Δ Ι Ε Θ Ν Ε Σ Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Τ Η Σ Ε Λ Λ Α Δ Ο Σ
Σ Χ Ο Λ Η Μ Η Χ Α Ν Ι Κ Ω Ν
Τ Μ Η Μ Α Μ Η Χ Α Ν Ι Κ Ω Ν Π Λ Η Ρ Ο Φ Ο Ρ Ι Κ Η Σ ,
Υ Π Ο Λ Ο Γ Ι Σ Τ Ω Ν Κ Α Ι Τ Η Λ Ε Π Ι Κ Ο Ι Ν Ω Ν Ι Ω Ν

ΥΛΟΠΟΙΗΣΗ ΔΙΑΤΑΞΗΣ CNC ΔΥΟ ΑΞΟΝΩΝ ΜΕ ΜΙΚΡΟΕΛΕΓΚΤΗ ATMEGA 328P

Πτυχιακή Εργασία του

Ιακώβου Κ. Πέτρος (3205)

Μηχανολόγος Μηχανικός
MS: Προηγμένα Βιομηχανικά Συστήματα Παραγωγής

Επιβλέπων: Ι. Καλόμοιρος, Καθηγητής

ΣΕΡΡΕΣ, Σεπτέμβριος 2021

Υπεύθυνη Δήλωση *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Σερρών της Ελλάδας.*

Περίληψη

Η πτυχιακή εργασία έγινε στα πλαίσια του προγράμματος σπουδών του τμήματος Πληροφορικής και Επικοινωνιών του ΑΤΕΙ Σερρών. Πρόκειται για την υλοποίηση ενός CNC δύο αξόνων, και επικεντρώνεται:

1) Στην ανάπτυξη λογισμικού CAM, το οποίο διαβάζει αρχεία CAD, ανακτά συντεταγμένες γεωμετρικών στοιχείων και δημιουργεί αρχείο κειμένου .nc που περιλαμβάνει τον κώδικα G του CAD αρχείου.

2) Στην ανάπτυξη (σχεδιασμός και υλοποίηση) hardware και software, CNC δύο αξόνων (x , y).

3) Στην εφαρμογή αλγορίθμου γραμμικής και κυκλικής παρεμβολής.

Κατά την λειτουργία: γίνεται εκτέλεση ενός προγράμματος εφαρμογής, το οποίο επεξεργάζεται δεδομένα, από δοθέντα αρχείο CAD. Μέσω σειριακής επικοινωνίας USART, το πρόγραμμα στέλνει τα δεδομένα στον μικροελεγκτή m328p (Atmega). Ο μικροελεγκτής, ανάλογα με τα δεδομένα, που θα δεχτεί εκτελεί αντίστοιχες ρουτίνες, και μέσω του DRV8825 driver στέλνει κατάλληλους παλμούς στα βηματικά μοτέρ. Ανάλογα με το αν χρειάζεται να γίνει ή όχι κατεργασία, ο ελεγκτής ενεργοποιεί ή απενεργοποιεί την κεφαλή (τρυπάνι, λέιζερ, ακροφύσιο, πένα κ.α.) αντίστοιχα.

Για τον πρακτικό έλεγχο της σωστής λειτουργίας, και την αξιολόγηση του αλγόριθμου που χρησιμοποιήθηκε, μετρήθηκε (σε πολλούς διαφορετικούς συνδυασμούς κινήσεων) ή ακρίβεια σε θέση (με παχύμετρο) καθώς και η ακρίβεια σε ταχύτητα (με χρονόμετρο). Ως συμπεράσματα επισημαίνονται κάποιες αστοχίες στην κατασκευή και περιορισμοί που θέτει το υλικό που χρησιμοποιήθηκε (π.χ. καταχωρητές 8bit). Τέλος γίνεται αναφορά σε τρόπους περαιτέρω εξέλιξης και βελτίωσης σε επίπεδο λογισμικού.

Τα μηχανολογικά σχέδια, καθώς και η μηχανολογική μελέτη του CNC, περιλαμβάνονται σε ξεχωριστό φάκελο.

Επικοινωνία: berxt@hotmail.com

Abstract

The following paper is the final project for the Department of Computer, Informatics and Telecommunications Engineering, of International Hellenic University. The project focus on:

1) The development of a CAM software, which reads CAD files, extract coordinates from geometrical shapes and generates a text file (.nc) with the corresponding G code.

2) The development (planning and implementation), hardware and software, of a two axis CNC.

3) The analysis of linear and circular interpolation algorithms.

Key Words: CAD, CAM , CNC , Numerical Control Interpolation

YouTube link: https://www.youtube.com/watch?v=j8LD028rm2o

Contact: berxt@hotmail.com

Περιεχόμενα

Πίνακας Εικόνων - Σχεδίων

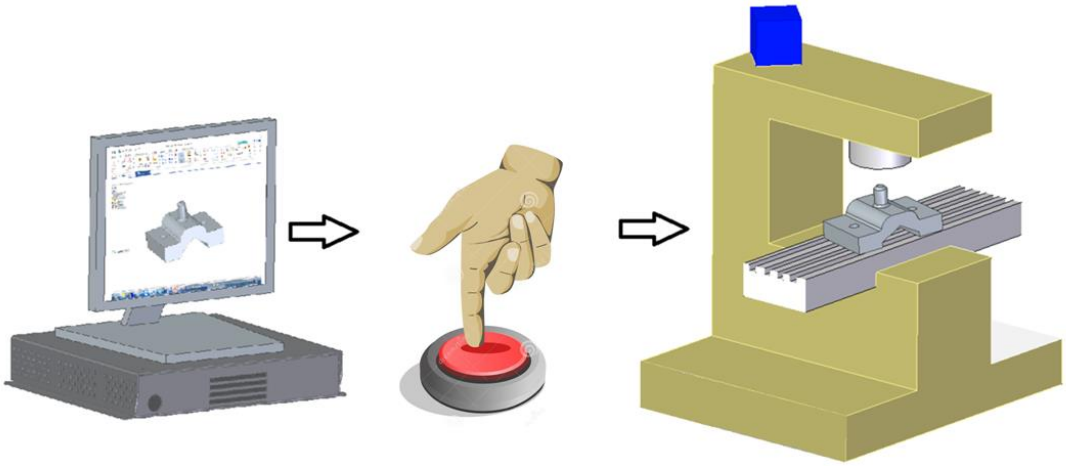
Πίνακας πινάκων

Εισαγωγή

Η βιομηχανία άλλαξε ριζικά στις αρχές του 1950 με την εμφάνιση των μηχανών αριθμητικού ελέγχου όπου αυτοματοποίησαν την παραγωγή αυξάνοντας την ποιότητα και την παραγωγικότητα. Ο όρος Computerized Numerical Control (CNC), αναφέρεται στον έλεγχο μηχανικής κίνησης μέσω ηλεκτρονικού υπολογιστή.

Τα βασικά πλεονεκτήματα των CNC είναι: ποιότητα, ταχύτητα, αυτοματισμός, επαναληψιμότητα, μείωση φθοράς εργαλείων κατεργασίας. Όλα τα παραπάνω οδηγούν σε αύξηση παραγωγικότητας και διασφάλιση ποιότητας. Επιπλέον υπάρχει δυνατότητα κατασκευής εξαιρετικά πολύπλοκων εξαρτημάτων που θα ήταν αδύνατη η κατασκευή τους με άλλον τρόπο.

Σήμερα δεν νοείται επιβίωση βιομηχανίας που δεν χρησιμοποιεί CNC μηχανήματα. Γενικότερα η τάση είναι να γίνει η παραγωγική διαδικασία όσο το δυνατόν πιο αυτοματοποιημένη, περιορίζοντας στο ελάχιστο τον ανθρώπινο παράγοντα. Η κεντρική ιδέα είναι: σχεδίαση → πάτημα ενός κουμπιού → παραγωγή προϊόντος.



Εικόνα. 1.1 - Σχεδιασμός - Παραγωγή

Και υπάρχουν αρκετές βιομηχανίες που το έχουν καταφέρει αυτό σε μεγάλο ποσοστό ολοκλήρωσης. Ο αυτοματισμός αυτός περιλαμβάνει τρία βασικά στάδια:

1) CAD (**C**omputer **A**ided **D**esign): Για το σχεδιασμό χρησιμοποιείται λογισμικό 2D/3D σχεδίασης. Το αρχείο σχεδίου περιέχει πληροφορίες για το εξάρτημα που σχεδιάστηκε όπως γεωμετρικά χαρακτηριστικά , φυσικές ιδιότητες (υλικό, βάρος, κέντρο βάρους), μηχανικές ιδιότητες (ελαστικότητα, θερμική αγωγιμότητα, ηλεκτρική αγωγιμότητα) κ.α. Τα 3D λογισμικά σχεδίασης προσφέρουν τόσο ολοκληρωμένη και ρεαλιστική εικόνα του εξαρτήματος που θα μπορούσε να πει κανείς πως το μόνο που δεν προσφέρεται είναι η φυσική αίσθηση του εξαρτήματος.

2) CAM (**C**omputer **A**ided **M**anufacturing): Το λογισμικό CAM εκμεταλλεύεται τα δεδομένα του αρχείου σχεδίου ώστε να αναπαράγει κώδικα, κατάλληλο για παραγωγή

του εξαρτήματος σε CNC μηχάνημα. Παρόλο που η μορφή αυτού του κώδικα είναι τυποποιημένη, υπάρχουν διαφοροποιήσεις ανάλογα με τον κατασκευαστή της μηχανής. Για τον λόγω αυτό πρέπει να δηλώνεται το συγκεκριμένο μηχάνημα στο οποίο θα γίνει η κατεργασία (postprocessing), ώστε ο κώδικας που θα παραχθεί να είναι κατάλληλα διαμορφωμένος στις απαιτήσεις του εκάστοτε μηχανήματος. Στα περισσότερα πακέτα λογισμικού CAM υπάρχει η δυνατότητα εικονικής προσομοίωσης της κατεργασίας.

3) NC (**N**umerical **C**ontrol): Ο αριθμητικός έλεγχος αναφέρεται στην διερμηνεία του κώδικα για την λειτουργία του CNC. Στους αλγόριθμους δηλαδή που χρησιμοποιούνται για την επίλυση εξισώσεων κίνησης. Ανάλογα με το hardware, και τον τρόπο με τον οποίο η κάθε μηχανή το εκμεταλλεύεται χρησιμοποιούνται τα αποτελέσματα των εξισώσεων για να παραχθούν κατάλληλοι παλμοί για την κίνηση των μοτέρ.

Εικόνα. 1.2 - CAD -> CAM -> CNC

Η εργασία αυτή αποτελεί μια προσωπική προσέγγιση 1) σχεδιασμού hardware, 2) ανάπτυξης software και 3) ανάπτυξης αλγορίθμου γραμμικής και κυκλική παρεμβολής και αποτελείται από πέντε κεφάλαια:

1. Computer Aided Manufacturing (CAM)

Στο πρώτο κεφάλαιο γίνεται αναφορά στο πρότυπο ISO 6983 το οποίο περιγράφει την μορφή του κώδικά για μηχανές αριθμητικού ελέγχου. Εξετάζεται η δομή των αρχείων CAD.

2. Hardware

Στο δεύτερο κεφάλαιο παρουσιάζονται τα βασικά τεχνικά χαρακτηριστικά του hardware. Αρχή λειτουργίας βηματικού μοτέρ και σέρβο μοτέρ, οδηγοί μοτέρ (driver DRV8825), αρχή λειτουργία της γέφυρας Η και ο μικροελεγκτής m328p, που είναι ο εγκέφαλος του CNC.

3. USART (Universal Synchronous Asynchronous Receiver Transmitter)

Γίνεται αναφορά στα διάφορα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται για την διασύνδεση μικροελεγκτή με άλλες εξωτερικές συσκευές, όπως άλλους μικροελεγκτές, αισθητήρια κ.α. Δίνεται μια πιο λεπτομερή περιγραφή της επικοινωνίας USART που χρησιμοποιήθηκε για την επικοινωνία του προγράμματος εφαρμογής με τον μικροελεγκτή.

4. Software (Αλγόριθμοι γραμμικής και κυκλικής παρεμβολής)

Αναφορά στα βασικά είδη των αλγορίθμων γραμμικής και κυκλικής παρεμβολής. Αναφορά σε δύο διαφορετικές προσεγγίσεις διερμηνείας του κώδικα G (αλγόριθμοι παλμού και αλγόριθμοι δυαδικής λέξης).

5. Υλοποίηση και Εφαρμογή

Αναλυτική περιγραφή της υλοποίησης του CNC. Εξαγωγή δεδομένων από αρχείο σχεδίου, δημιουργία κώδικα G, εφαρμογή αλγορίθμου γραμμικής και κυκλικής παρεμβολής, προγραμματισμός μικροελεγκτή.

Για την υλοποίηση της διάταξης έγινε μηχανολογική μελέτη (υπολογισμός ροπής) και μηχανολογικός σχεδιασμός, σε προϋπάρχουσα δουλειά. Επίσης έγινε ανάπτυξη κωδικοποίησης προϊόντος ώστε το κάθε εξάρτημα στον σχεδιασμό να περιγράφεται από τον δικό του μοναδικό κωδικό.

1. Computer Aided Manufacturing (CAM)

1.1 ISO 6983 [1]

Ο Οργανισμός EIA (Electronic Industries Association) έχει αναπτύξει το πρότυπο EIA-274-D (αργότερα ISO 6983) το οποίο τυποποιεί την μορφή του κώδικα που προορίζεται για έλεγχο μηχανών αριθμητικού ελέγχου. Το πρότυπο αυτό είναι γνωστό ως G/M Code και περιλαμβάνει.

- πληροφορίες για την κίνηση (είδος και συντεταγμένες).

- πληροφορίες σχετικά με την πρόωση (ταχύτητα κατεργασίας)

- πληροφορίες για τον έλεγχο της μηχανής

1.1.1 Εντολές δρομολόγησης - G Code

Στον πίνακα 1.1 φαίνονται οι εντολές που χρησιμοποιήθηκαν για την υλοποίηση του CNC και σχετίζονται με την κίνηση.

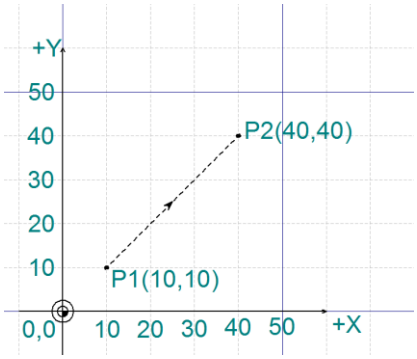
Κωδικός	Περιγραφή
G00	Γραμμική κίνηση με μέγιστη ταχύτητα
G01	Γραμμική κίνηση με ταχύτητα κατεργασίας
G02	Κυκλική δεξιόστροφη κίνηση με ταχύτητα κατεργασίας
G03	Κυκλική αριστερόστροφη κίνηση με ταχύτητα κατεργασίας
G04	Παύση κίνησης με την κεφαλή ενεργοποιημένη

Πίνακας 1.1 - Βασικές εντολές περιγραφής κίνησης κώδικά G

- G00: Δηλώνει ευθύγραμμη κίνηση με την μέγιστη ταχύτητα στο επιθυμητό σημείο. Χρησιμοποιείται στην περίπτωση που θέλουμε να μεταφέρουμε την κεφαλή σε συγκεκριμένη θέση χωρίς να υπάρχει κατεργασία.

Παράδειγμα: μετατόπιση χωρίς κατεργασία από την αρχική θέση P1(x1:10,y1:10) στην τελική θέση P2(x2:40,y2:40)

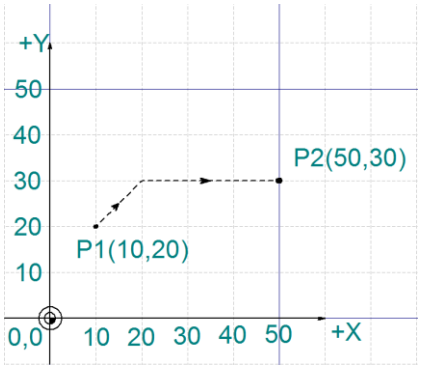
Εντολή:N001 G00 X40 Y40



Σχέδιο 1.1 - G00, Γραμμική κίνηση χωρίς κατεργασία με μέγιστη ταχύτητα στον άξονα με την μεγαλύτερη μετατόπιση

Το πρότυπο αναφέρει πως η κίνηση στον άξονα με την μεγαλύτερη μετατόπιση γίνεται με την μέγιστη ταχύτητα και η ταχύτητα στους άλλους άξονες προσαρμόζεται ανάλογα, ώστε να φθάσουν στο τελικό σημείο ταυτόχρονα.

