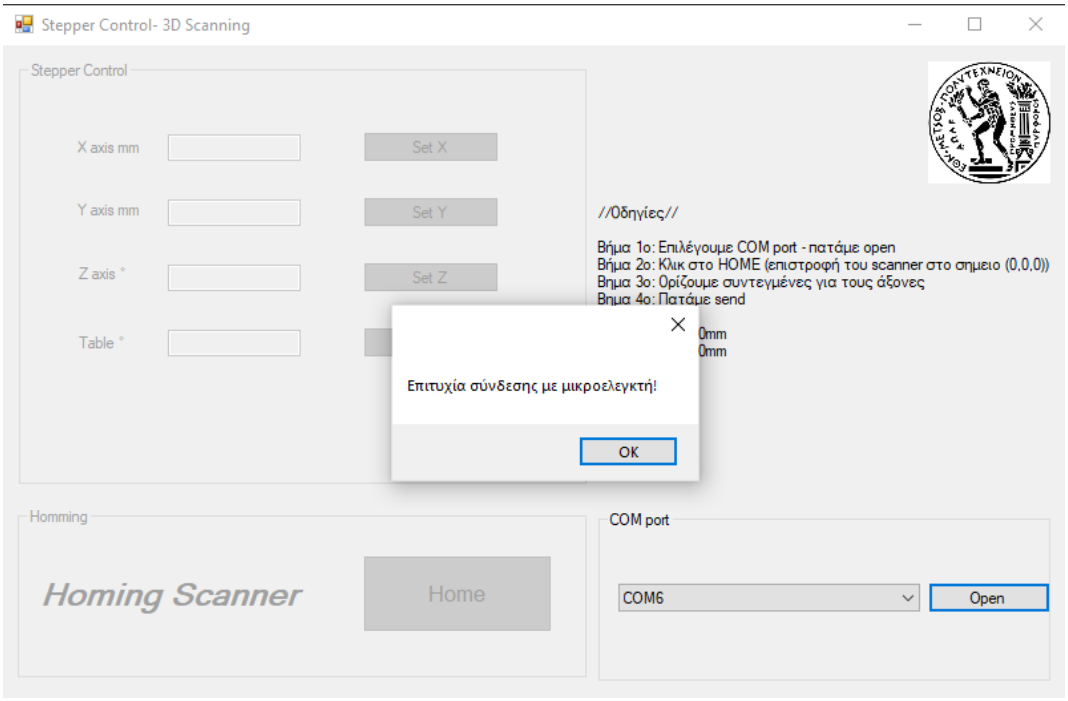
Σκοπός του UI είναι η δημιουργία ενός απλού και εύκολου εργαλείου μέσω του οποίου ο χρήστης θα μπορεί να επικοινωνεί με το μηχανικό σύστημα. Το UI σχεδιάστηκε για να παρέχει στον χρήστη τον έλεγχο καθενός από τους τέσσερις κινητήρες ξεχωριστά αλλά και να επιλέγει το αν θα ξεκινήσει η διαδικασία ‘Homing’. Homing είναι η διαδικασία που επιστρέφει τους κινητήρες στην αρχή των αξόνων ορίζοντας αυτό σαν σημείο 0. Έτσι με το

Homing ο χρήστης μπορεί να επιλέξει τις θέσεις των κινητήρων με βάση ένα σταθερό σύστημα συντεταγμένων. Χωρίς την διαδικασία του Homing το σύστημα δέχεται σαν μηδενική θέση την θέση που είχε όταν τέθηκε σε λειτουργία. Εδώ να σημειώσουμε ότι η διαδικασία του Homing σχεδιάστηκε για του κινητήρες που ορίζουν την κίνηση του scanner και όχι για αυτόν του τραπεζίου.

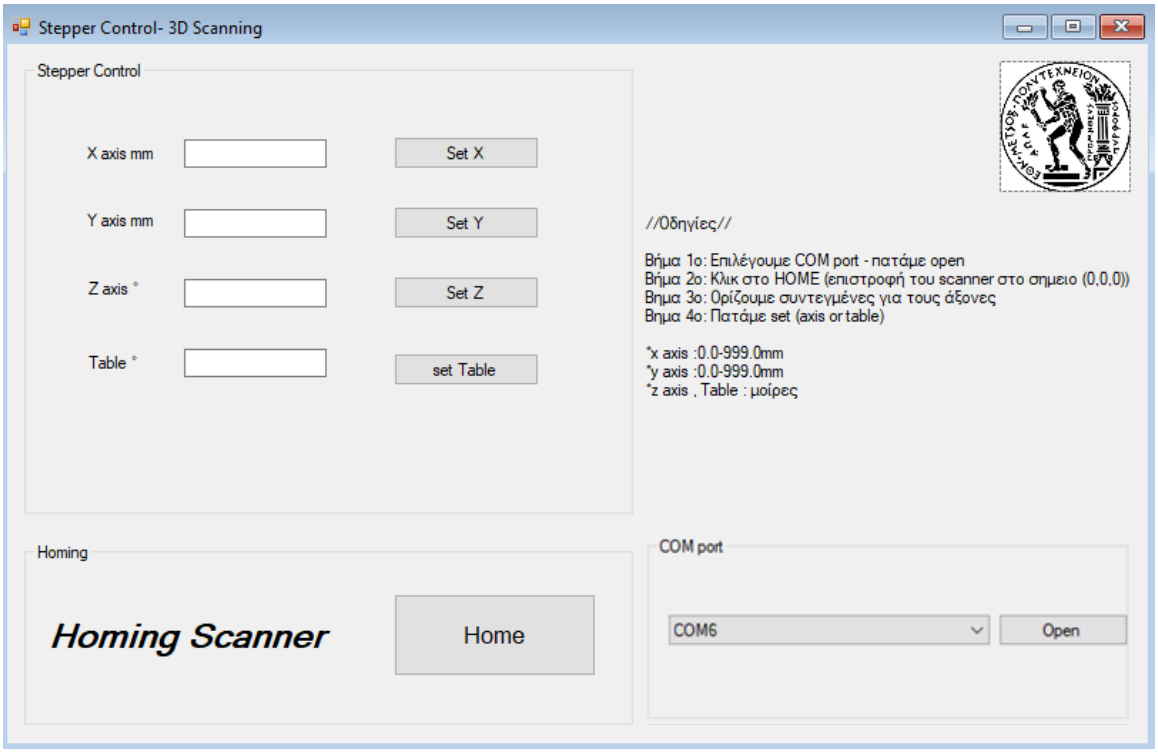
Ανοίγοντας το πρόγραμμα ο χρήστης έχει σαν μόνη επιλογή τον ορισμό του COM port για τη σύνδεση με τον μικροελεγκτή. Το drop down μας δείχνει τις διαθέσιμες προς επιλογή COM ports. Αφού επιλέξουμε την κατάλληλη πύλη, η οποία πρέπει να είναι ίδια με αυτή που χρησιμοποιεί ο μικροελεγκτής και πατήσουμε open, μας εμφανίζεται παράθυρο με το μήνυμα "Επιτυχία σύνδεσης με μικροελεγκτή". Έπειτα από αυτό το βήμα είναι διαθέσιμες και οι επιλογές για την κίνηση των κινητήρων.




Εικόνα 4.3 Παράθυρο εφαρμογής - Διαθέσιμα COM ports




Εικόνα 4.4 Εμφάνιση παραθύρου επιτυχίας σύνδεσης




Εικόνα 4.5 Παράθυρο με διαθέσιμες επιλογές κίνησης των κινητήρων

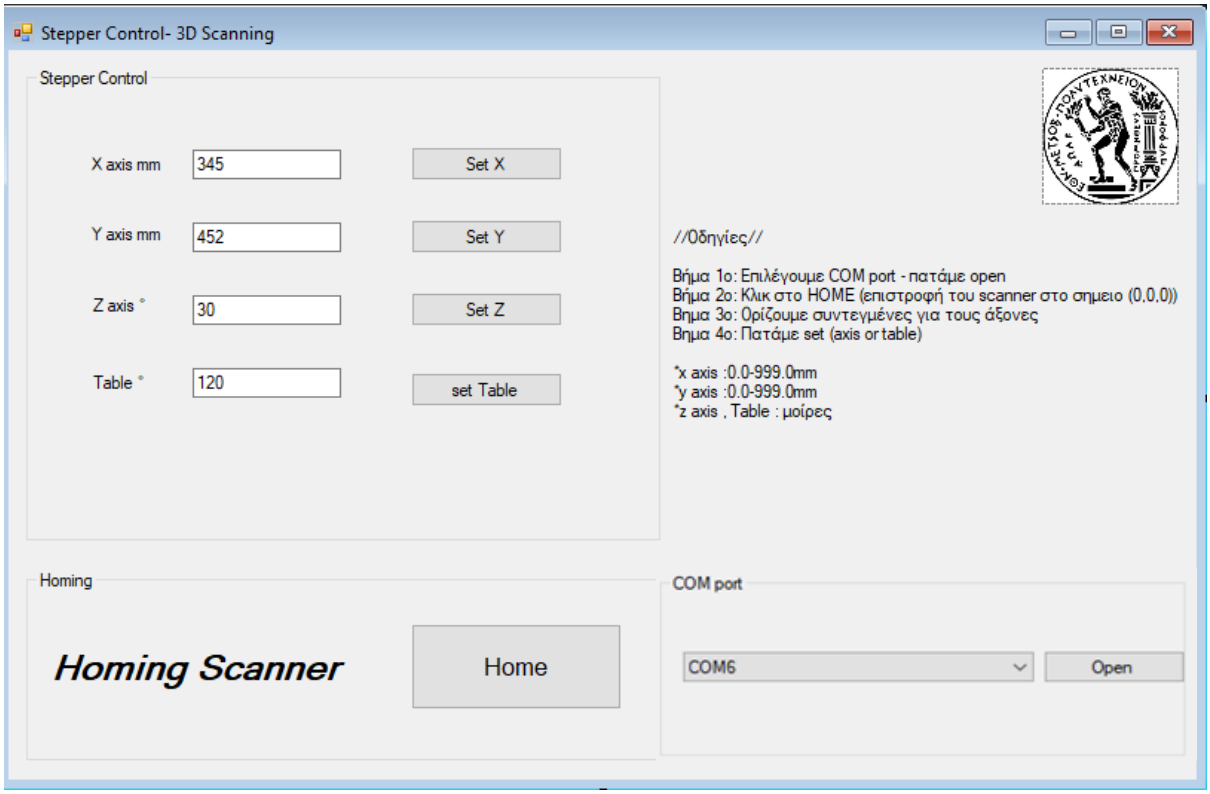
Όπως μπορούμε να διακρίνουμε και από τις εικόνες το παράθυρο του προγράμματος είναι χωρισμένο στα εξής τέσσερα κύρια μέρη:

- Stepper Control
  Το μέρος πάνω αριστερά όπου ο χρήστης ορίζει τις θέσεις των τεσσάρων κινητήρων.

- Homing
  Το μέρος κάτω αριστερά όπου είναι διαθέσιμη η επιλογή HOME που ενεργοποιεί την διαδικασία homing.

- COM port
  Το μέρος κάτω δεξιά που μας επιτρέπει την επιλογή της κατάλληλης θήρας COM.