Στον σχεδιασμό που έγινε για το CNC και οι δύο άξονες έχουν την μέγιστη ταχύτητα. Συνεπώς ο άξονας με την μικρότερη μετατόπιση φθάνει πρώτος στο τελικό σημείο και ακολουθεί ο άξονας με την μεγαλύτερη μετατόπιση.

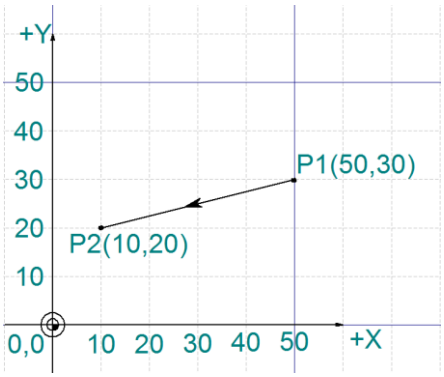


Σχέδιο 1.2 - G00, Γραμμική κίνηση με μέγιστη ταχύτητα και στους δύο άξονες

- G01: Δηλώνει ευθύγραμμη κίνηση με ταχύτητα κατεργασίας στο επιθυμητό σημείο. Χρησιμοποιείται στην περίπτωση που θέλουμε η κεφαλή να κινηθεί σε συγκεκριμένη ευθύγραμμη τροχιά με συγκεκριμένη ταχύτητα κατεργασίας. Η κεφαλή είναι ενεργοποιημένη.

Παράδειγμα: ευθύγραμμη κατεργασία από την αρχική θέση P1(x1:50,y1:30) στην τελική θέση P2(x2:10,y2:20) με ταχύτητα v=20 mm/sec.

Εντολή:N002 G01 X10 Y20 F20



Σχέδιο 1.3 - G01, Ευθύγραμμη κατεργασία

Οι ταχύτητες στον κάθε άξονα προσαρμόζονται ανάλογα έτσι ώστε η συνισταμένη ταχύτητα να είναι ίση με την επιθυμητή ταχύτητα κατεργασίας. Συνεπώς οι δύο άξονες έρχονται στο τελικό σημείο ταυτόχρονα

- G02: Δηλώνει κυκλική δεξιόστροφη κίνηση με ταχύτητα κατεργασίας στο επιθυμητό σημείο. Χρησιμοποιείται στην περίπτωση που θέλουμε η κεφαλή να κινηθεί σε συγκεκριμένη δεξιόστροφη κυκλική τροχιά και με συγκεκριμένη ταχύτητα κατεργασίας. Η κεφαλή είναι ενεργοποιημένη.

Παράδειγμα: κυκλική δεξιόστροφη κατεργασία με

αρχική θέση: P1(x1:20,y1:20)

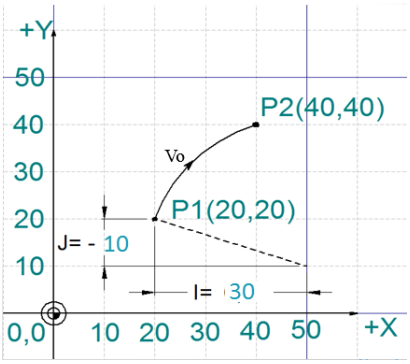
τελική θέση: P2(x2:40,y2:40)

σχετική συντεταγμένη κέντρο τόξου από την αρχική θέση κατά X: I=30

σχετική συντεταγμένη κέντρο τόξου από την αρχική θέση κατά Y: J= -10

ταχύτητα κατεργασίας: v=20 mm/sec.

Εντολή:N003 G02 X40 Y40 I30 J-10 F20



Σχέδιο 1.4 - G02, κατεργασία, δεξιόστροφο τόξο

Οι ταχύτητες στον κάθε άξονα προσαρμόζονται ανάλογα έτσι ώστε η συνισταμένη ταχύτητα να είναι ίση με την επιθυμητή ταχύτητα κατεργασίας. Συνεπώς οι δύο άξονες έρχονται στο τελικό σημείο ταυτόχρονα

- G03: Ισχύει ότι και στην εντολή G02 με την διαφορά ότι η κίνηση είναι αριστερόστροφη

Παράδειγμα: κυκλική αριστερόστροφη κατεργασία με

αρχική θέση: P1(x1:40,y1:40)

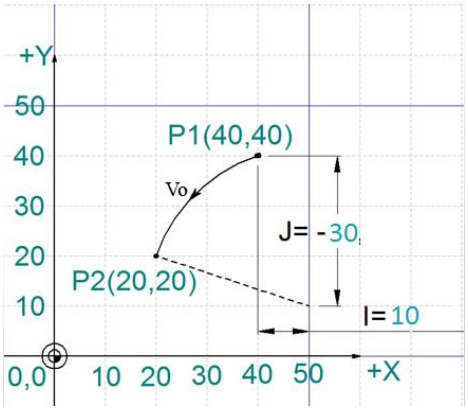
τελική θέση: P2(x2:20,y2:20)

σχετική συντεταγμένη κέντρο τόξου από την αρχική θέση κατά X: I= 10

σχετική συντεταγμένη κέντρο τόξου από την αρχική θέση κατά Y: J= -30

ταχύτητα κατεργασίας: v=20 mm/sec.

Εντολή:N004 G03 X20 Y20 I10 J-30 F20



Σχέδιο 1.5 - G03, κατεργασία, αριστερόστροφο τόξο

- G04: Δηλώνει παύση της κίνησης ενώ ταυτόχρονα η κεφαλή είναι ενεργοποιημένη, για καθορισμένο χρονικό διάστημα σε sec.

1.1.2 Συμπληρωματικές Εντολές Ελέγχου Κατάστασης Μηχανής

- M Code

Ο κώδικας M περιέχει βοηθητικές εντολές που σχετίζονται με την κατάσταση της μηχανής . Οι βασικές εντολές που χρησιμοποιήθηκαν είναι:

Κωδικός	Περιγραφή
M4	Ενεργοποίηση Κεφαλής
M5	Απενεργοποίηση Κεφαλής
M6	Αλλαγή Εργαλείου
M8	Ενεργοποίηση Ψύξης
M9	Απενεργοποίηση Ψύξης
M30	Τέλος προγράμματος

Πίνακας 1.2 - Βασικές εντολές ελέγχου κατάστασης μηχανής κώδικάς M

- Πρόωση F

Η πρόωση είναι η ταχύτητα κατεργασίας και δηλώνεται σε mm/rev. ή σε mm/sec. Για παράδειγμα F100 = 100mm/sec

1.2 Computer Aided Design (CAD)

Τα λογισμικά σχεδίασης πλέον έχουν εξελιχθεί σε βαθμό που προσφέρουν μια ολοκληρωμένη περιγραφή του εξαρτήματος (γεωμετρικά χαρακτηριστικά, φυσικές και μηχανικές ιδιότητες). Το γηγενές αρχείο ενός προγράμματος, περιέχει όλες αυτές τις πληροφορίες σε κωδικοποιημένη μορφή και μπορεί να διαβαστεί στις περισσότερες περιπτώσεις μόνο από το ίδιο το λογισμικό που δημιούργησε το αρχείο.

Για παράδειγμα τα αρχεία AutoCAD έχουν κατάληξη .dwg .Αν προσπαθήσουμε να ανοίξουμε ένα αρχείο .dwg με κάποιο άλλο πρόγραμμα θα προκύψει σφάλμα. Επίσης αν δοκιμάσουμε να ανοίξουμε ένα αρχείο .dwg με το notepad θα εμφανιστούν διάφορα ακαταλαβίστικα σύμβολα.

Αυτό το "δέσιμο" αρχείου με πρόγραμμα αποτελεί περιοριστικό παράγοντα, σε ένα περιβάλλον αυτοματισμού που απαιτεί η πληροφορία να είναι άμεσα διαθέσιμη. Είναι σημαντικό το αρχείο που σχεδιάστηκε σε ένα πρόγραμμα να μπορεί να χρησιμοποιηθεί και από άλλα προγράμματα άμεσα. Για τον λόγο αυτό όλα τα λογισμικά σχεδίασης δίνουν την δυνατότητα αποθήκευσης του αρχείου σε μορφή κατάλληλη για επεξεργασία από άλλα λογισμικά.

Τα αρχεία αυτού του τύπου είναι αρχεία κειμένου ASCII και περιέχουν όλες τις πληροφορίες για το εξάρτημα σε μορφή κειμένου. Αυτό δίνει την δυνατότητα να μπορεί να διαβαστεί το αρχείο και να εξάγουμε από αυτό τις πληροφορίες που χρειαζόμαστε.

Όσο αναφορά το 3D σχέδιο υπάρχουν αρκετοί τύποι αρχείων κατάλληλα για μεταφερσιμότητα. Οι πιο διαδεδομένοι τύποι είναι τα αρχεία **.step** η **.obj** (για 3D). Στο δυσδιάστατο σχέδιο έχει επικρατήσει κυρίως η μορφή **.dxf**.

1.2.1 Drawing Exchange Format (DXF)

Το Drawing Exchange Format αποτελεί ένα format αρχείων CAD που αναπτύχθηκε το 1982 από την AutoDesk με στόχο την μεταφερσιμότητα αρχείων σχεδίου .dwg. Πρόκειται για αρχείο κειμένου το οποίο περιέχει όλη την πληροφορία που ορίζει ο χρήστης κατά την σχεδίαση, σε μορφή ASCII (πάχος, τύπος, χρώμα γραμμής, στιλ διαστάσεων, στρώσεις σχεδίασης, συντεταγμένες γεωμετρικών στοιχείων, στιλ διαγράμμισης, υλικό κ.α.).

Το αρχείο διαχωρίζει δύο βασικά στοιχεία:

Entities - Οντότητες (γεωμετρικά αντικείμενα): περιλαμβάνει γεωμετρικά χαρακτηριστικά. Οτιδήποτε έχει να κάνει σχέση με το πραγματικό σχέδιο (διαγραμμίσεις, διαστάσεις, γραμμή, κύκλος κ.α.).

Objects - Αντικείμενα (μη γεωμετρικά αντικείμενα): Στα αντικείμενα περιλαμβάνεται οτιδήποτε δεν αφορά γεωμετρικά χαρακτηριστικά (π.χ. λεξικά, υλικό, στρώσεις).

Κάθε ένα από τα δεδομένα που παρουσιάζονται στο αρχείο προσδιορίζονται από το **Group Code** (ετικέτα). Πρόκειται για έναν κωδικό, ο οποίος είναι ακέραιος αριθμός, και βρίσκεται πριν από κάθε δεδομένο (στην προηγούμενη γραμμή). Η ετικέτα δηλώνει τον τύπο και το είδος των δεδομένων που ακολουθεί. Στους πίνακες 1.3 και 1.4 παρουσιάζονται κάποια παραδείγματα Group Code

Group Code Range	**Τύπος δεδομένων**
0-9	String
10-39	Double precision point value
40-59	Double-precision floating-point value
60-79	16-bit integer
100	String (255-character maximum)

Πίνακας 1.3 - τύπος δεδομένων group code αρχείων .dxf

Group Code	**Είδος δεδομένων**
7	Στιλ κειμένου (fixed)
8	Όνομα στρώσης (fixed)
10	Συντεταγμένη κατά x του αρχικού σημείου γραμμής ή του κέντρου κύκλου
20	Συντεταγμένη κατά y του αρχικού σημείου γραμμής, ή του κέντρου κύκλου
30	Συντεταγμένη κατά z του αρχικού σημείου γραμμής, ή του κέντρου κύκλου
11-18	Συντεταγμένη άλλων σημείων κατά x
21-28	Συντεταγμένη άλλων σημείων κατά y
31-38	Συντεταγμένη άλλων σημείων κατά z
39	πάχος γραμμής (fixed)
50-58	Γωνία σε μοίρες
62	Χρώμα (fixed)
100	Όνομα κλάσης στην οποία ανήκει η οντότητα
999	Σχόλια

Πίνακας 1.4 - είδος δεδομένων group code αρχείων .dxf

Ορισμένες ετικέτες σε ένα στοιχείο entity υπάρχουν μόνιμα. Υπάρχουν όμως και προαιρετικές ετικέτες οι οποίες εμφανίζονται μόνο όταν έχουν διαφορετική τιμή από την καθορισμένη (default). Το τέλος κάθε οντότητας υποδηλώνεται με το μηδέν (0).

Στους παρακάτω πίνακες φαίνονται οι ετικέτες των γεωμετρικών στοιχείων που θα χρησιμοποιηθούν στην εφαρμογή (τόξο, κύκλος, γραμμή) και το είδος των δεδομένων που αντιπροσωπεύουν.

- Arc

Code Range	Είδος Δεδομένων
100	Κλάση κύκλου (AcDbCircle)
10	Το κέντρο του τόξου κατά τον άξονα X (OCS)
20	Το κέντρο του τόξου κατά τον άξονα Y (OCS)
30	Το κέντρο του τόξου κατά τον άξονα Z (OCS)
100	Κλάση τόξου (AcDbArc)
50	Γωνία μεταξύ άξονα X και ευθείας αρχικού σημείου - κέντρου τόξου
51	Γωνία μεταξύ άξονα X και ευθείας τελικού σημείου - κέντρου τόξου

Πίνακας 1.5 - group code τόξου

- Circle

Code Range	Τύπος δεδομένων
100	Κλάση κύκλου (AcDbCircle)
10	Το κέντρο του κύκλου κατά τον άξονα X (OCS)
20	Το κέντρο του κύκλου κατά τον άξονα Y (OCS)
40	Ακτίνα

Πίνακας 1.6 - group code κύκλου

- Line

Code Range	Τύπος δεδομένων
100	Κλάση ευθείας (AcDbLine)
10	Αρχικό σημείο στον άξονα X (wcs)
20	Αρχικό σημείο στον άξονα Y (wcs)
30	Αρχικό σημείο στον άξονα Z (wcs)
11	Τελικό σημείο στον άξονα X (wcs)
21	Τελικό σημείο στον άξονα Y (wcs)
31	Τελικό σημείο στον άξονα Z (wcs)

Πίνακας 1.7 - group code γραμμής

Σημείωση! Καθώς είναι πιθανόν σε επόμενες εκδόσεις να προστεθούν επιπλέον ετικέτες για τον προσδιορισμό επιπρόσθετων χαρακτηριστικών είναι προτιμότερο το πρόγραμμα ανάγνωσης του αρχείου για την εξόρυξη δεδομένων να αγνοεί οντότητες οι οποίες είναι προαιρετικές όπως και την σειρά με την οποία παρουσιάζονται οι ετικέτες.

Σημείωση! Τα τόξα στο αρχείο .dxf περιγράφονται πάντα αριστερόστροφα, ακόμα και αν κατά την σχεδίαση γίνουν δεξιόστροφα.

Στο Κεφάλαιο 5 (Εφαρμογή) που αφορά την υλοποίηση του CNC, έχει γίνει ανάπτυξη λογισμικού σε γλώσσα προγραμματισμού C++ που χρησιμοποιεί το αρχείο .dxf προκυμμένου να ανακτηθούν οι συντεταγμένες των γεωμετρικών στοιχείων του σχεδίου. Οι συντεταγμένες χρησιμοποιούνται για την δημιουργία αρχείο κειμένου .nc το οποίο περιέχει τον κώδικά G που περιγράφει το σχέδιο. Η παραπάνω διαδικασία υλοποιείται με την κλήση της συνάρτησης G_Code_Generator().

2. Hardware

2.1. Μετάδοση κίνησης

Το πρώτο στάδιο της σχεδίασης περιλαμβάνει την επιλογή του τρόπου μετατροπής της περιστροφικής κίνησης του μοτέρ σε γραμμική κίνηση. Υπάρχουν δύο βασικές σχεδιαστικές λύσεις για την επίτευξη αυτής της μετατροπής.

1. Μετάδοση κίνησης με ιμάντα χρονισμού: Η μετατροπή περιστροφικής σε γραμμική κίνηση είναι αποτέλεσμα της συνεργασίας τροχαλίας - ιμάντα.

2. Μετάδοση κίνησης με κοχλία: Η μετατροπή περιστροφικής σε γραμμική κίνηση είναι αποτέλεσμα της συνεργασίας κοχλίας - περικόχλιο.

Οι κυριότεροι παράγοντες που εξετάζονται για την αξιολόγηση και επιλογή του τρόπου μετάδοσης είναι:

Ταχύτητα: Υψηλές ταχύτητες θέτουν περιορισμούς στην καλή λειτουργία.

Ακρίβεια: Διακρίνονται δύο περιπτώσεις. Ακρίβεια στη μετατόπιση: αναφέρεται στην απόλυτη διαφορά μεταξύ της επιθυμητής (θεωρητικής μετατόπισης) και στην πραγματική. Επαναληψιμότητα: η σταθερότητα σε επαναλαμβανόμενες ίσες μετατοπίσεις.

Backlash (αναστροφή): Η συναρμογή μεταξύ των στοιχείων που συμμετέχουν στην μετάδοση της κίνησης έχει μια μικρή χάρη. Το κενό αυτό οδηγεί κάποιες φορές σε "πισωγύρισμα", την στιγμή που σταματάει η κίνηση.

Πολυπλοκότητα στον σχεδιασμό: περιλαμβάνει το σύνολο τον εξαρτημάτων που απαιτούνται και την μεταξύ τους συναρμολόγηση

Κλείδωμα θέσης: Η ικανότητα να διατηρείται η θέση μετά το τέλος της κίνησης χωρίς επιβοήθεια.

Μέγιστη μετατόπιση: Το μέγιστο μήκος της διαδρομής που είναι εφικτό να καλυφθεί.

Επιτάχυνση: Μεγάλη επιτάχυνση θέτει περιορισμούς στην καλή λειτουργία

Φορτίο: Το συνολικό φορτίο (βάρος + δυνάμεις) το οποίο συμμετέχει στην κίνηση.

Στο παρακάτω πίνακα γίνεται σύγκριση των παραπάνω χαρακτηριστικών της κίνησης με ιμάντα έναντι της κίνησης με κοχλία.

Σύγκριση μηχανισμών κίνησης Ιμάντα VS Κοχλία		
Χαρακτηριστικό	Ιμάντας	Κοχλίας
Ταχύτητα	+	-
Ακρίβεια	-	+
(Αναστροφή) Backlash	-	+
Κόστος	+	-
Πολυπλοκότητα	+	-
Σταθεροποίηση θέσης (locking)	-	+
Μήκος διαδρομής	+	-
Επιτάχυνση	-	+
Φορτίο	-	+

Πίνακας 2.1 - Belt Vs Screw driven linear actuator

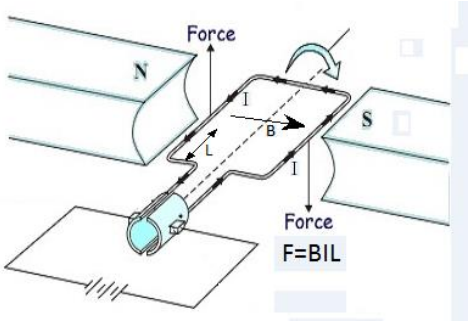
Ανάλογα με τις απαιτήσεις της κάθε της εφαρμογής γίνεται η επιλογή του τρόπου που θα χρησιμοποιηθεί. Εφαρμογές όπου απαιτείται μεγάλη ακρίβεια ή/και κατακόρυφη κίνηση είναι καταλληλότερη η χρήση κοχλία. Ενώ ιμάντας χρονισμού είναι καταλληλότερος σε εφαρμογές που χρειάζεται μεγάλη ταχύτητα, μεγάλο μήκος διαδρομής και απλότητα στον σχεδιασμό.

2.2. Κινητήρες

Επόμενη σημαντική απόφαση είναι ο τύπος μοτέρ που θα χρησιμοποιηθεί. Υπάρχουν διάφοροι τύποι κινητήρων που διαφέρουν μεταξύ τους όσο αναφορά την αρχή λειτουργίας τους και τα τεχνικά χαρακτηριστικά. Ο τύπος που είναι καταλληλότερος εξαρτάται από την εφαρμογή. Παρακάτω γίνεται μια αναφορά στους τύπους μοτέρ που θα χρησιμοποιηθούν στην εφαρμογή και στις αρχές λειτουργίας τους.

2.2.1. DC κινητήρας

Ο κινητήρας συνεχούς ρεύματος είναι ίσως ο πιο απλός τύπος ηλεκτρικού κινητήρα. Μπορεί να μετατρέψει ηλεκτρική ενέργεια σε μηχανική και αντίστροφα. Η λειτουργία του είναι απόρροια του φαινομένου της ηλεκτρομαγνητικής επαγωγής (Στον αγωγό που διαρρέεται από ρεύμα και βρίσκεται μέσα σε μαγνητικό πεδίο τότε ασκείται πάνω του δύναμη).



Εικόνα 2.1 - Αρχή λειτουργίας DC κινητήρα

Αν εφαρμόσουμε τάση στα άκρα του κινητήρα, αυτός περιστρέφεται με την μέγιστη ταχύτητα. Για να περιστραφεί κατά την αντίθετη φορά αρκεί να αλλάξει η πολικότητα στα άκρα.

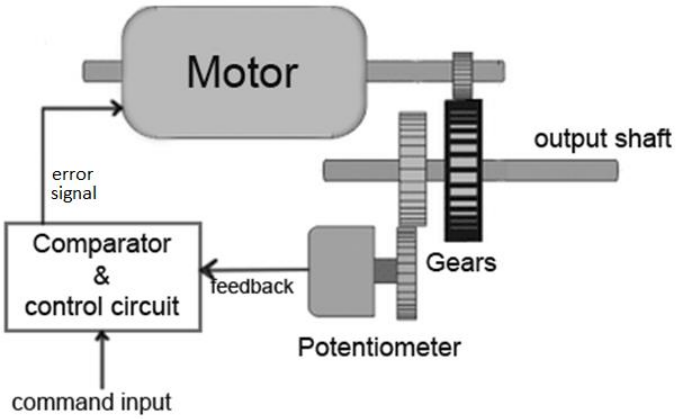
Ένας τρόπος για την ρύθμιση της ταχύτητας είναι εναλλαγή του σήματος on/off. Η ταχύτητα είναι ανάλογη του ποσοστού παρουσία/απουσίας σήματος. Αν για παράδειγμα δοθεί παλμός ο οποίος την μισή χρονική διάρκεια είναι on και την άλλη μισή off τότε ο κινητήρας θα περιστραφεί με το 50% της μέγιστης ταχύτητας.

Αυτού του είδους ο κινητήρας μπορεί να χρησιμοποιηθεί για την λειτουργία ανεμιστήρα στο σύστημα ψύξης εφόσον χρειάζεται. Στην συγκεκριμένη εφαρμογή δεν θα χρησιμοποιηθεί κάποιο σύστημα ψύξης.

2.2.2. Σέρβο κινητήρας

Ο σέρβο-κινητήρας είναι κατάλληλος για εφαρμογές που απαιτούν έλεγχο θέσης, ταχύτητας και μεγάλη ροπή. Χρησιμοποιείται ένα σύστημα κλειστού βρόγχου όπου ένα σήμα ανά-τροφοδοσίας συγκρίνεται με ένα σήμα ελέγχου ώστε να παραχθεί το σήμα εισόδου στον κινητήρα.

Αποτελείται από έναν DC μοτέρ, ένα ποτενσιόμετρο, μειωτήρα και το κύκλωμα ελέγχου. Ο ρόλος του μειωτήρα είναι η μείωση των στροφών και η αύξηση της ροπής. Στην αρχική θέση , η θέση του ποτενσιόμετρου είναι τέτοια ώστε στην έξοδο του να μην υπάρχει σήμα. Όταν δοθεί στην είσοδο σήμα ο ελεγκτής συγκρίνει το σήμα εισόδου με το σήμα ανά-τροφοδοσίας (έξοδος του ποτενσιόμετρου). Εφόσον η διαφορά των δύο σημάτων δεν είναι μηδέν ο ελεγκτής παράγει σήμα (error signal) ώστε να περιστραφεί ο κινητήρας. Ο άξονας του μοτέρ είναι συνδεδεμένος με το ποτενσιόμετρο και καθώς περιστρέφεται αλλάζει το σήμα εξόδου στο ποτενσιόμετρο. Σε κάποια θέση η έξοδος στο ποτενσιόμετρο γίνεται ίση με το σήμα εισόδου – ελέγχου και το σήμα στην είσοδο του μοτέρ μηδενίζεται και σταματάει η περιστροφή του μοτέρ

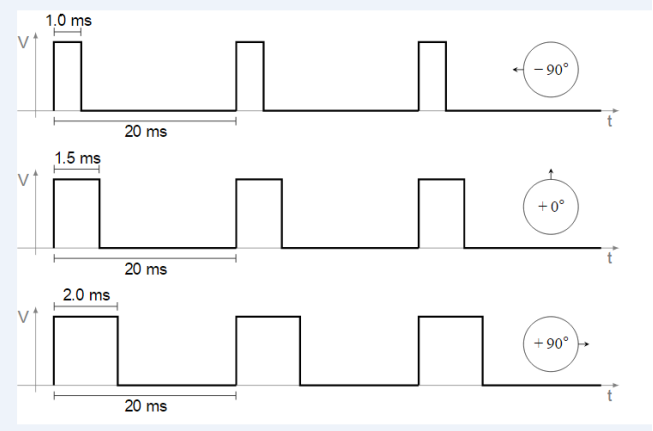


Εικόνα 2.2 - Έλεγχος θέσης Servo κινητήρα

Ο σέρβο-κινητήρας που θα χρησιμοποιήσουμε έχει τρεις εισόδους. Δύο για την τροφοδοσία (θετικό (+5V), γείωση) και μία είσοδο για το σήμα ελέγχου το οποίο θα πρέπει να σταλεί από τον μικροελεγκτή.

Μπορεί να περιστραφεί κατά 90 μοίρες σε σχέση με την θέση ηρεμίας (90^o 0^o - 90^o). Οπότε το εύρος της περιστροφής είναι 180. Ο έλεγχος του σήματος στο μοτέρ γίνεται κάθε 20 milliseconds (ms) και ή διάρκεια του θετικού παλμού καθορίζει το πόσες μοίρες θα περιστραφεί ο άξονας. Παλμός διάρκειας 1.5ms οδηγεί τον άξονα στις 0^o ,1ms οδηγεί τον άξονα στις - 90^o και 2ms οδηγεί τον άξονα στις 90^o

Το σήμα ελέγχου δημιουργείται με διαμόρφωση πλάτους παλμού PWM (Pulse Width Modulation). Το πως ο μικροελεγκτής αναπαράγει το σήμα ελέγχου μέσω PWM θα εξεταστεί μετέπειτα.



Εικόνα 2.3 - Παλμοί για περιστροφή servo κινητήρα σε συγκεκριμένη γωνία

Ο τύπος του σέρβο-κινητήρας που περιγράφτηκε παραπάνω θα χρησιμοποιηθεί στην περίπτωση που στην κεφαλή προσαρμοστεί κατάλληλο σύστημα με ακίδα γραφής (π.χ. μολύβι, μαρκαδόρος). Όταν ο κώδικας δώσει εντολή για κίνηση χωρίς κατεργασία (G00) το μοτέρ θα οδηγείται στην θέση 0^o (πάνω) ενώ όταν θα πρέπει να γίνει κίνηση με κατεργασία (G01, G02, G03) το μοτέρ θα οδηγείται στην θέση 90^o (κάτω).

2.2.3. Βηματικός κινητήρας

Πρόκειται για ηλεκτροκινητήρα που βασικό χαρακτηριστικό του είναι ότι περιστρέφεται διαγράφοντας διαδοχικά διακριτά βήματα (step angle) σε κάθε παλμό που δέχεται. Η λειτουργία αυτή οφείλεται στην κατασκευή του και δίνει την δυνατότητα ελέγχου της θέσης μετρώντας μόνο τους παλμούς - βήματα (χωρίς την ύπαρξη επιπλέον κυκλώματος ελέγχου) και τον έλεγχο της ταχύτητας ρυθμίζοντας την συχνότητα εναλλαγής των παλμών.

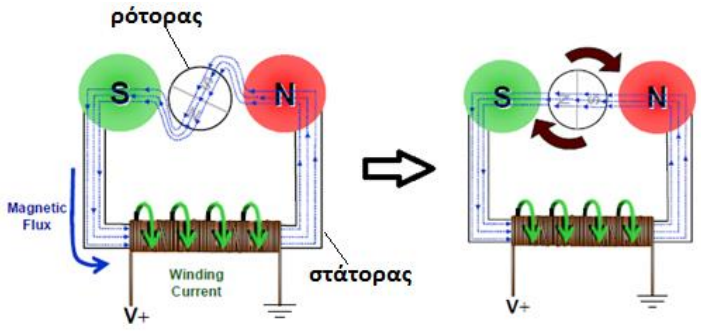
Αποτελείται από το στατικό μέρος (στάτορα) όπου περιέχει την περιέλιξη του σύρματος και το κινητό μέρος (ρότορα). Ανάλογα με την τεχνολογία περιέλιξης του σύρματος στον στάτορα οι βηματικοι κινητήρες διακρίνονται σε:

διπολικούς: Το σύρμα περιπλέκεται ακολουθώντας μια συγκεκριμένη φορά σε όλο το μήκος του στάτορα. Για να αλλάξει η πολικότητα του μαγνητικού πεδίου που δημιουργείται πρέπει να αλλάξει η φορά το ρεύματος. Η τεχνική αυτή προσφέρει μεγαλύτερη ροπή αλλά απαιτεί περίπλοκο σύστημα ελέγχου.

μονοπολικούς: Η φορά περιέλιξης στον στάτορα διαχωρίζεται. Το μισό σύρμα περιτυλίγεται έχοντας μια συγκεκριμένη φορά ενώ στο υπόλοιπο μισό η φορά αντιστρέφεται. Για να αλλάξει η πολικότητα του πεδίου αρκεί να αλλάξει η κατεύθυνση του ρεύματος, Το κύκλωμα για τον έλεγχο είναι απλούστερο , παρόλο αυτά η ροπή μειώνεται στο μισό καθώς χρησιμοποιείται το μισό τύλιγμα κάθε φορά.

Ένας άλλος διαχωρισμός έχει να κάνει με το είδος του ρότορα σε μόνιμου μαγνήτη, μεταβλητής αντίδρασης και υβριδικούς

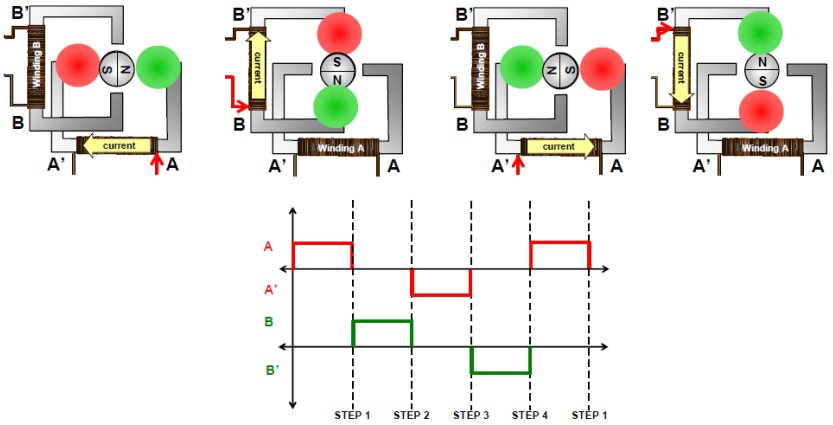
Ο στάτορας περιλαμβάνει τουλάχιστον δύο ανεξάρτητες περιελίξεις που ονομάζονται φάσεις. Εφαρμόζοντας τάση σε κάποιο από τα τυλίγματα του στάτορα αναπτύσσεται μαγνητικό πεδίο ως αποτέλεσμα του ρεύματος που διαρρέει τον αγωγό (η φορά της μαγνητικής ροής, βρίσκεται με τον κανόνα του δεξιού χεριού).Ο ρότορας αναγκάζεται να προσαρμόσει την θέση του ανάλογα (θέση ισορροπίας) ώστε να ευθυγραμμιστεί με την φορά του πεδίου της μαγνητικής ροής (η μαγνητική ροή ψάχνει να βρει μονοπάτι με την μικρότερη αντίσταση).