- Φόντο παραθύρου
  Το φόντο στην μέση και δεξιά του παραθύρου περιέχει οδηγίες για την ορθή χρήση του προγράμματος. Επίσης, πάνω δεξιά βρίσκεται το λογότυπο του ΕΜΠ.




Εικόνα 4.6 Παράθυρο με τις θέσεις των αξόνων

Πριν οριστούν οι θέσεις στους άξονες προτείνεται στον χρήστη να ξεκινήσει την αυτόματη διαδικασία homing. Αφού ολοκληρωθεί η διαδικασία (με οπτική επιβεβαίωση από το φυσικό σύστημα) ο χρήστης μπορεί να ορίσει τις θέσεις που θέλει να μεταφερθεί ο σαρωτής αλλά και την περιστροφή του τραπεζίου. Για να ορίσει τις θέσεις αυτές ο χρήστης, πολύ απλά, εισάγει την τιμή που επιθυμεί στο αντίστοιχο πλαίσιο και πατάει set (axis). Με το πάτημα του κουμπιού ξεκινάει η κίνηση του αντίστοιχου κινητήρα η οποία πραγματοποιείται με σταθερή ταχύτητα για το μεγαλύτερο μέρος της διαδρομής. Εξαίρεση η αρχή και το τέλος

της κίνησης που έχουμε σταθερή επιτάχυνση και επιβράδυνση αντίστοιχα. Οι τιμές των ταχυτήτων είναι σταθερές και έχουν οριστεί από τον κώδικα ελέγχου των κινητήρων. Μετά τον ορισμό της θέσης η τιμή παραμένει στο κελί και μας ενημερώνει για τις συντεταγμένες του συστήματος. Εδώ να σημειώσουμε πως για τις ευθύγραμμες κινήσεις ο ορισμός γίνεται σε mm και για τις περιστροφικές σε μοίρες. Οι οδηγίες προτρέπουν τον χρήστη να εισάγει τιμές για τις ευθύγραμμες κινήσεις από 0 έως 999 όσο δηλαδή η μέγιστη διαδρομή στους άξονες αυτούς ενώ για τις περιστροφικές δεν υπάρχει κάποιος περιορισμός.

Ο σχεδιασμός του UI έγινε παράλληλα με τη δημιουργία του κώδικα ελέγχου των κινητήρων. Προτεραιότητα όμως δόθηκε στον κώδικα των κινητήρων με το UI να μπορεί να υποστηρίζει όλες τις απαραίτητες λειτουργίες.

## 4.3 Προγραμματισμός μικροελεγκτή

Για τον έλεγχο των κινητήρων δημιουργήθηκε κώδικας C++ στο IDE Arduino. Χρησιμοποιήθηκε η βιβλιοθήκη “AccelStepper” η οποία παρέχεται ελεύθερα από την κοινότητα Arduino. Η βιβλιοθήκη αυτή μας επιτρέπει να ορίσουμε την ταχύτητα, την επιτάχυνση, τη θέση αναφοράς, τη μετατόπιση και τη μετατόπιση σε θέση ενός βηματικού κινητήρα.

Ο κώδικας περιέχει τρείς συναρτήσεις πέραν τον βασικών loop() και setup() και θα αναφερθούμε σε καθεμία από αυτές.

void setup ()
Είναι βασική συνάρτηση του Arduino και τρέχει μόλις ανοίξουμε το πρόγραμμα. Σε αυτήν έχουμε ορίσει την εντολή Serial.begin(9600) και την λειτουργία INPUT_PULLP για τα pin, στα οποία θα είναι οι διακόπτες που θα μας βοηθήσουν στην διαδικασία του homing, όπως θα δούμε αργότερα.

void stepper position ()
Σε αυτήν τη συνάρτηση έχουμε ορίσει την κίνηση των κινητήρων με τη βοήθεια της AccelStepper βιβλιοθήκης. Εδώ έχουμε ορίσει για τον κάθε κινητήρα την ταχύτητα και την επιτάχυνση, με την οποία θα κινείται. Επίσης έχουν οριστεί και οι εντολές κίνησης για τον κάθε κινητήρα, οι οποίες θα ενεργοποιούνται με την κατάλληλη είσοδο. Η εντολή κίνηση είναι η stepper.moveTo () το οποίο σημαίνει ότι η τιμή που θα πάρει η συνάρτηση θα μετακινήσει τον κινητήρα σε αυτήν τη θέση. Δηλαδή αν βάλουμε τιμή 10 θα μετακινήσει τον κινητήρα στη θέση 10 και όχι για 10.
Η μονάδα εισόδου είναι τα steps του κινητήρα. Έτσι για να έχουμε αντιστοιχία σε mm πολλαπλασιάζουμε τα steps με κατάλληλη τιμή. Για τις ευθύγραμμες μετατοπίσεις λαμβάνοντας υπόψιν τους κινητήρες και τους κοχλίες που έχουμε επιλέξει, η τιμή αυτή είναι

25. Για την περιστροφή του τραπεζιού 3 και για την περιστροφή του σαρωτή λαμβάνοντας υπόψιν τον μειωτήρα στροφών είναι 27.8.

void Homing ()

Η συνάρτηση αυτή θα τοποθετήσει τους κινητήρες του σαρωτή σε ένα σημείο Α και θα το ορίσει σαν νέα αρχή των αξόνων. Το σημείο Α και άρα το μηδενικό σημείο (0,0,0) στο νέο σύστημα συντεταγμένων ορίζεται από τη θέση που έχουν τοποθετηθεί οι διακόπτες πάνω στους άξονες.

```
//////////////Homing StepperX////////////////////////////////////////////////////////////
while (digitalRead(homeswitchX))     // stepper moves backwards until hits the switch
{ stepperX.setMaxSpeed(250);         //X axis speed for homing
  stepperX.setAcceleration(100);     //X axis accelaration for homing

  stepperX.moveTo(homingX); // decrease one step at a time
  homingX--;
  stepperX.run();          // runs stepper
  delay(5);
}

stepperX.setCurrentPosition(0);     //  Set the current possition as 0
stepperX.setMaxSpeed(250);         //X axis speed for homing
stepperX.setAcceleration(100);     //X axis accelaration for homing

homingX = 1;
```

Εικόνα 4.7 Μέρος συνάρτησης Homing

Αρχίζοντας, ο κώδικας κινεί τον κινητήρα ένα βήμα την φορά προς τα πίσω μέχρι να ενεργοποιήσει τον διακόπτη. Όταν και για όση διάρκεια ο διακόπτης είναι πατημένος ο κινητήρας κινείται ένα βήμα τη φόρα με κατεύθυνση προς τα εμπρός αυτήν τη φορά. Όταν σταματήσει η επαφή με τον διακόπτη, ο κινητήρας σταματάει και η θέση αυτή ορίζεται ως μηδενική. Η διαδικασία αυτή πραγματοποιείται ταυτόχρονα για όλους τους κινητήρες που ορίζουν την θέση του σαρωτή. Ο κινητήρας του περιστρεφόμενου τραπεζιού δεν συμμετέχει σε αυτήν τη διαδικασία καθώς δε θεωρήθηκε απαραίτητο. Για τις κινήσεις που ορίζονται από αυτή τη συνάρτηση έχουμε ορίσει μικρές ταχύτητες και επιταχύνσεις με σκοπό την υψηλότερη ακρίβεια.

void receive Serial Data ()

Με αυτήν τη συνάρτηση δεχόμαστε τα δεδομένα από τη σειριακή θύρα -στην περίπτωση μας

από το λογισμικό που έχουμε δημιουργήσει- και τα αποθηκεύουμε πρώτα σε μεταβλητή τύπου char και στην συνέχεια σε μεταβλητή τύπου string. Η τελευταία είναι αυτή που επεξεργαζόμαστε και στη συνέχεια χρησιμοποιούμε για τον έλεγχο των κινητήρων. Εδώ να σημειώσουμε ότι η πληροφορία που στέλνεται από κάθε κινητήρα τελειώνει με '\n'. Αυτό μας βοηθάει στην διακριτοποίηση της πληροφορίας εισόδου.

```
void Receive_Serial_Dara() //function to read serial data
{
  while (Serial.available() > 0)
  {
    c = Serial.read();
    if (c == '\n') {
      break;
    }
    else        {
      dataIn += c;
    }
  }
}
```

Εικόνα 4.8 Συνάρτηση Receive_Serial_Data ()

void Parse_the_data ()

Στην συνάρτηση αυτή επεξεργαζόμαστε τις εισόδους του κώδικα και τις μετατρέπουμε σε μορφή κατάλληλη για τον χειρισμό των κινητήρων. Όταν στέλνουμε μια είσοδο στο πρόγραμμα αυτή ακολουθείται και από ένα γράμμα διαφορετικό για κάθε κινητήρα. Έτσι για τον κινητήρα X η πληροφορία που στέλνει το πρόγραμμα στον κώδικα είναι ("value"+"Q"+"\n"). Για τους υπόλοιπους κινητήρες τα γράμματα είναι R S T αντίστοιχα, ενώ για το Homing το γράμμα που στέλνεται είναι το H χωρίς κάποιο value.

Για την αντιστοιχιση της πληροφορίας εισόδου με κάποιον κινητήρα δημιουργήσαμε τις μεταβλητές indexOf(letter)=dataIn.indexOf("letter") για κάθε κινητήρα καθώς και για την διεργασία Homing. Η μεταβλητή αυτή έχει έξοδο -1 αν δεν υπάρχει το αντίστοιχο γράμμα στην είσοδο και 0 έαν υπάρχει. Έτσι ξέρουμε αν η πληροφορία που δεχόμαστε αντιστοιχεί σε κάποιον κινητήρα ή διεργασία.

Στη συνέχεια δημιουργούμε substrings της εισόδου για να έχουμε σε μια μεταβλητή μόνο την πληροφορία που αντιστοιχεί στον κάθε κινητήρα. Η εντολή κίνησης για τους κινητήρες δέχεται σαν είσοδο μεταβλητές τύπου Int, όποτε τελευταίο βήμα της επεξεργασίας είναι η μετατροπή της μεταβλητής σε μορφή int με την εντολή int_variable = substring.toInt(). Με

αυτή την διαδικασία έχουμε δημιουργήσει κατάλληλες μεταβλητές για τον έλεγχο των κινητήρων αλλά και την ενεργοποίηση την διαδικασίας homing.

```
void Parse_the_data() //parse data to int according stepper
{

  //search if character exists in dataIn
  indexOfQ = dataIn.indexOf('Q');
  indexOfR = dataIn.indexOf('R');
  indexOfS = dataIn.indexOf('S');
  indexOfT = dataIn.indexOf('T');
  indexOfH = dataIn.indexOf('H');

  //if caracter exist create a substring and then convert to int
  if (indexOfQ > -1) {
    str_stepperX = dataIn.substring(0, indexOfQ);
    stepperXvalue = str_stepperX.toInt();
  }
  if (indexOfR > -1) {
    str_stepperY = dataIn.substring(indexOfQ + 1, indexOfR);
    stepperYvalue = str_stepperY.toInt();
  }
  if (indexOfS > -1) {
    str_stepperZ = dataIn.substring(indexOfR + 1, indexOfS);
    stepperZvalue = str_stepperZ.toInt();
  }
  if (indexOfT > -1) {
    str_stepperT = dataIn.substring(indexOfS + 1, indexOfT);
    stepperTvalue = str_stepperT.toInt();
  }

}
```

Εικόνα 4.9 Συνάρτηση Parse_the_data ()

void loop ()

Είναι η κύρια συνάρτηση, η οποία τρέχει συνέχεια και σε επανάληψη. Σε αυτήν ορίσαμε τις λειτουργίες του προγράμματος και ποια συνάρτηση να καλεί σε περιπτώσεις που η είσοδος αναφέρεται σε κινητήρα ή στην διαδικασία homing.