Εικόνα 2.4 - Αρχή λειτουργίας βηματικού κινητήρα

Μεταφέροντας την τάση ακολουθιακά στο γειτονικό τύλιγμα και αλλάζοντας διαδοχικά την πολικότητα αναγκάζεται ο ρότορας να περιστραφεί. Υπάρχουν διάφορες τεχνικές για τον τρόπο με τον οποίο γίνεται η εναλλαγή των φάσεων

Wave Step : Κατά την λειτουργία αυτή είναι ενεργοποιημένη μια φάση κάθε φορά και η ελάχιστη μετακίνηση του ρότορα αντιστοιχεί σε ένα βήμα.



Εικόνα 2.5 - Wave mode

Στην εικόνα 2.5

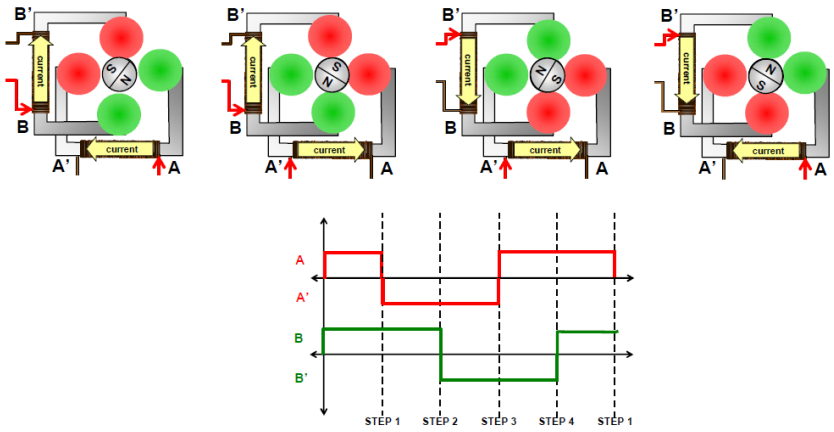
βήμα 1:αρχικά ενεργοποιείται η φάση A και ο ρότορας έρχεται στην θέση ισορροπίας.

βήμα 2: ενεργοποιείται η φάση B και απενεργοποιείται η φάση A με αποτέλεσμα ο ρότορας να περιστραφεί κατά ένα step (90^o) δεξιόστροφα.

βήμα 3: ενεργοποιείται η φάση A' (αντιστρέφεται η πολικότητα στο τύλιγμα A). Και απενεργοποιείται η B

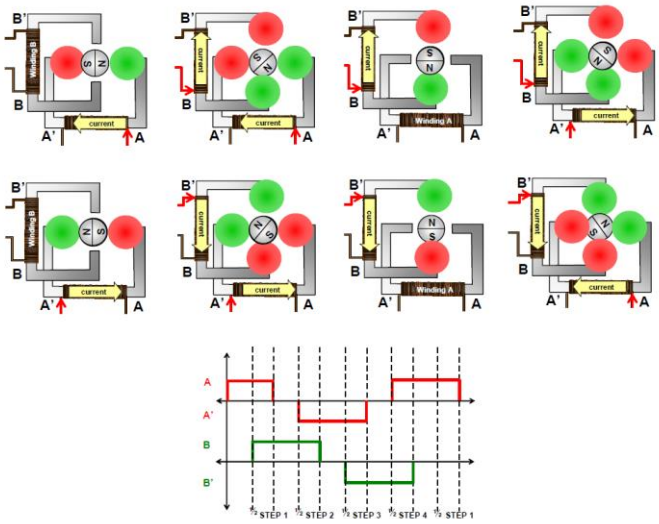
βήμα 4: ενεργοποιείται η φάση B' (αντιστρέφεται η πολικότητα στο τύλιγμα B). Και απενεργοποιείται η A'

Full Step: Σε κάθε βήμα είναι ταυτόχρονα ενεργοποιημένες δύο φάσεις ταυτόχρονα. Η ελάχιστη μετακίνηση είναι ένα βήμα: Η θέση ισορροποίας του ρότορα είναι ανάμεσα στις δύο φάσεις. Το πλεονέκτημα αυτής της μεθόδου έναντι της προηγούμενης είναι η αύξηση της ροπής.



Εικόνα 2.6 - Full Step

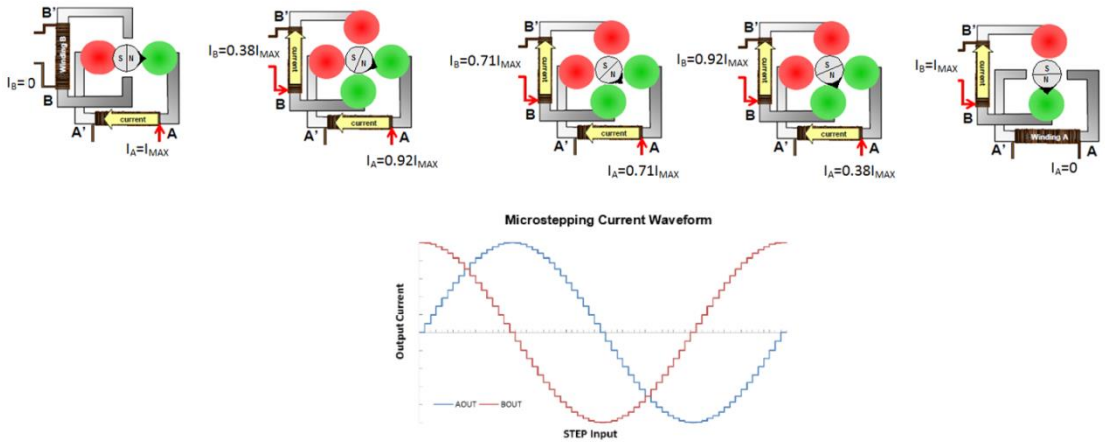
Half Step: Είναι συνδυασμός των δύο παραπάνω τρόπων. Κάθε δεύτερο βήμα μια φάση είναι ενεργοποιημένη (wave step) ενώ στις ενδιάμεσες θέσεις δύο φάσεις είναι ενεργοποιημένες (full step). Η ελάχιστη μετακίνηση στην περίπτωση αυτή είναι μισό βήμα. Με τον τρόπο αυτό αυξάνεται η ανάλυση αλλά μειώνεται η ροπή.



Εικόνα 2.7 - Half mode

Micro-step:

Είναι ουσιαστικά μια περεταίρω διαίρεση του half-step σε ακόμα μικρότερα step και μπορεί να οδηγήσει σε μεγαλύτερη αύξηση αλλά μικρότερη ροπή. Αυτό επιτυγχάνεται ελέγχοντας την ένταση του ρεύματος που διέρχεται από το κάθε τύλιγμα. Κάθε στιγμή είναι ενεργοποιημένες ταυτόχρονα δύο διαδοχικές φάσεις. Η ένταση του ρεύματος α-κολουθεί μια ημιτονική μεταβολή με αποτέλεσμα καθώς μειώνεται ή ένταση στο ένα τύλιγμα να αυξάνεται αντίστοιχα στο επόμενο. Με την τεχνική αυτή μπορεί το βασικό βήμα του βηματικού κινητήρα υποδιαιρεθεί και να πάρει πολύ μικρότερες τιμές π.χ. 1/32.



Εικόνα 2.8 - Micro-step

Εκτός των παραπάνω, ένας επιπλέον τρόπος για να μειωθεί το βήμα είναι να αυξηθεί ο αριθμός των πόλων του ρότορα και του στάτορα που περιλαμβάνει ή κάθε φάση.

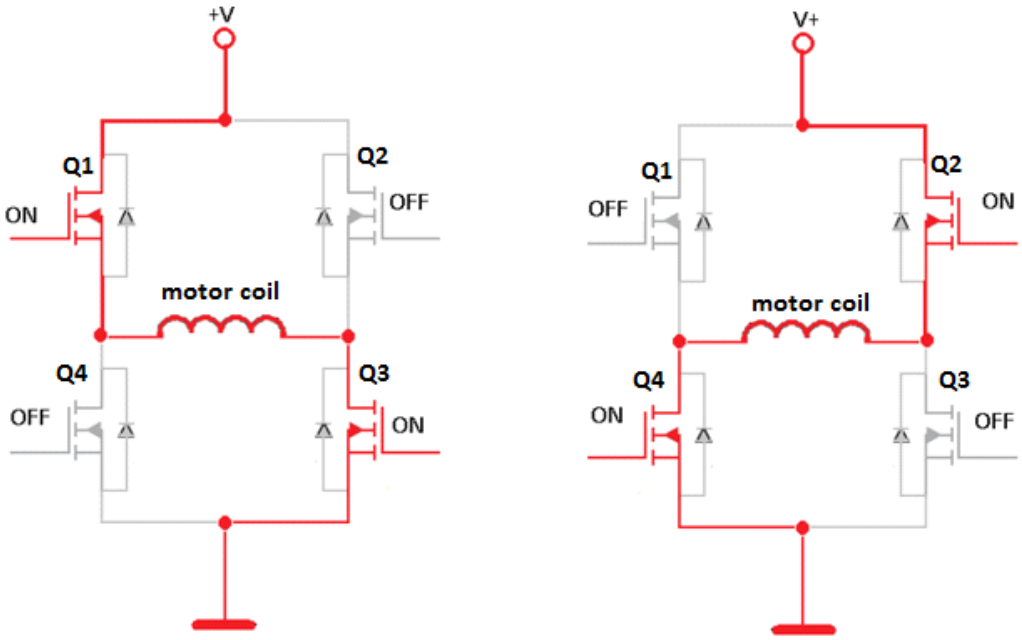
Ο βηματικός κινητήρας θα χρησιμοποιηθεί για την κίνηση της κεφαλής στους δύο άξονες. Βασικό χαρακτηριστικό για την επιλογή του κινητήρα είναι η ροπή που μπορεί να αποδώσει. Για την επιλογή του μοτέρ έγινε υπολογισμός της απαιτούμενης ροπής.

2.3 Έλεγχος Βηματικού Διπολικού Κινητήρα - Γέφυρα Η

Η γέφυρα Η αποτελεί μια ηλεκτρονική διάταξη με τρανζίστορ κατάλληλη για τον έλεγχο της φοράς του ρεύματος στο φορτίο (στην συγκεκριμένη περίπτωση το τύλιγμα του στάτορα του βηματικού κινητήρα).

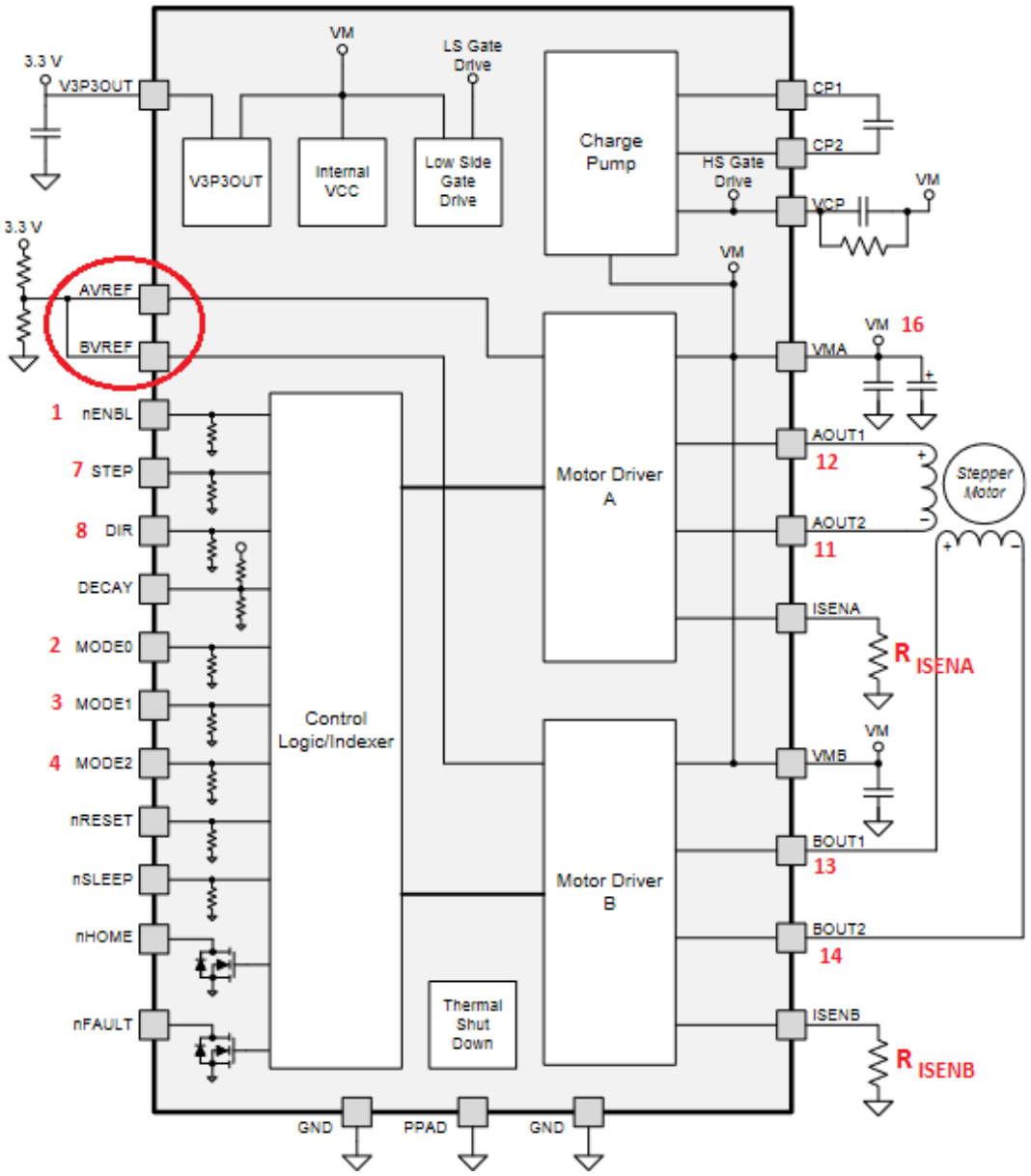
Τα Q1 και Q2 είναι συνδεδεμένα με την τροφοδοσία ενώ τα Q3 και Q4 οδηγούν στη γείωση. Όταν εφαρμόζεται θετική τάση στην βάση του τρανζίστορ τότε η δίοδος συλλέκτη - εκπομπού άγει. Με τα Q1 και Q3 σε κατάσταση ON και τα Q2 και Q4 σε κατάσταση OFF, ρεύμα διαρρέει το τύλιγμα κατά την θετική φορά (+). Ενώ με Q2 και Q4 είναι σε κατάσταση ON και τα Q1 και Q3 σε κατάσταση OFF, ρεύμα διαρρέει το τύλιγμα κατά την αρνητική φορά (-).

Η εναλλαγή των τρανζίστορ σε on/off πρέπει να γίνει με μεγάλη ακρίβεια. Αν κάποια στιγμή, τύχη να είναι σε κατάσταση on δύο τρανζίστορ στην ίδια μεριά της γέφυρας (π.χ. Q1 και Q4) θα καταστραφεί το κύκλωμα.