Οι κώδικες τόσο του UI όσο και του Arduino βρίσκονται ολόκληροι στο παράρτημα της εργασίας.

# 5 Συμπεράσματα και Μελλοντική Εργασία

## 5.1 Συμπεράσματα

Η δημιουργία και ο σχεδιασμός ενός ολοκληρωμένου ηλεκτρομηχανικού συστήματος και του προγράμματος ελέγχου του είναι μια διαδικασία, η οποία εμπεριέχει τους κλάδους της μηχανικής της ηλεκτρονικής και του προγραμματισμού. Ο σχεδιασμός ενός τέτοιου συστήματος απαιτεί εξοικείωση με εργαλεία όλων αυτών των κλάδων. Έτσι, κατά τη διάρκεια της εργασίας αυτής υπήρξε τριβή με εργαλεία σχεδιασμού, προσομοίωσης και προγραμματισμού.

Η επιλογή μηχανισμών για τις κινήσεις ήταν ένα πολύ σημαντικό βήμα, με κάθε επιλογή μηχανισμού να επηρεάζει την σχεδίαση και την επιλογή μηχανισμού για τις επόμενες κινήσεις. Έτσι η καθεμιά από αυτές θα έπρεπε να λαμβάνει υπόψιν τον σχεδιασμό των επόμενων αξόνων αλλά και την φυσική σύνδεση μεταξύ τους. Στόχος της εργασίας επίσης ήταν ο σχεδιασμός ενός συστήματος με μειωμένο κόστος. Το τελικό κόστος εξαρτάται άμεσα από τα στοιχεία μετάδοσης της κίνησης κι έτσι επιλέχθηκε να έχουμε όσον το δυνατόν λιγότερα τέτοια εξαρτήματα. Τα εξαρτήματα μετάδοσης όμως, κυρίως τα προφίλ αλουμινίου, είναι υπεύθυνα για την στιβαρότητα και την ευστάθεια της κατασκευής. Έτσι ενώ θα μπορούσαμε να έχουμε μια ογκώδη κατασκευή με την στατικότητα της να προκύπτει από τους ίδιους τους μηχανισμούς κίνησης, για παράδειγμα δύο ράγες για την οριζόντια κίνηση με ένα μεγάλο φορείο και πάνω σε αυτό τέσσερις κάθετες ράγες οδηγοί, στις οποίες θα υπήρχε φορείο για την κατακόρυφη κίνηση, το σύστημα σχεδιάστηκε με τρόπο όπου η στατικότηκα προσφέρεται σε αυτό από πλαϊνές στηρίξεις με προφίλ αλουμινίου. Αυτό ανταποκρίνονταν στον αρχικό στόχο που δεν ήταν άλλος από το να έχουμε μειωμένο αριθμό εξαρτημάτων και ακολούθως μειωμένο κόστος χωρίς προβλήματα στατικότητας.

Κατά τη διάρκεια του σχεδιασμού μας απασχόλησε αρκετά ο μηχανισμός περιστροφής του σαρωτή. Πολλά αντίστοιχα συστήματα τοποθετούν τον σαρωτή σε τραπέζι, το οποίο με την βοήθεια γραμμικού οδηγού, ανυψώνεται από την μία μεριά δημιουργώντας την επιθυμητή κλίση στον σαρωτή. Αυτός ο μηχανισμός θα αύξανε επίσης το κόστος και την πολυπλοκότητα της κατασκευής. Έτσι επιλέχθηκε ο μηχανισμός τύπου ΄΄κούνια΄΄ ο οποίος είναι εμπνευσμένος από αντίστοιχους μηχανισμούς για περιστροφή φωτογραφικών καμερών.

Ένα αρκετά σημαντικό στοιχείο κατά τον προγραμματισμό του μικροελεγκτή ήταν η επιλογή κατάλληλης βιβλιοθήκης. Η βιβλιοθήκη AccelStepper μας πρόσφερε την δυνατότητα της λειτουργίας home και τη δημιουργία κινήσεων με χρήση συντεταγμένων και όχι απλής μετατόπισης. Χωρίς τη χρήση της συγκεκριμένης βιβλιοθήκης για να έχουμε τα ίδια αποτελέσματα θα έπρεπε να έχουμε αρκετά πιο ογκώδη κώδικα. Τέλος, ένα κομμάτι που ήθελε ιδιαίτερη ενασχόληση ήταν αυτό της μετατροπής δεδομένων που δέχονταν το Arduino μέσω serial port από το GUI σε δεδομένα κατάλληλα για να κινήσουν τους κινητήρες.