Εικόνα 2.9 - Αρχή λειτουργίας γέφυρας Η

Το ολοκληρωμένο **DRV8825** (Texas Instrument) είναι κατάλληλο για έλεγχο διπολικών βηματικών κινητήρων. Περιλαμβάνει 2 γέφυρες Η που υλοποιούνται με τρανζίστορ NMOS, αισθητήρα ρεύματος, κύκλωμα ελέγχου και ευρετήριο για λειτουργία micro-step. Έχει ικανότητα να παρέχει στην έξοδο μέχρι και 2.5 Α ενώ το εύρος της τάση τροφοδοσίας κυμαίνεται από 8.2-45 V.



Εικόνα 2.10 - DRV8825 block diagram

Ρύθμιση ρεύματος:

Το ρεύμα στο τύλιγμα του μοτέρ ρυθμίζεται μέσω ενός σήματος πλάτους διαμόρφωσης παλμού (PWM) , συγκεκριμένης συχνότητας. Αρχικά μόλις ενεργοποιηθεί η γέφυρα Η το ρεύμα στο τύλιγμα αρχίζει σταδιακά να αυξάνεται. Ο ρυθμός αύξησης εξαρτάται από την εφαρμοζόμενη τάση και από την αυτεπαγωγή του σύρματος. Όταν η ένταση φθάσει το κατώφλι του παλμού η γέφυρα απενεργοποιείται. Η διαδικασία επαναλαμβάνεται στο επόμενο σήμα (PWM).

Για την αναπαραγωγή του σήματος (PWM) χρησιμοποιείται ένα κύκλωμα συγκριτή. Το κύκλωμα συγκρίνει το πενταπλάσιο της τάσης στα άκρα μιας αντίστασης ευαισθησίας ρεύματος Risense με μία τάση αναφοράς Vref.

Σε λειτουργία full-scale το κατώφλι υπολογίζεται από την σχέση

$$I_{CHOP} = \frac{V_{REF}}{5xR_{ISEN}} \quad (2.2)$$

Όπου I_{CHOP}=Το μέγιστο επιτρεπτό ρεύμα στο τύλιγμα του μοτέρ. Ανάλογα με τις ανάγκες του μοτέρ σε ρεύμα θα πρέπει να προσαρμόσουμε την V_{REF} και R_{ISEN} ώστε να έχουμε το σωστό I_{CHOP}.

Το block Control/Logic Indexer αποτελεί ένα κύκλωμα το οποίο περιλαμβάνει κατά κάποιο τρόπο, σε μορφή πίνακα (look-up table), την κατανομή του ρεύματος στα τυλίγματα του διπολικού κινητήρα. Μέσω των ακροδεκτών MODE0,MODE1 και MODE2 γίνεται η επιλογή του λειτουργίας σύμφωνα με τον παρακάτω πίνακα.

MODE2(M2)	MODE1(M1)	MODE0(M0)	STEP MODE
0	0	0	full step (two-phase)
0	0	1	1/2 step (1-2 phase)
0	1	0	1/4 step
0	1	1	8 microstep/step
1	0	0	16 microstep/step
1	0	1	32 microstep/step
1	1	0	32 microstep/step
1	1	1	32 microstep/step

Πίνακας 2.2 - step modes

Ο παρακάτω πίνακας δείχνει την κατανομή του ρεύματος σε κάθε τύλιγμα ανάλογα με το mode που έχει επιλεγεί. Σε κάθε θετικό παλμό που δέχεται ο ακροδέκτης STEP ένας δείκτης μετατοπίζεται στην επόμενη κατάσταση του πίνακα. Ο πίνακας συνεχίζεται μέχρι τις 357^{o}.

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
1	1	1	1	1		100%	0%	0
2						100%	5%	3
3	2					100%	10%	6
4						99%	15%	8
5	3	2				98%	20%	11
6						97%	24%	14
7	4					96%	29%	17
8						94%	34%	20
9	5	3	2			92%	38%	23
10						90%	43%	25
11	6					88%	47%	28
12						86%	51%	31
13	7	4				83%	56%	34
14						80%	60%	37
15	8					77%	63%	39
16						74%	67%	42
17	9	5	3	2	1	71%	71%	45

Πίνακας 2.3 - Micro-step indexer

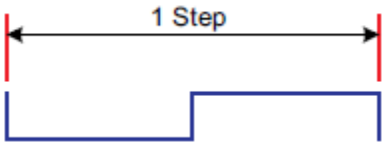
Το ολοκληρωμένο είναι προσαρτημένο σε μια μικρή πλακέτα. Η πλακέτα παρέχει τις εξής διεπαφές.

Εικόνα 2.11 - πλακέτα με ενσωματωμένο το ολοκληρωμένο DRV8825

1(EN): Ο driver είναι ενεργοποιημένος όταν ο ακροδέκτης (EN) είναι σε λογικό "0" (default) και απενεργοποιείται όταν πάρει λογικό "1".

2(M0),3(M1),4(M2): Ο συνδυασμός αυτών τον ακροδεκτών καθορίζει την συμπεριφορά του driver σύμφωνα με τον πίνακα 2.2:

7(STEP): Ο κινητήρας περιστρέφεται κατά 1 micro-step όταν παραχθεί ένας πλήρης παλμός σε αυτόν τον ακροδέκτη (1- 0)



Η συχνότητα με την οποία αναπαράγεται ο παλμός είναι η ταχύτητα και ο αριθμός των παλμών είναι η μετατόπιση.

8(DIR): Η τιμή στον ακροδέκτη αυτόν ρυθμίζει την φορά περιστροφή του κινητήρα

0: δεξιόστροφα

1: αριστερόστροφα

11(A2),12(A1),13(B1),14(B2): Για την σύνδεση του τυλίγματος των φάσεων του μοτέρ σύμφωνα με το datasheet του βηματικού κινητήρα.

15(GND MOT): Η γείωση της τροφοδοσίας

16(VMOT): Τροφοδοσία που απαιτείται για την λειτουργία του μοτέρ

2.4. Atmega 328p[2]

Ο μικροελεγκτής Atmega 328p (Atmel) χρησιμοποιεί αρχιτεκτονική Harvard (πρόγραμμα και δεδομένα έχουν ξεχωριστές μνήμες και διαύλους). Το set εντολών (131) είναι δομημένο σύμφωνα με την αρχιτεκτονική RISC (σε κάθε χτύπο ρολογιού εκτελείται μια εντολή). Κατά την διάρκεια εκτέλεσης μιας εντολής η επόμενη εντολή έχει διαβαστεί από την μνήμη προγράμματος και αναμένει για εκτέλεση.

Έχει 32 καταχωρητές γενικού σκοπού των 8 bit οι οποίοι είναι προσβάσιμοι σε έναν κύκλο ρολογιού και επίσης είναι απευθείας προσβάσιμοι από την λογική αριθμητική μονάδα. Μια τυπική λειτουργία της λογική αριθμητικής μονάδας που εκτελείται σε ένα χτύπο είναι: προσκόμιση δύο τελεστών από καταχωρητές, εκτέλεση πράξης, αποθήκευση του αποτελέσματος σε καταχωρητή. Έξι από τους καταχωρητές αυτούς μπορούν να χρησιμοποιηθούν ως τρεις καταχωρητές των 16-bit

Features	**ATmega328/P**
Pin Count	28/32
Flash (Bytes)	32K
SRAM (Bytes)	2K
EEPROM (Bytes)	1K
General Purpose I/O Lines	23
SPI	28/32
TWI (I2C)	1K
USART	1K
ADC	10-bit 15kSPS
ADC Channels	8
8-bit Timer/Counters	2
16-bit Timer/Counters	10-bit 15kSPS

Πίνακας 2.4 - χαρακτηριστικά m328p

2.4.1 Timers - Counters

Ο m328p έχει τρείς μετρητές. Δύο των 8 bit (timer0 και timer2) και έναν των 16 bit (timer1). Ανάλογα με το mode λειτουργίας στο οποίο έχει προγραμματιστεί ο timer, μπορεί να αυξάνεται, μειώνεται ή να μηδενίζεται. Ο timer αλλάζει τιμή σε καθορισμένο αριθμό χτύπων ρολογιού ανάλογα με το ορισμό ενός prescaller. Επιπλέον υπάρχει η δυνατότητα σύγκρισης της τιμής του counter με την τιμή που έχει οριστεί σε έναν καταχωρητή σύγκρισης.

Όταν ο timer υπερχειλίσει ή όταν ταυτιστεί με την τιμή του καταχωρητή σύγκρισης, τότε εκτελείται μια ενέργεια ανάλογα με το set up του timer. Η πιο απλή λειτουργία είναι να πάρει ο counter την τιμή εκκίνησης και να ξεκινήσει από την αρχή το μέτρημα. Πιο σύνθετα θα μπορούσε να προκαλέσει ένα interrupt και την εκτέλεση μιας ρουτίνας.

Για την συγκεκριμένη εφαρμογή θα εκμεταλλευτούμε την ιδιότητα που έχει ο timer να κάνει toggle έναν συγκεκριμένο ακροδέκτη όταν ταυτιστεί με την τιμή που υπάρχει στο καταχωρητή σύγκρισης.

3. Universal Asynchronous Receiver – Transmitter (UART)

Έχοντας καταφέρει από το σχέδιο να δημιουργήσουμε τον κώδικα G που περιέχει τις συντεταγμένες, και έχοντας υλοποιήσει το hardware, το επόμενο στάδιο είναι η επικοινωνία μεταξύ του προγράμματος εφαρμογής και του μικροελεγκτή. Ο m328p υποστηρίζει τρεις διαφορετικά πρωτόκολλα για επικοινωνία με το εξωτερικό περιβάλλον.

-**I2C** (Inter Integrated Circuit): Χρησιμοποιείται ευρέως σε εφαρμογές όπου χρειάζεται επικοινωνία διαφόρων αισθητηρίων με μικροελεγκτή.

-**SPI** (Serial Peripheral Interface): Χρησιμοποιείται όταν μια συσκευή (master) π.χ. μικροελεγκτής πρέπει να επικοινωνήσει με άλλες περιφερειακές συσκευές (slaves) όπως κάρτες sd, άλλους μικροελεγκτές και άλλα.

-**USART** (Universal Synchronous Asynchronous Receiver Transmitter): Χρησιμοποιείται συνήθως για σειριακή μεταφορά δεδομένων μεταξύ ενσωματωμένων συστημάτων, η μεταξύ PC και μικροελεγκτή. Η διαφορά μεταξύ Synchronous και Asynchronous έχει να κάνει με τον τρόπο που συντονίζεται ο πομπός με τον δέκτη.

Asynchronous: Ένας παλμός (baud rate) δημιουργείται εσωτερικά στον hardware του πομπού. Ο παλμός συγχρονίζεται με την ροή δεδομένων. Απαιτείται, bit εκκίνησης που υποδηλώνει την αρχή ενός πακέτου δεδομένων και bit τερματισμού. Προϋπόθεση για σωστή επικοινωνία είναι το ρολόι στον δέκτη να έχει συντονιστεί ώστε να παράγει παλμό ίδιας συχνότητας.

Synchronous: Ο πομπός δημιουργεί ένα ρυθμό μετάδοσης. Ο δέκτης μπορεί να ανακτήσει τον ρυθμό μετάδοσης (baud rate) από το πακέτο (δεν χρειάζεται να τον ξέρει από πριν). Δεν χρειάζεται bit εκκίνησης και τερματισμού.

Για τις ανάγκες της εφαρμογής θα χρησιμοποιηθεί UART (Universal Asynchronous Receiver Transmitter) επικοινωνία.

3.1. UART επικοινωνία στον m328p [2]

Για την υλοποίηση χρησιμοποιούνται δύο γραμμές, μια για την μετάδοση (Transmitter - PD1/TXD) και μια για την λήψη των δεδομένων (Receiver - PD0/RXD). Συνεπώς αυτά τα δύο pin είναι δεσμευμένα από την UART επικοινωνία.

Τα δεδομένα που πρόκειται να μεταφερθούν οδηγούνται στην μονάδα UART του πομπού μέσω διαύλου παράλληλα. Από εκεί τα δεδομένα μεταδίνονται σειριακά bit προς bit μέσω της γραμμής TXD στην μονάδα UART του δέκτη. Ο δέκτης με την σειρά του μετατρέπει τα σειριακά δεδομένα σε παράλληλα.

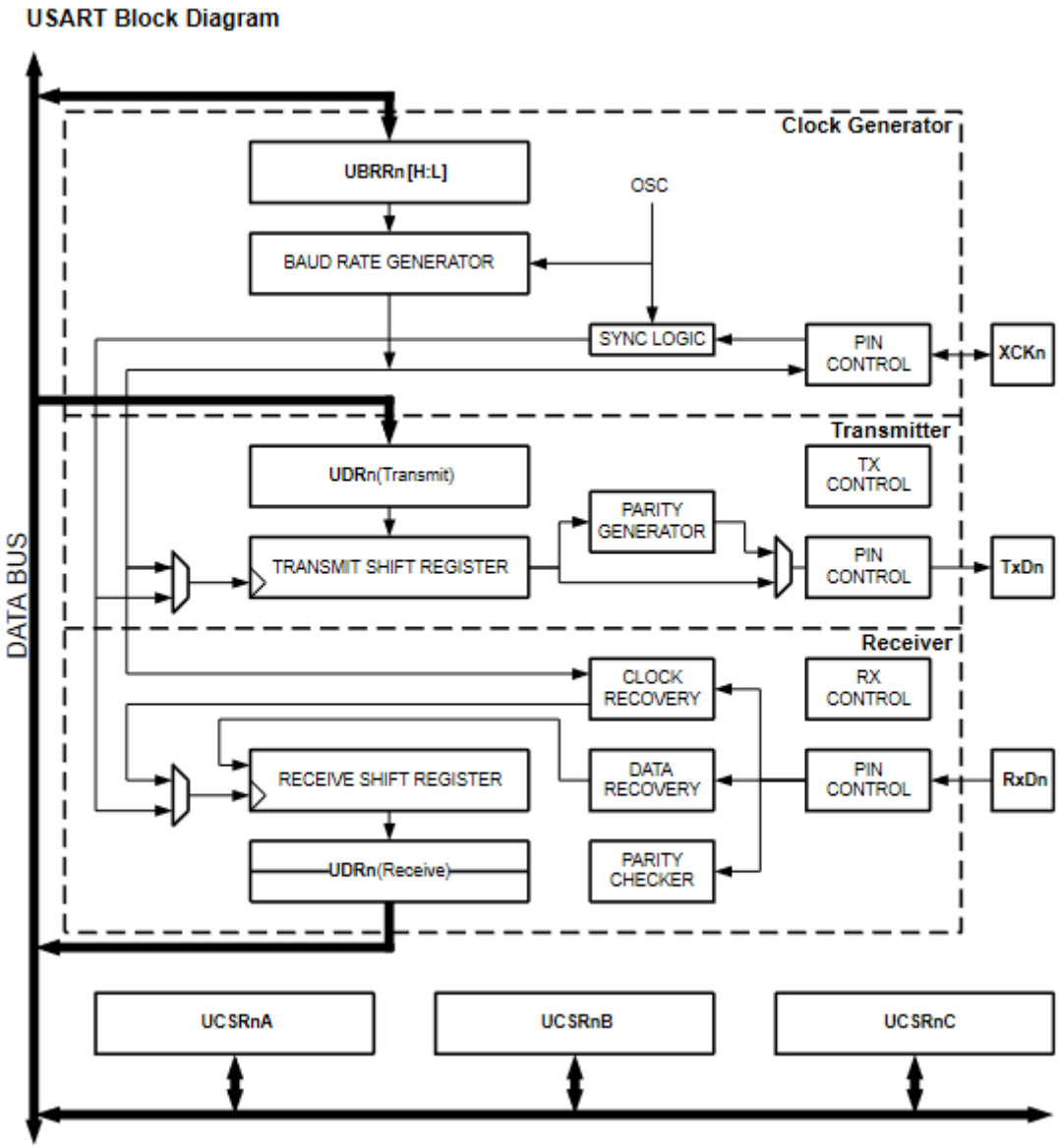
Στην περίπτωση της εφαρμογής τα δεδομένα "ταξιδεύουν" στο φυσικό μέσο, στα επίπεδα της TTL τάσης. Οπότε δεν χρειάζεται να γίνει μετάβαση της τάσης για την δρομολόγηση των δεδομένων στο φυσικό μέσο και το UART hardware δεν περιλαμβάνει κύκλωμα για μετατροπή τάσης. Στην εικόνα 3.7 φαίνεται το block διάγραμμα του hardware του μικροελεγκτή για την υλοποίηση USART επικοινωνίας. Το hardware διαχωρίζεται σε τρία κύρια τμήματα.