## 5.2 Μελλοντική Εργασία

Η εργασία αυτή κατάφερε να αυτοματοποιήσει την διαδικασία της σάρωσης μέχρι ενός σημείου. Οι μετακινήσεις του σαρωτή που γινόντουσαν χειροκίνητα τώρα θα γίνονται από το σχεδιασμένο σύστημα. Αυτό ευελπιστούμε να μειώσει σημαντικά τον συνολικό χρόνο της διαδικασίας αλλά και να προσδώσει μεγαλύτερη ακρίβεια, κάτι το οποίο μένει να αποδειχθεί στο φυσικό σύστημα. Η διαδικασία της σάρωσης έχει αυτοματοποιηθεί μέχρις ενός βαθμού καθώς οι επιλογές των θέσεων ορίζονται αποκλειστικά από τον χρήστη χωρίς κάποια καθοδήγηση από το λογισμικό. Ως συνέχεια της εργασίας θα μπορούσε να σχεδιαστεί ένα λογισμικό που θα προτείνει θέσεις σάρωσης βασισμένες σε υπάρχοντες αλγόριθμους ή ακόμα και να αυτοματοποιήσει πλήρως την διαδικασία. Όσον αφορά στο μηχανικό σύστημα η χρήση αισθητήρων απόστασης ή ακόμα και σερβοκινητήρων θα αύξανε ακόμα περισσότερο την ακρίβεια. Επίσης, η δημιουργία ενός κουτιού που θα εμπεριέχει όλα τα ηλεκτρονικά στοιχεία του συστήματος όπως τον μικροελεγκτή, τους οδηγούς κινητήρων και το τροφοδοτικό, σχεδιασμένο κατάλληλα για την απόρριψη θερμότητας θα ολοκλήρωνε το σύστημα. Τέλος, το σύστημα αυτό θα μπορούσε να λειτουργεί χωρίς την χρήση Η/Υ με τον χρήστη να δίνει εντολές στο σύστημα μέσω UI ενσωματωμένο στον μικροελεγκτή π.χ. (οθόνη αφής σε σύνδεση με το Arduino).

# 6 Βιβλιογραφία


[1] Abd-Raheem A., AlDeiri F., Alyaman M. "Design of an automated 3D scanner."*International Arab Conference on Information Technology (ACIT), 2018,* DOI:10.1109/ACIT.2018.8672679

[2] Alontseva L. D.. Ghassemieh E, Krasavin A.L, Kadyroldima T.A. "Development of 3D scanning system for robotics plasma processing of medical products with complex geometries" *Journal of electronics Science and Technology* 18 2020

[3] Banta, J.E., Wong, L.M., Dumont, C., Abidi, M.A.: A next-best view system for autonomous 3-D object reconstruction. *Syst.Man Cybern.* Part A Syst. Hum. IEEE Trans. 30(5), 589–598 (2000)

[4] Borangiu, T.,Dogar,A., Dumitrache, A.: Constraints-based motion planning for an automatic, flexible laser scanning robotized platform. In: *Automation, Quality and Testing, Robotics (AQTR) IEEE International Conference* on, vol. 2, pp. 65–69 (2008)

[5] Chen, S., Li, Y.F.,Wang,W., Zhang, J.: Active Sensor Planning for Multiview Vision Tasks. Springer, New York (2008)

[6] Connolly, C.I.: The determination of next best views. In: *Robotics and Automation. Proceedings of 1985 IEEE International Conference* on, vol. 2, pp. 432–435 (1985)

[7] Hudaib A., Al Azzam S. "A simple and inexpensive 3D scanning of Remote objects by Robot." *Modern Applied Science* 2018 pp 171-177 DOI:10.5539/mas.v12n4p171

[8] Javaid M., Haleem A., Singh R P., Suman R., "Industrial perspectives of 3d scanning: Features, roles and it's analytical applications" *Sensors International*, 2021

[9] Jiménez M. Palomera R. Couvertier I. "*Introduction to Embedded Systems"* Springer, New York, NY, https://doi.org/10.1007/978-1-4614-3143-5

[10] Khalfaoui, S., Aigueperse, A., Seulin, R., Fougerolle, Y., Fofi, D.: Fully automatic 3D digitization of unknown objects using progressive data bounding box. In: IS&T/SPIE Electronic Imaging, p.829011 (2012)

[11] Kazuyuki M. Takafumi A. "A robot-based 3D body scanning system using passive stereo vision" 2008. *15th IEEE International Conference on image processing,* 2008, DOI:10.1109/ICIP.2008.4711752

[12] Linder L., Sergiyenko O., Rodriguez-Quinoez J., Vera Tyrsa V., Mercorelli P., Fuentes W.F, Muerrieta-Rico F.N., Nieto-Hipolito J. "Continuous 3D scanning mode using servomotors istead of stepping motor in dynamic laser triangulation*" 24th International Symposium on Industrial Electronics (ISIE)* 2015

[13] Maver, J., Bajcsy, R.: Occlusions as a guide for planning the next view. *Pattern Anal. Mach. Intell.* IEEE Trans. 15(5), 417–433 (1993)

[14] Olague, G.: Planification du placement de caméras pour des mesures 3D de précision. Institut National Polytechnique de Grenoble-INPG (1998)

[15] Pito, R., Bajcsy, R.K.: Solution to the next best view problem for automated cad model acquisiton of free-form objects using range cameras. In: *Photonics East* '95, pp. 78–89 (1995)

[16] Reinhart G., Tekouo W. "Automatic programming of robot mounted 3d optical scanning devices to easily measure parts in high-variant assembly." *CIRPS Annals - Manufacturing Technology* 2009 Elsevier pp 25-28 DOI: 10.1016/j.cirp.2009.03.125

[17] Tarbox, G.H., Gottschlich, S.N.: Planning for complete sensor coverage in inspection. *Comput. Vis. Image Underst*. 61(1), 84–111 (1995)

[18] Wagner M., Heβ P., Reitelshofer S., Franke J. “3D Scanning of Workpieces with Cooperative Industrial Robot Arms.” *47th International Symposium on Robotics.* 2016

[19] Xie, S., Calvert, T., Bhattacharya, B.: Planning views for the incremental construction of body models. In: *Proceedings of the International Conference on Pattern Recognition*, Paris, pp. 154–157 (1986)

[20] Zhao Y., Zhao L., Zhang L.,Qi Li “Development of a Robotics 3D Scanning System for Reverse Engineering of Freeform Part.” *International Conference on Advanced Computer Theroy and Engineering.* (2008)

[21] Τύρης Δημήτρης, “Σχεδιασμός λήψεων στην τρισδιάστατη σάρωση αντικειμένων με χρήση λευκού φωτός” dspace.lib.ntua.gr(αρχείο μεταπτυχιακών εργασιών ΕΜΠ) (2015)

## Παράρτημα Α

Κώδικας Arduino

```
#include <AccelStepper.h>   //library for stepper motor


//Global Varialbles//
char c;
String dataIn;
//Parse variables//
uint8_t stepperXvalue, stepperYvalue, stepperZvalue,
stepperTvalue ;
int8_t indexOfQ, indexOfR, indexOfS, indexOfT, indexOfH ;
String str_stepperX, str_stepperY, str_stepperZ, str_stepperT;
// Define the steppers and the pins they will use//
AccelStepper stepperX(1, 4, 3);                 // Stepper for x
axis
AccelStepper stepperY(1, 5, 6);                 // Stepper for y
axis
AccelStepper stepperZ(1, 7, 8);                 // Stepper for z
rotation
AccelStepper stepperT(1, 12, 13);               // Stepper for
table rotation

// Define home switch pins for each axis (only for the scanner)
#define homeswitchX 9  //switch PIN x axis
#define homeswitchY 10 //switch PIN y axis
#define homeswitchZ 11 //switch PIN z rotation

//Homing Variables
int  finished_move = 1;   //checks is homing is ok
long homingX = -1;         //move stepper to home position
long homingY = -1;         //move stepper to home position
long homingZ = -1;         //move stepper to home position



void Homing()
{
  //////////////Homing
StepperX////////////////////////////////////////////////////////
///
  while (digitalRead(homeswitchX))     // stepper moves backwards
```

```
until hits the switch
  { stepperX.setMaxSpeed(250);        //X axis speed for homing
    stepperX.setAcceleration(100);    //X axis accelaration for
homing

    stepperX.moveTo(homingX); // decrease one step at a time
    homingX--;
    stepperX.run();          // runs stepper
    delay(5);
  }

  stepperX.setCurrentPosition(0);    //  Set the current
possition as 0
  stepperX.setMaxSpeed(250);        //X axis speed for homing
  stepperX.setAcceleration(100);    //X axis accelaration for
homing

  homingX = 1;

  while (!digitalRead(homeswitchX)) { //moves stepper forworad
until switch is deactiavated
    stepperX.moveTo(homingX);
    stepperX.run();
    homingX++;
    delay(5);
  }

  stepperX.setCurrentPosition(0);
  stepperX.setMaxSpeed(900);        //set values as setup (for
regular movement)
  stepperX.setAcceleration(150);    //set values as setup (for
regular movement)

  ////////////Homing X
complete////////////////////////////////////////////////////////////
//

  ////////////Homing
StepperY////////////////////////////////////////////////////////////
///
```

```
  while (digitalRead(homeswitchY))     // stepper moves backwards
until hits the switch
  { stepperY.setMaxSpeed(250);        //Y axis speed for homing
    stepperY.setAcceleration(100);    //Y axis accelaration for
homing

    stepperY.moveTo(homingY); // decrease one step at a time
    homingY--;
    stepperY.run();          // runs stepper
    delay(5);
  }

  stepperY.setCurrentPosition(0);    //  Set the current
possition as 0
  stepperY.setMaxSpeed(250);        //Y axis speed for homing
  stepperY.setAcceleration(100);    //Y axis accelaration for
homing

  homingY = 1;

  while (!digitalRead(homeswitchY)) { //moves stepper forworad
until switch is deactiavated
    stepperY.moveTo(homingY);
    stepperY.run();
    homingY++;
    delay(5);
  }

  stepperY.setCurrentPosition(0);
  stepperY.setMaxSpeed(900);        //set values as setup (for
regular movement)
  stepperY.setAcceleration(150);    //set values as setup (for
regular movement)

  //////////////Homing Y
complete////////////////////////////////////////////////////////////
/

  //////////////Homing
StepperZ////////////////////////////////////////////////////////////
```

```
///
  while (digitalRead(homeswitchZ))     // stepper moves backwards
until hits the switch
  { stepperZ.setMaxSpeed(250);        //Z axis speed for homing
    stepperZ.setAcceleration(100);    //Z axis accelaration for
homing

    stepperZ.moveTo(homingZ); // decrease one step at a time
    homingZ--;
    stepperZ.run();          // runs stepper
    delay(5);
  }

  stepperZ.setCurrentPosition(0);   //  Set the current possition
as 0
  stepperZ.setMaxSpeed(250);        //Z axis speed for homing
  stepperZ.setAcceleration(100);    //Z axis accelaration for
homing

  homingZ = 1;

  while (!digitalRead(homeswitchZ)) { //moves stepper forworad
until switch is deactiavated
    stepperZ.moveTo(homingZ);
    stepperZ.run();
    homingZ++;
    delay(5);
  }

  stepperZ.setCurrentPosition(0);
  stepperZ.setMaxSpeed(900);        //set values as setup (for
regular movement)
  stepperZ.setAcceleration(150);    //set values as setup (for
regular movement)
}
////////////Homing Y
complete////////////////////////////////////////////////////////

////////////Homing
COMPLETE//////////////////////////////////////////////////////////
```

```
void Stepper_Possition()
{

  // Set max speed and accelaration for each stepper
  stepperX.setMaxSpeed(900);        //X axis max speed
  stepperX.setAcceleration(150);    //X axis accelaration

  stepperY.setMaxSpeed(900);        //Y axis max speed
  stepperY.setAcceleration(150);    //Y axis accelaration

  stepperZ.setMaxSpeed(900);         //Z axis max speed - without
use of gear
  stepperZ.setAcceleration(150);     //Z axis accelaration -
without use of gear

  stepperT.setMaxSpeed(900);        //T axis motor max speed
  stepperT.setAcceleration(150);    //T axis motor accelaration

  ///////////////Set Steppers
possition/////////////////////////////////////////////////
  stepperX.moveTo(25 * stepperXvalue); //lead screw lead = 8mm
-- onew revolution stepper 200 steps -----   1mm=25steps
  stepperX.run();

  stepperY.moveTo(25 * stepperYvalue);
  stepperY.run();

  stepperZ.moveTo(27.8 * stepperZvalue);   //gear 50:1 , stepper
1.8 degrees
  stepperZ.run();

  stepperT.moveTo(3 * stepperTvalue);      //gear 6:1 , stepper 1.
8 degrees
  stepperT.run();

}
```

```
void Receive_Serial_Dara() //function to read serial data
{
  while (Serial.available() > 0)
  {
    c = Serial.read();
    if (c == '\n') {
      break;
    }
    else        {
      dataIn += c;
    }
  }
}

void Parse_the_data() //parse data to int according stepper
{

  //search if character exists in dataIn
  indexOfQ = dataIn.indexOf('Q');
  indexOfR = dataIn.indexOf('R');
  indexOfS = dataIn.indexOf('S');
  indexOfT = dataIn.indexOf('T');
  indexOfH = dataIn.indexOf('H');

  //if caracter exist create a substring and then convert to int
  if (indexOfQ > -1) {
    str_stepperX = dataIn.substring(0, indexOfQ);
    stepperXvalue = str_stepperX.toInt();
  }
  if (indexOfR > -1) {
    str_stepperY = dataIn.substring(indexOfQ + 1, indexOfR);
    stepperYvalue = str_stepperY.toInt();
  }
  if (indexOfS > -1) {
    str_stepperZ = dataIn.substring(indexOfR + 1, indexOfS);
    stepperZvalue = str_stepperZ.toInt();
  }
  if (indexOfT > -1) {
    str_stepperT = dataIn.substring(indexOfS + 1, indexOfT);
    stepperTvalue = str_stepperT.toInt();
```

```
  }

}

void setup() {
  Serial.begin(9600);

  pinMode(homeswitchX, INPUT_PULLUP);
  pinMode(homeswitchY, INPUT_PULLUP);
  pinMode(homeswitchZ, INPUT_PULLUP);

  delay(5);
}

void loop() {
  // put your main code here, to run repeatedly:
  Receive_Serial_Dara();                                  // receive
data function
  if (c == '\n')
  {
    Parse_the_data();
    c = 0;
    dataIn = "";
  }
  if (indexOfH > -1) { // if H exist in data receive then start
homing..
    void Homing();
  }
  else {
    void Stepper_Possition();
  }
Serial.print(str_stepperY);
}
```