1. Ρολόι: Είναι ή γεννήτρια ρολογιού για τον συντονισμό πομπού και δέκτη. Μπορεί να υποστηρίξει τέσσερα διαφορετικά mode λειτουργίας (Normal asynchronous, Double Speed asynchronous, Master synchronous και Slave synchronous mode).

Η μεταφορά των δεδομένων γίνεται ασύγχρονα (χωρίς την ύπαρξη ρολογιού για τον συντονισμό μεταξύ πομπού-δέκτη). Για τον λόγο αυτό είναι απαραίτητη προϋπόθεση ο ορισμός ενός κοινού ρυθμού μετάδοσης δεδομένων (baud rate).Ο πομπός στέλνει μια ροή από bit με καθορισμένο ρυθμό χρησιμοποιώντας το εσωτερικό του ρολόι και ο δέκτης χρησιμοποιεί το δικό του εσωτερικό ρολόι το οποίο είναι ρυθμισμένο να λειτουργεί με τον ίδιο ρυθμό για να λάβει τα δεδομένα. Σφάλμα μεγαλύτερο από 2% στον ρυθμό μετάδοσης μεταξύ πομπού και δέκτη μπορεί να δημιουργήσει ασάφεια στα δεδομένα.

Σε λειτουργία Normal asynchronous ο USART Baud Rate Register (UBRRn) χρησιμοποιείται για την δημιουργία του ρυθμού μετάδοσης δεδομένων «baud rate». Η τιμή του UBRRn φορτώνεται σε έναν μετρητή (ο οποίος λειτουργεί με την συχνότητα του συστήματος - f_{OSC}). Ο μετρητής μειώνει την τιμή του σε καθορισμένο αριθμό χτύπων ρολογιού (ανάλογα με τις ρυθμίσεις που έχουν γίνει). Κάθε φορά που ο μετρητής μηδενίζεται δημιουργεί ένα σήμα ρολογιού και ξαναφορτώνει την τιμή του UBRRn. Η σχέση μεταξύ του UBRRn και του baud rate (BAUD) είναι:

$$UBRR_n = \frac{fosc}{16BAUD} - 1 \quad (3.1)$$



Εικόνα 3.1 – USART Block Diagram m328p

2. Πομπός: Αποτελείται από έναν buffer για την προσωρινή αποθήκευση των δεδομένων (UDRN0), έναν καταχωρητή σειριακής μετατόπισης (Shift Register), Γεννήτρια ισοτιμίας (Parity Generator) και Μονάδα ελέγχου (Tx Control) για τον χειρισμό διάφορων μορφών πλαισίων.

Ο πομπός ενεργοποιείται όταν το Transmit Enable (TXEN)bit στον UCSRnB καταχωρητή πάρει τιμή '1'. Όταν ο πομπός είναι ενεργοποιημένος το TxDn pin λειτουργεί ως σειριακή έξοδος. Τα προς μεταφορά δεδομένα αποθηκεύονται αρχικά στον buffer και παραμένουν εκεί όσο ο shift register δεν είναι άδειος. Ο Shift Register ανανεώνει το περιεχόμενο του, όταν είναι άδειος και δεν εξελίσσεται εκπομπή δεδομένων ή αμέσως μετά το τελευταίο bit τερματισμού του προηγούμενου μεταδιδόμενου πλαισίου. Όταν ο shift register πάρει τα δεδομένα, θα μεταδώσει ένα ολοκληρωμένο πλαίσιο με ρυθμό μετάδοσης ίσο με το baud rate.

Η κατάσταση του πομπού ελέγχεται μέσω δύο σημαιών:

Το bit USART Data Register Empty (UDRE) όπου δηλώνει πότε ο buffer είναι έτοιμος να δεχτεί νέα δεδομένα. Όταν έχει τιμή ‘1’ ο buffer είναι άδειος και μπορεί να φορτωθεί με νέα δεδομένα όταν είναι ‘0’ σημαίνει πως ο buffer είναι φορτωμένος με δεδομένα που αναμένουν να μεταδοθούν στον shift register και συνεπώς δεν μπορεί να πάρει νέα δεδομένα

Το bit Transmit Complete (TXC) για την κατάσταση του shift register. Όταν είναι σε λογικό ‘1’ σημαίνει ότι το πλαίσιο έχει μεταφερθεί και ότι δεν υπάρχουν νέα δεδομένα προς μετάδοση. Το bit αυτό παίρνει αυτόματα την τιμή ‘1’ όταν ολοκληρώνεται μεταφορά ενός πλαισίου.

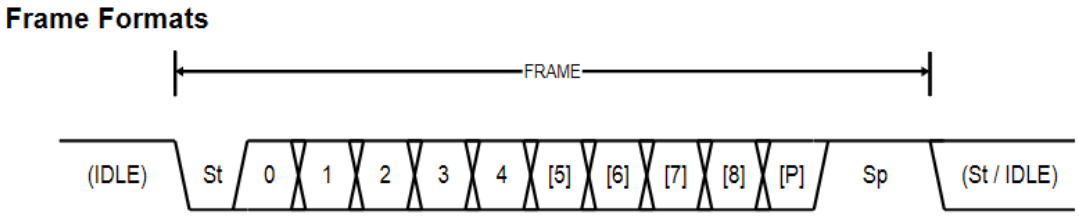
3. Δέκτης: Είναι το πιο πολύπλοκο τμήμα της USARTεπικοινωνίας. Αποτελείται από την μονάδα ανάκτησης (recovery unit), τον εκλεχτή ισοτιμίας (parity checker),μονάδα ελέγχου (Control Logic), καταχωρητή σειριακής μετατόπισης (Shift Register) και buffer δύο επιπέδων First-In First-Out (UDR0). Ο δέκτης υποστηρίζει τις ίδιες μορφές πλαισίου όπως ο πομπός και μπορεί να ανιχνεύσει πλαίσια, υπερχείλιση δεδομένων και λάθη ισοτιμίας.

Ο δέκτης ενεργοποιείται όταν το bit Receive Enable (RXEN) του UCSRnB register είναι ‘1’. Το pin RxDn λειτουργεί ως σειριακή είσοδος. Η λήψη των δεδομένων ξεκινάει όταν ανιχνευθεί το bit τεκκίνησης. Ακολουθούν με ρυθμό μετάδοσης ίσο με το baud rate τα επόμενα bit μέχρι να ληφθεί το πρώτο stop bit (αν υπάρχει δεύτερο stop bit αγνοείται). Όταν ολοκληρωθεί η μεταφορά ενός byte ο buffer το τοποθετεί στο "ταβάνι" τις λίστας FIFO. Μόλις γίνει η λήψη του πρώτου stop bit ο buffer ελέγχει αν είναι άδειος ο Receive Shift Register και το μεταφέρει εκεί. Αν ο Shift Register δεν είναι άδειος ο buffer είναι σε αναμονή. Σε περίπτωση που ο buffer δεχτεί και άλλο byte ενώ είναι σε αναμονή το τοποθετεί στη λίστα FIFO.

Ο δέκτης έχει μια σημαία που υποδηλώνει σε τι κατάσταση βρίσκεται. Το Receive Complete Flag (RXC) bit έχει τιμή ‘1’ όταν υπάρχουν δεδομένα στον buffer που δεν έχουν διαβαστεί και τιμή ‘0’ όταν ο buffer είναι άδειος.

3.2.. Μορφή Πακέτου

Τα δεδομένα μεταδίδονται σε πακέτα. Ένα πακέτο αποτελείται από το bit εκκίνησης, τα δεδομένα, το bit ελέγχου και τα bit τερματισμού.



Εικόνα 3.2 – Πλαίσιο USART Επικοινωνίας

-Το bit εκκίνησης: Όταν έχει τιμή ‘1’ η γραμμή για μεταφορά δεδομένων είναι απενεργοποιημένη. Για να ξεκινήσει η μεταφορά θα πρέπει το bit εκκίνησης να πάρει τιμή ‘0’. Από την πλευρά του δέκτη όταν αυτό το bit γίνει ‘0’ ξεκινάει η ανάγνωση των δεδομένων

-Δεδομένα: Είναι τα δεδομένα που πρέπει να μεταφερθούν. Ο αριθμός των bit που θα μεταφερθεί πρέπει δηλωθεί και μπορεί να είναι από 5 έως 9 bit. Αρχικά είναι το bit μικρότερης αξίας και ακολουθούν τα υπόλοιπα, μέχρι το bit μεγαλύτερης αξίας.

-Bit ισοτιμίας: Εμφανίζεται μόνο εφόσον είναι ενεργοποιημένο και ανάλογα με την τιμή που έχει μπορεί να υποδηλώνει μονή η ζυγή. Η τιμή του bit ισοτιμίας υπολογίζεται εκτελώντας την λογική πράξη exclusive-or σε όλα τα bit των «πραγματικών» δεδομένων. Αν έχει δηλωθεί περιττή ισοτιμία το αποτέλεσμα της πράξης αντιστρέφεται.

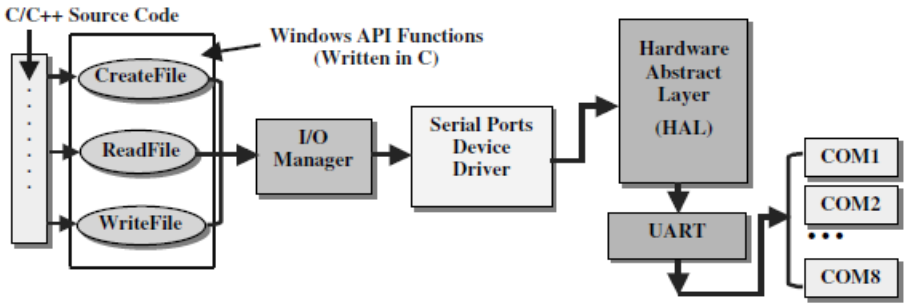
-Bit τερματισμού: Μπορεί να είναι 1 ή 2 bit και είναι πάντα ‘1’.

Οι ρυθμίσεις που αφορούν το format του πακέτου πρέπει να είναι ίδιες σε πομπό και δέκτη. Μετά την ολοκλήρωση της μεταφοράς ενός πακέτου, μπορεί να ξεκινήσει απευθείας μετάδοση καινούργιου πλαισίου ή να αδρανοποιηθεί η γραμμή επικοινωνίας ‘παίρνοντας λογικό 1’ ώστε να είναι σε αναμονή για το επόμενο πλαίσιο.

Για να υλοποιηθεί η επικοινωνία πρέπει να γίνει αρχικοποίηση της επικοινωνίας. Η αρχικοποίηση περιλαμβάνει τον ορισμό του ρυθμού μετάδοσης, του format του πλαισίου και την ενεργοποίηση πομπού και δέκτη.

3.3. UART επικοινωνία στον πρόγραμμα εφαρμογής [3]

Ένας serial port driver χρησιμοποιείται για γενικές λειτουργίες και εφαρμόζεται σε όποια συσκευή χρησιμοποιεί σειριακή επικοινωνία. Ο driver αυτός είναι λογισμικό που λειτουργεί ως γέφυρα για την διασύνδεση του προγράμματος και την σειριακής θύρας για την υλοποίηση σειριακής επικοινωνίας μεταξύ του υπολογιστή και της συσκευής. Στην εικόνα φαίνεται η διεπαφή σειριακής θύρας.



Εικόνα 3.3 – Διεπαφή σειριακής θύρας στο λειτουργικό των Windows

Όλες οι διεπαφές με την σειριακή θύρα γίνονται με την κλήση τριών συναρτήσεων του WIN32 API (Applications Programming Interface): CreateFile(), ReadFile(), WriteFile(). Οι συναρτήσεις παρέχουν ένα μονοπάτι για επικοινωνία με τον driver σειριακής θύρας προκειμένου να αποκτήσουν πρόσβαση στο hardware της σειριακής επικοινωνίας.

Οι συναρτήσεις δεν έχουν απευθείας πρόσβαση στο driver καθώς τα Windows είναι ένα σύστημα πολυδιεργασίας. Για αυτό ο I/O manager διαχειρίζεται τις διεργασίες που ζητάνε πρόσβαση στο hardware της σειριακής επικοινωνίας. Οποιαδήποτε διεργασία ζητήσει πρόσβαση στη σειριακή θύρα , στέλνει μια αίτηση στον I/O manager ο οποίος είναι σε απευθείας επικοινωνίας με τον Serial Port Device Driver. Ανάμεσα στον Serial Port Device Driver και στο πραγματικό hardware της θύρας παρεμβάλλεται το Hardware Abstract Layer (HAL), το οποίο είναι ένα τμήμα κώδικα που χαρτογραφεί το UART στην μνήμη του συστήματος. Το HAL ελέγχει την πρόσβαση στο hardware ενώ το UART ελέγχει τις σειριακές θύρες.

Στο σχήμα φαίνεται η διαδικασία αίτησης για σειριακή επικοινωνίας στο λειτουργικό Windows NT/2000/XP. Η πολυπλοκότητα του μηχανισμού δεν αποτελεί πρόβλημα καθώς το μόνο που απαιτείται είναι η χρήση των κατάλληλων συναρτήσεων για την διαχείριση των θυρών και τα υπόλοιπα τα αναλαμβάνει το λειτουργικό.

Η αρχικοποίηση της επικοινωνίας γίνεται ορίζοντας παραμέτρους στην δομή δεδομένων DCB. Η συνάρτηση CreateFile() ανοίγει μια σειριακή θύρα με προκαθορισμένες (default) ρυθμίσεις. Συνήθως χρειάζεται να γίνουν αλλαγές στις ρυθμίσεις. Η συνάρτηση GetCommState() ανακτά τις προκαθορισμένες ρυθμίσεις και η συνάρτηση SetCommState() ορίζει νέες τιμές. Επίσης μέσω της δομής COMMTIMEOUTS ορίζονται χρονικά όρια κατά την εγγραφή/διάβασμα δεδομένων. Υπέρβαση αυτών των ορίων προκύπτει σφάλμα και επιστρέφει τον αριθμό των bit που μεταφέρθηκαν επιτυχώς.

4. Αλγόριθμοι γραμμικής και κυκλικής παρεμβολής

Σε μια μηχανή CNC, ο έλεγχος περιλαμβάνει την κίνηση του κάθε άξονα ακολουθώντας συγκεκριμένες συντεταγμένες έτσι ώστε η κεφαλή (και άρα το εργαλείο κατεργασίας) να διαγράψει καθορισμένη τροχία σε σχέση με το δοκίμιο κατεργασίας. Η διαδικασία περιλαμβάνει, το διάβασμα των συντεταγμένων από τον κώδικα G και την εφαρμογή αλγόριθμου διερμηνίας του κώδικα. Ο αλγόριθμος αναπαράγει τα κατάλληλα σήματα που περιγράφουν την γεωμετρία του σχεδίου, τα σήματα οδηγουνται στους drivers και τα moter περιστρέφονται αναλόγως.

Η συνολική διαδρομή που καλείται να διαγράψει το εργαλείο κατεργασίας μπορεί να αναλυθεί σε μικρότερα ευθύγραμμα και κυκλικά τμήματα. Συνεπώς το πρόγραμμα ελέγχου είναι συνδυασμός γραμμικών και κυκλικών αλγορίθμων δρομολόγησης. Οι αλγόριθμοι αυτοί συνηθίζεται να γράφονται σε γλώσσα assembly προκυμμένου να επιτευχθεί όσο το δυνατόν πιο άμεση απόκριση.

Η εκτέλεση του αλγορίθμου γίνεται από το Software ενώ η εκτέλεση των επαναλήψεων γίνεται στο Hardware. Υπάρχουν δύο διαφορετικές φιλοσοφίες αλγορίθμων για αριθμητικό έλεγχο. Η πρώτη κατηγορία επιστρέφει ως αποτέλεσμα έναν διακριτό παλμό για τον κάθε άξονα. Κάθε φορά που ο αλγόριθμος εκτελείται χρησιμοποιεί ως δεδομένα, την τρέχουσα θέση του κάθε άξονα, και με βάση τις τελικές συντεταγμένες αναπαράγει έναν παλμό. Ο παλμός θα περιστρέψει το μοτέρ κατά ένα step. Η συχνότητα δημιουργίας των παλμών ορίζει την ταχύτητα. Η συχνότητα αυτή σχετίζεται άμεσα με τον χρόνο που χρειάζεται να ολοκληρωθεί ο αλγόριθμος. Ο αλγόριθμος επαναλαμβάνεται από την αρχή έχοντας σαν δεδομένο την νέα θέση και τερματίζεται όταν φθάσει στο τελικό σημείο.

Η δεύτερη κατηγορία αλγορίθμου, αντί για μεμονωμένους παλμούς επιστρέφει μια διαδική λέξη n – bit για την μετατόπιση και μία δεύτερη για την ταχύτητα στον κάθε άξονα. Η διαδική λεξή υπολογίζεται με βάση το αρχικό και τελικό σημείο. Η μετατόπιση αποθηκεύεται σε έναν καταχωρητή που λειτουργεί ως μετρητής και κάθε παλμός που παράγεται για την κίνηση του moter μειώνει τον μετρητή μέχρι να μηδενιστεί. Σε έναν δεύτερο καταχωρητή αποθηκεύεται μια τιμή που σχετίζεται με την ταχύτητα και κάθε φορά που αυτός ο μετρητής μηδενίζεται δημιουργείται ένας παλμός.

Ο σχεδιασμός μιας μηχανής αριθμητικού ελέγχου σχετίζεται με την επιλογή της κατηγορίας του αλγορίθμου που θα χρησιμοποιηθεί και την βέλτιστη επιλογή των παραμέτρων για την εκτελεσή του. Οι αλγόριθμοι που βασίζονται στην αναπαραγωγή διακριτών παλμών είναι απλοί στον προγραμματισμό αλλά θέτουν περιορισμό στην ταχύτητα και είναι ακατάλληλοι για εφαρμογές όπου απαιτείται μεγάλη ταχύτητα. Οι αλγόριθμοι που περιγράφουν τα χαρακτηριστικά της κίνησης, με δυαδική λέξη δεν

θέτουν περιορισμό στην ταχύτητα αλλά είναι πολύπλοκοι και υστερούν συγκριτικά στην ακρίβεια θέσης.

Η ερμηνεία ευθύγραμμης κίνησης είναι σχετικά απλή. Πιο σύνθετοι είναι οι αλγόριθμοι που περιγράφουν τόξο καθώς χρειάζεται να γίνει επίλυση εξισώσεων δευτέρου βαθμού.

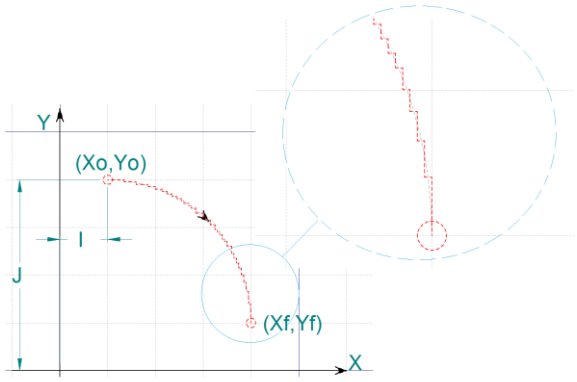
4.1 Αλγόριθμοι κυκλικής παρεμβολής βασισμένοι σε διακριτούς παλμούς[8]

Πολλοί από τους αλγορίθμους αυτής της κατηγορίας εφαρμόζονται στην ψηφιακή επεξεργασία εικόνας. Η αντιστοίχιση είναι ότι στην μια περίπτωση αναφερόμαστε σε περιστροφή ενός κινητήρα κατά συγκεκριμένη γωνία ενώ στην άλλη σε pixels.

Οι αλγόριθμοι κυκλικής παρεμβολής που βασίζονται σε παλμούς δημιουργούνε μια αλληλουχία παλμών. Ένας παλμός αντιστοιχεί σε μια βασική μονάδα μετατόπισης (**BLU** – μετατόπιση ένα micro-step). Η ταχύτητα είναι ανάλογη τις συχνότητας των παλμών ενώ η θέση υπολογίζεται από το γινόμενο του αριθμού των παλμών και της βασικής μονάδας μετατόπισης.

Σε όλους τους αλγόριθμους – παλμού ένα σήμα διακοπής (interrupt) προκαλεί την εκτέλεση του αλγορίθμου και την δημιουργία ενός παλμού. Η μέγιστη ταχύτητα κατεργασίας είναι ανάλογη της συχνότητας που μπορεί να παραχθεί ένας παλμός και εξαρτάται από τον χρόνιο εκτέλεσης του αλγορίθμου.

Η κεντρική ιδέα των αλγορίθμων αυτού του είδους είναι: η σύγκριση της πραγματικής θέσης της κεφαλής σε σχέση με την θεωρητική και μετακίνηση της κεφαλής κατά ένα micro step προς την θεωρητική θέση.



Βασικοί αλγόριθμοι αυτής της κατηγορίας είναι: Διαφορικής Ψηφιακής Ανάλυσης (DDA), Μέθοδος Σκαλοπάτι, Απευθείας Αναζήτησης (DSM)

4.2 Αλγόριθμοι κυκλικής παρεμβολής βασισμένοι σε δυαδική λέξη [9]

Η «αναλογική» προσέγγιση στην δημιουργία κυκλικού τόξου είναι η ταυτόχρονη κίνηση σε δύο άξονες. Προκειμένου να επιτευχθεί σταθερή ταχύτητα κατεργασίας Vo θα πρέπει η ταχύτητα του κάθε άξονα να μεταβάλλεται συνεχώς έτσι ώστε κάθε χρονική στιγμή t η συνισταμένη (που είναι εφαπτόμενη σε κάθε σημείο του τόξου) να είναι ίση με Vo (εικόνα 4.5).

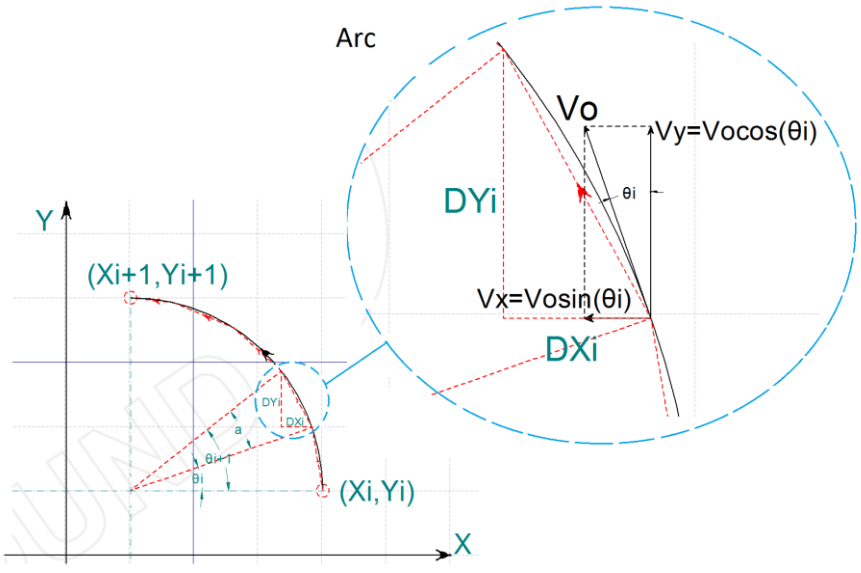
$$V_x(t) = V_0 sin\theta(t) \ (4.16a)$$

$$V_y(t) = V_0 cos\theta(t) \ (4.16\beta)$$

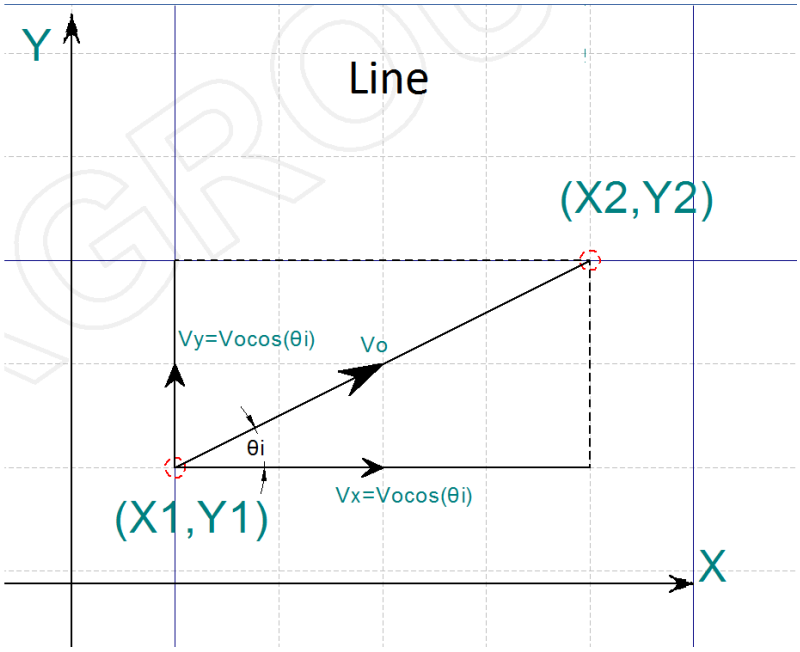
όπου:

$$\theta(t) = \frac{V_0}{R} t$$

Καθώς το όλο σύστημα βασίζεται σε ψηφιακή λογική ο υπολογισμός της ταχύτητας γίνεται σε ορισμένα σημεία της περιφέρειας του τόξου. Ο αλγόριθμος υπολογίζει τις ταχύτητες Vx και Vy που αντιστοιχούν σε κάποιο σημείο του τόξου και οι άξονες κινούνται διαγράφοντας ευθύγραμμη ομαλή κίνηση. Ανάλογα με την τιμή που έχουν οι ταχύτητες η κεφαλή διαγράφει ευθύγραμμη ομαλή κίνηση με κλίση Vy/Vx και με ταχύτητα κατεργασίας Vo. Όταν ολοκληρωθεί η υπολογισμένη μετατόπιση σε κάθε άξονα (όποτε η κεφαλή βρεθεί στο επόμενο σημείο της περιφέρειας του τόξου) ο αλγόριθμος υπολογίζει τις νέες τιμές για τις ταχύτητες Vx και Vy για την νέα θέση. Η διαδικασία συνεχίζεται μέχρι να φθάσουμε στο τελικό σημείο. Κατά αυτόν τον τρόπο η κίνηση αναλύεται σε πολύ μικρές χορδές. Ουσιαστικά δεν είναι τόξο αλλά ένα πολύγωνο (η τμήμα ενός πολυγώνου) το οποίο είναι εγγεγραμμένο στο τόξο. Ο αριθμός των γωνιών του πολυγώνου είναι αρκετά μεγάλος και μάλιστα όσο περισσότερες οι γωνίες τόσο καλύτερη η ανάλυση και η προσέγγιση στο πραγματικό τόξο (εικόνα 4.5).



Εικόνα 4.1 – πολύγωνο εγγεγραμμένο σε τόξο



Εικόνα 4.2 – ευθύγραμμη κίνηση με σταθερή ταχύτητα

Μετά από κάθε μετακίνηση η κεφαλή έχει διαγράψει μια γωνία a. Ο τρόπος υπολογισμού της γωνίας a εξαρτάται από τον αλγόριθμο που θα χρησιμοποιηθεί. Αρχικά όλοι οι αλγόριθμοι δέχονται:

$$cos\theta(i+1) = Acos\theta(i) - Bsin\theta(i)\ (4.17\ a)$$

$$sin\theta(i+1) = Asin\theta(i) - Bcos\theta(i)\ (4.17\ \beta)$$

με τους συντελεστές A και B:

$$A = cosa\ (4.18\ a)$$

$$B = sina\ (4.18\ \beta)$$

και

$$\theta(i+1) = \theta(i) + a\ (4.19)$$

και οι αντίστοιχες μετατοπίσεις να τερματίζονται στα σημεία X(i+1),Y(i+1):

$$X(i+1) = R(i)cos\theta(i+1)\ (4.20\ \alpha)$$

$$Y(i+1) = R(i)sin\theta(i+1)\ (4.20\ \beta)$$

Αντικαθιστώντας την 4.17 στην 4.20 προκύπτει:

$$X(i+1) = AX(i) - BY(i)(4.21\ \alpha)$$

$$Y(i+1) = AY(i) + BX(i)(4.21\ \beta)$$

Η εξίσωση 4.21 χρησιμοποιείται για τον υπολογισμών των διαδοχικών σημείων. Η μόνη διαφορά μεταξύ των αλγορίθμων αυτής της κατηγορίας είναι η μέθοδος προσδιορισμού των συντελεστών Α και Β. Με την επιλογή της γωνίας α εκτελείται ο αλγόριθμός ως εξής:

1. Για κάθε σημείο X(i),Y(i), υπολογίζονται οι συντεταγμένες του επόμενου σημείου X(i+1), Y(i+1) σύμφωνα με την εξίσωση . Η μετατόπιση σε κάθε άξονα είναι:

$$DX(i) = X(i+1) - X(i) = (A-1)X(i) - BY(i) \ (4.22\ \alpha)$$

$$DY(i) = Y(i+1) - Y(i) = (A-1)Y + BX(i) \ (4.22\ \beta)$$

με:

$$V_X(i) = V_0 DX(i)/DS(i) (4.23\ \alpha)$$

$$V_Y(i) = V_0 DY(i)/DS(i) (4.23\ \beta)$$

όπου:

$$DS(i) = \sqrt{DX^2(i) + DY^2(i)}$$

2. Οι τιμές τις εξίσωσης 7 (4.22) είναι η μετατόπιση και οι τιμές της εξίσωσης (4.23) είναι οι ταχύτητες. Οι τιμές αυτές θα αποθηκευτούνε σε κατάλληλους καταχωρητές

3. Όταν ολοκληρωθεί η μετακίνηση επαναλαμβάνεται ο αλγόριθμος με τις νέες τιμές X(i) και Y(i).

Επειδή η γωνία α είναι σχετικά μικρή το μήκος της μετατόπισης DS μπορεί να προσεγγιστεί με το μήκος του τόξου που αντιστοιχεί στην γωνία α (Ra) και άρα ο υπολογισμός της ταχύτητας απλοποιείται σε :

$$V_x(i) = KDX(i) (4.24\ \alpha)$$

$$V_Y(i) = KDY(i) (4.24\ \beta)$$

Όπου K=Vo/R. Η παράμετρος K είναι σταθερή και υπολογίζεται μόνο μια φορά για κάθε τόξο. Συνεπώς η ταχύτητα είναι ανάλογη της μετατόπισης στον κάθε άξονα.