Κώδικας UI Visual Studio (C#)

C:\Users\giorg\source\repos\thesis 2nd\thesis 2nd\Form1.cs 1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace thesis_2nd
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            groupBox_stepper.Enabled = false;
            groupBox_Homming.Enabled = false;
        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void buttonHome_Click(object sender, EventArgs e)
        {

            serialPort1.Write("H" + "\n");

        }

        private void comboBox_dropdown_DropDown(object sender, EventArgs e)
        {
            string[] portlists = SerialPort.GetPortNames();
            comboBox_com.Items.Clear();
            comboBox_com.Items.AddRange(portlists);

        }

```

```
57          private void button_COM_Click(object sender, EventArgs e)
58          {
59              try
60              {
61                  serialPort1.PortName = comboBox_com.Text;
62                  serialPort1.BaudRate = 9600;
63                  serialPort1.Open();
64
65                  MessageBox.Show("Επιτυχία σύνδεσης με μικροελεγκτή!");
66                  groupBox_stepper.Enabled = true;
67                  groupBox_Homming.Enabled = true;
68
69              }
70              catch (Exception error)
71              {
72                  MessageBox.Show(error.Message);
73              }
74          }
75
76          private void Form1_FormClosing(object sender, FormClosingEventArgs e)
77          {
78
79              try
80              {
81                  if (serialPort1.IsOpen)
82                  {
83                      serialPort1.Close();
84                  }
85              }
86
87              catch (Exception error)
88              {
89                  MessageBox.Show(error.Message);
90              }
91
92  }
93
94          private void f(object sender, EventArgs e)
95          {
96
97          }
98
99          private void textBoxXaxis_TextChanged(object sender, EventArgs e)
100         {
101
102         }
103
104         private void buttonX_Click(object sender, EventArgs e)
105         {
106             // send thesis value to steper
107             string m1 = 'Q' + textBoxXaxis.Text;
108             serialPort1.Write(m1 + "\n");
109
110         }
111
112         private void buttonY_Click(object sender, EventArgs e)
```

```
113         {
114             // send thesis value to steper
115             string m2 = 'R' + textBoxYaxis.Text;
116             serialPort1.Write(m2 + "\n");
117         }
118 
119         private void buttonZ_Click(object sender, EventArgs e)
120         {
121             // send thesis value to steper
122             string m3 = 'S' + textBoxZaxis.Text;
123             serialPort1.Write(m3 + "\n");
124         }
125 
126         private void buttonTable_Click(object sender, EventArgs e)
127         {
128             // send thesis value to steperT
129             string m4 = 'T' + textBoxTable.Text;
130             serialPort1.Write(m4 + "\n");
131         }
132 
133         private void button2_Click(object sender, EventArgs e)
134         {
135 
136         }
137 
138         private void label5_Click(object sender, EventArgs e)
139         {
140 
141         }
142     }
143 }
144 
```