Με την προσέγγιση δημιουργίας τόξου με μικρά ευθύγραμμα τμήματα προκύπτουν δύο τύποι σφάλματος

- Ακτινικό σφάλμα (Σφάλμα ακτίνας) ER:

$$ER(i) = R(i) - R = \sqrt{X^2(i) + Y^2(i)} - R \ (4.25)$$

-Σφάλμα ύψους χορδής ΕΗ:

$$EH(i) = R - R(i)\cos\left(\frac{a}{2}\right) (4.26)$$

Το σφάλμα ακτίνας οφείλεται στην προσέγγιση που γίνεται στα Α και Β. Καθώς ο αλγόριθμος επαναλαμβάνεται όσες φορές χρειαστεί για να φθάσουμε στο τελικό σημείο αυτό το σφάλμα είναι συσωρευτικό. Το σφάλμα χορδής μετά από i επαναλήψεις του αλγόριθμου είναι:

$$ER(i) = i(C-1)R (4.27)$$

με

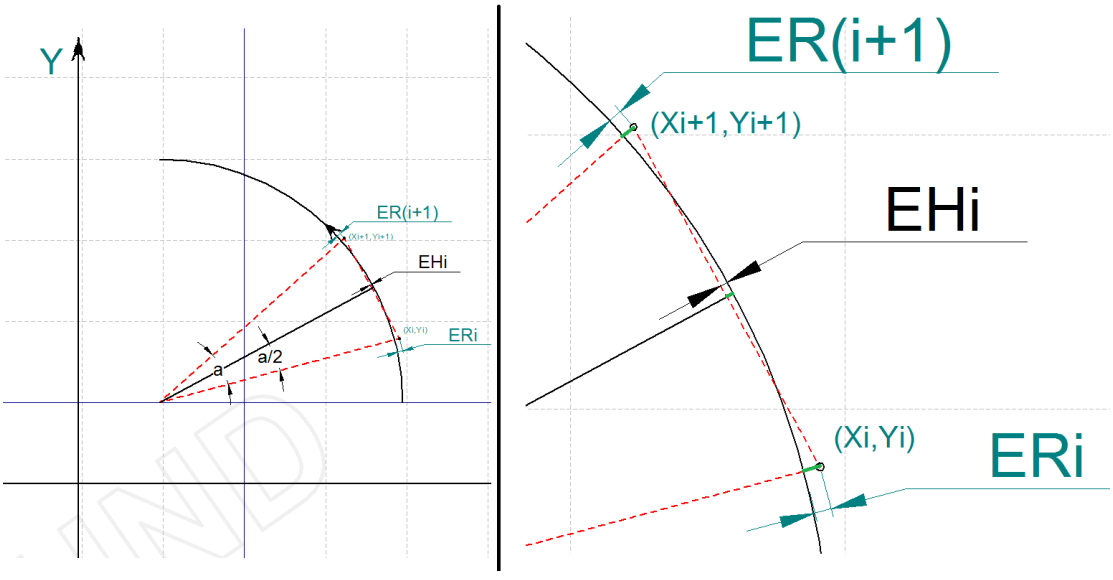
$$C = \sqrt{A^2 + B^2}$$

Το σφάλμα ακτίνας αυξάνεται με τον αριθμό των επαναλήψεων. Το μεγαλύτερο τόξο που μπορεί αποδώσει μια εντολή κώδικα G αντιστοιχεί σε ένα τεταρτημόριο. Και ο αριθμός τον επαναλήψεων για δεδομένη γωνία α είναι:

$$N=\pi/2\alpha (4.28)$$

Σε αυτή την περίπτωση το σφάλμα ακτίνας είναι:

$$ER_{max} = \left(\frac{\pi}{2\alpha}\right)(C-1)R (4.29)$$



Εικόνα 4.3 – Σφάλμα ακτίνας (ER) και Σφάλμα Χορδής (ΕΗ)

Σε αντίθεση το σφάλμα χορδής δεν συσσωρεύεται. Χρησιμοποιώντας την σχέση:

$$\cos\left(\frac{a}{2}\right) = \sqrt{\frac{1+cosa}{2}} = \sqrt{\frac{1+A}{2}} (4.30)$$

Στην εξίσωση 4.26:

$$EH(i) = R - R(i)\sqrt{\frac{1+A}{2}} \quad (4.31)$$

Και τα δύο σφάλματα εξαρτώνται από την γωνία α. Η επιλογή της γωνίας α πρέπει να γίνει έτσι ώστε τόσο το σφάλμα χορδής όσο και το σφάλμα ακτίνας να μην ξεπερνούν το ένα step.

Όπως είπαμε οι αλγόριθμοι αυτής της κατηγορίας διαφέρουν στον τρόπο υπολογισμού των συντελεστών Α και Β στην εξίσωση 4.22.Μερικοί γνωστοί αλγόριθμοι είναι η μέθοδος Euler, η βελτιωμένη μέθοδος Euler, η μέθοδος Taylor, η μέθοδος Tustin, η βελτιωμένη μέθοδος Tustin.

5. Υλοποίηση και Εφαρμογή

5.1. Computer Aided Manufacturing

5.1.1. Εξόρυξη δεδομένων από αρχείο CAD

Το λογισμικό CAM είναι γραμμένο σε C++. Με την έναρξη το πρόγραμμα ζητάει από τον χρήστη το αρχείο που προορίζεται για επεξεργασία. Αφού βρεθεί το αρχείο, ανοίγει σε κατάσταση για διάβασμα και γίνεται σάρωση γραμμή προς γραμμή. Κατά την ανάγνωση αν διαβαστεί ετικέτα που προσδιορίζει μια οντότητα που μας ενδιαφέρει (γραμμή, κύκλος τόξο), αποθηκεύει σε μεταβλητές, τον τύπο της οντότητας και τις συντεταγμένες.

Για την αποθήκευση των δεδομένων δημιουργήθηκε μια κλάση **Entity** η οποία περιλαμβάνει τα εξής μέλη δεδομένα:

char type[2]: για προσδιορισμό του τύπου της οντότητας. Οι τιμές που μπορεί να πάρει είναι: G1 για γραμμή, G2 για αριστερόστροφο τόξο, G3 για δεξιόστροφο τόξο

float x1,y1: για τις συντεταγμένες του αρχικού σημείου

float x2,y2: για τις συντεταγμένες του τελικού σημείου

float I1,J1: σχετικές συντεταγμένες του αρχικού σημείου σε σχέση με το κέντρο του τόξου

και τις συναρτήσεις μέλη:

void SetLine(char type_1[],float x_1,float y_1,float x_2,float y_2) και **void SetArc**(char type_1[],float x_1,float y_1,float x_2,float y_2,float I_1,float J_1) για την δημιουργία οντότητας ευθείας και τόξου αντίστοιχα

v**oid PrintLine**() και **void PrintArc**() για εκτύπωση των μεταβλητών οντότητας ευθείας και τόξου αντίστοιχα.

char *Get_type(), **float Get_x1**(), **float Get_y1**(), **float Get_x2**(), **float Get_y2**(), **float Get_I1**() και **float Get_J1**() για επιστροφή των μεταβλητών type,x1,y1,x2,y2,I και J αντίστοιχα.

Όσο αναφορά την ευθεία: οι αρχικές και τελικές συντεταγμένες δίνονται σε σχέση με το σημείο (0,0).

Όσο αναφορά το τόξο: Ανεξάρτητα με τον αν αυτό σχεδιάστηκε δεξιόστροφα ή αριστερόστροφα το .dxf αρχείο το περιγράφει ως αριστερόστροφο. Δίνονται οι συντεταγμένες (cx,cy) του κέντρου σε σχέση με το σημείο (0,0), η ακτίνα cr και οι γωνίες a1 και a2 που σχηματίζει ο οριζόντιος άξονας (άξονας x) με την ευθεία που διέρχεται από το κέντρο του τόξου και το αρχικό και τελικό σημείο αντίστοιχα. Οπότε για να βρούμε τις

συντεταγμένες του αρχικού και τελικού σημείου του τόξου καθώς και την σχετική συντεταγμένη του κέντρου σε σχέση με το αρχικό σημείο πρέπει να γίνουν οι εξής πράξεις

I=cos $\left(\frac{a1\times\pi}{180}\right)\times cr$ κέντρο τόξου κατά x σε σχέση με το αρχικό σημείο

J= sin $\left(\frac{a1\times\pi}{180}\right)\times cr$ κέντρο τόξου κατά y σε σχέση με το αρχικό σημείο

x1=I+cx αρχικό σημείο κατά x σε σχέση με την αρχή των αξόνων

y1=J+cy αρχικό σημείο κατά y σε σχέση με την αρχή των αξόνων

x2=cos $\left(\frac{a2\times\pi}{180}\right)\times cr$ +cx τελικό σημείο κατά x σε σχέση με την αρχή των αξόνων

y2=sin $\left(\frac{a2\times\pi}{180}\right)\times cr$ +cy τελικό σημείο κατά y σε σχέση με την αρχή των αξόνων

Για παράδειγμα για την οντότητα του σχεδίου 1.8. περιγράφεται αριστερόστροφο τόξο με a1=9.46^o, a2=99.46^o, cr=60.65 mm. Για τα υπόλοιπα δεδομένα ισχύει:

I=cos $\left(\frac{9,46\times\pi}{180}\right)\times 60,83 = 60\ mm$

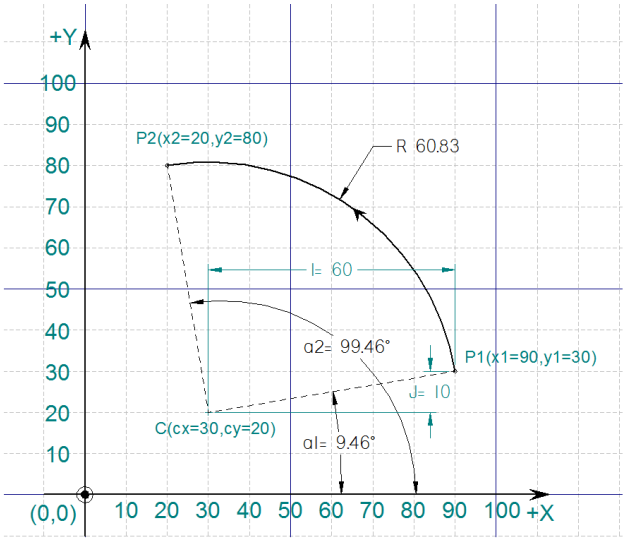
J= sin $\left(\frac{9,46\times\pi}{180}\right)\times 60,83 = 10\ mm$

x1=60+30=90

y1=10+20=30

x2=cos $\left(\frac{99,46\times\pi}{180}\right)\times 60,83$ +30=20

y2=sin $\left(\frac{99,46\times\pi}{180}\right)\times 60,83$ +20=80



Σχέδιο 5.1 - G03, υπολογισμός δεδομένων αριστερόστροφου τόξου

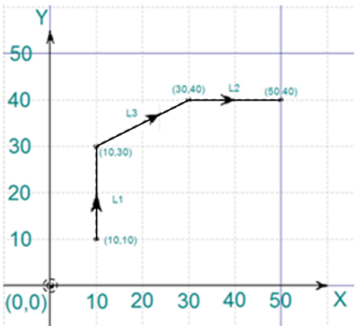
5.1.2. Αλγόριθμος στοίχισης γεωμετρικών στοιχείων (Παράρτημα Α1)

Η σειρά με την οποία παρουσιάζονται οι οντότητες στο αρχείο .dxf είναι ίδια με την σειρά που σχεδιάστηκαν. Το σχέδιο 1.7 αποτελείται από τρεις ευθείες γραμμές οι οποίες έγιναν με την εξής σειρά:

1) L1:P1(10,10) -> P2(10,30)

2) L2:P1(30,40) -> P2(50,40)

3) L3:P1(10,30) -> P2(30,40)




```
N001 G00 X10 Y10
N002 G01 X10 Y30 F20
N003 G00 X30 Y40
N004 G01 X50 Y40 F20
N005 G00 X10 Y30
N006 G01 X30 Y40 F20
M30
```

Σχέδιο 5.2 - κατεργασία όπως αρχικά σχεδιάστηκε

Ο κώδικάς που περιγράφει το παραπάνω σχέδιο αποτελείται από 6 εντολές και η κεφαλή ενεργοποιείται και απενεργοποιείται (on/off) τρεις φορές .

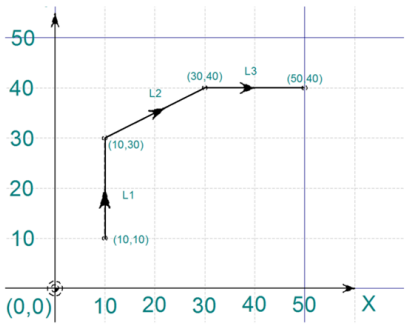
Αν αλλάξουμε την σειρά με την οποία θα γίνουν οι κατεργασίες και δεν ακολουθήσουμε την σειρά με την οποία έγινε το σχέδιο μπορεί να μειωθεί ο αριθμός των εντολών (αυτό συνεπάγεται μείωση χρόνου για την ολοκλήρωση της κατεργασίας). Επίσης μειώνεται ο αριθμός των εναλλαγών on/off της κεφαλής (άρα λιγότερες φθορές και πιο ομαλή κατεργασία).

Στο σχέδιο 1.8 έγινε ανταλλαγή (swap) στην σειρά της κατεργασίας L3 με την L2.

1) L1:P1(10,10) -> P2(10,30)

2) L2:P1(10,30) -> P2(30,40)

3) L3:P1(30,40) -> P2(50,40)




```
N001 G00 X10 Y
N002 G01 X10 Y30
N003 G01 X30 Y40
N004 G01 X50 Y40
N005 M30
```

Σχέδιο 5.3 - κατεργασία μετά από αναδιάταξη του κώδικα

Ο μήκος του κώδικα πλέον είναι 4 εντολές και η κεφαλή ενεργοποιείται - απενεργοποιείται (on/off) μια φορά. Σε πραγματικά σχέδια που περιλαμβάνουν χιλιάδες οντότητες η αναδιάταξη και στοίχιση των οντοτήτων οδηγεί σε σημαντική βελτίωση στην λειτουργία.

Διακρίνονται οι εξής περιπτώσεις:

case1: το τέλος μιας οντότητας E(i) συμπίπτει με το τέλος μιας άλλης οντότητας E(j), σχέδιο 1.9.

Αν η E(j) είναι:

Ευθεία:

1) γίνεται ανταλλαγή (swap) του αρχικού με το τελικού σημείου της οντότητας

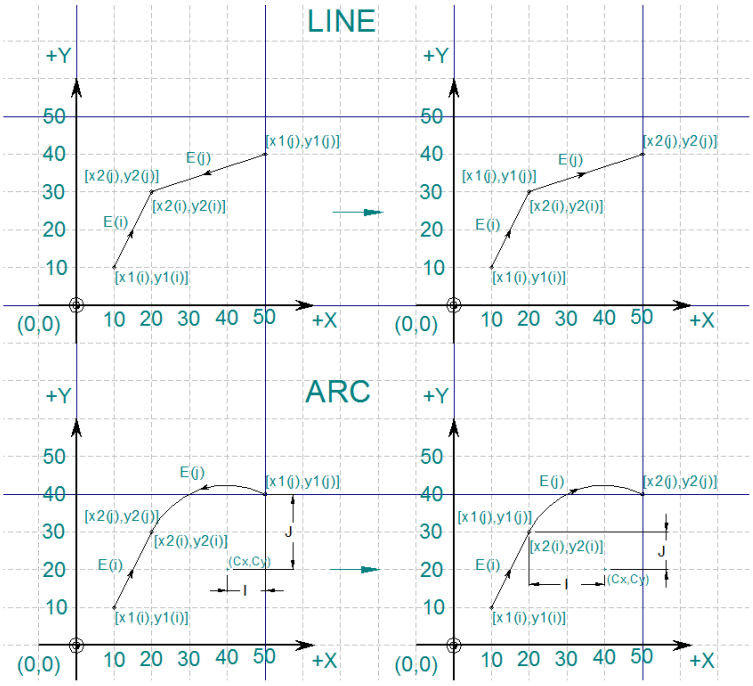
Τόξο:

1) Αλλάζει η φορά διαγραφής από αριστερόστροφη G03 σε δεξιόστροφη G02

2) Γίνεται ανταλλαγή (swap) του αρχικού με το τελικού σημείου της οντότητας

3) Υπολογίζονται οι σχετικές συντεταγμένες του νέου αρχικού σημείου σε σχέση με το κέντρο του τόξου

$$I=Cx-x1$$
$$J=Cy-y1$$



Σχέδιο 5.4 - Case 1: Το τέλος μιας οντότητας συμπίπτει με το τέλος άλλης οντότητας

case2: το αρχικό σημείο μιας οντότητας E(i) συμπίπτει με το αρχικό σημείο μιας άλλης οντότητας E(j), σχέδιο 1.10

Αν η E(j) είναι:

Ευθεία:

1) γίνεται ανταλλαγή (swap) του αρχικού με το τελικού σημείου της οντότητας

2) γίνεται ανταλλαγή (swap) των οντοτήτων

Τόξο:

1) Αλλάζει η φορά διαγραφής από αριστερόστροφη G03 σε δεξιόστροφη G02

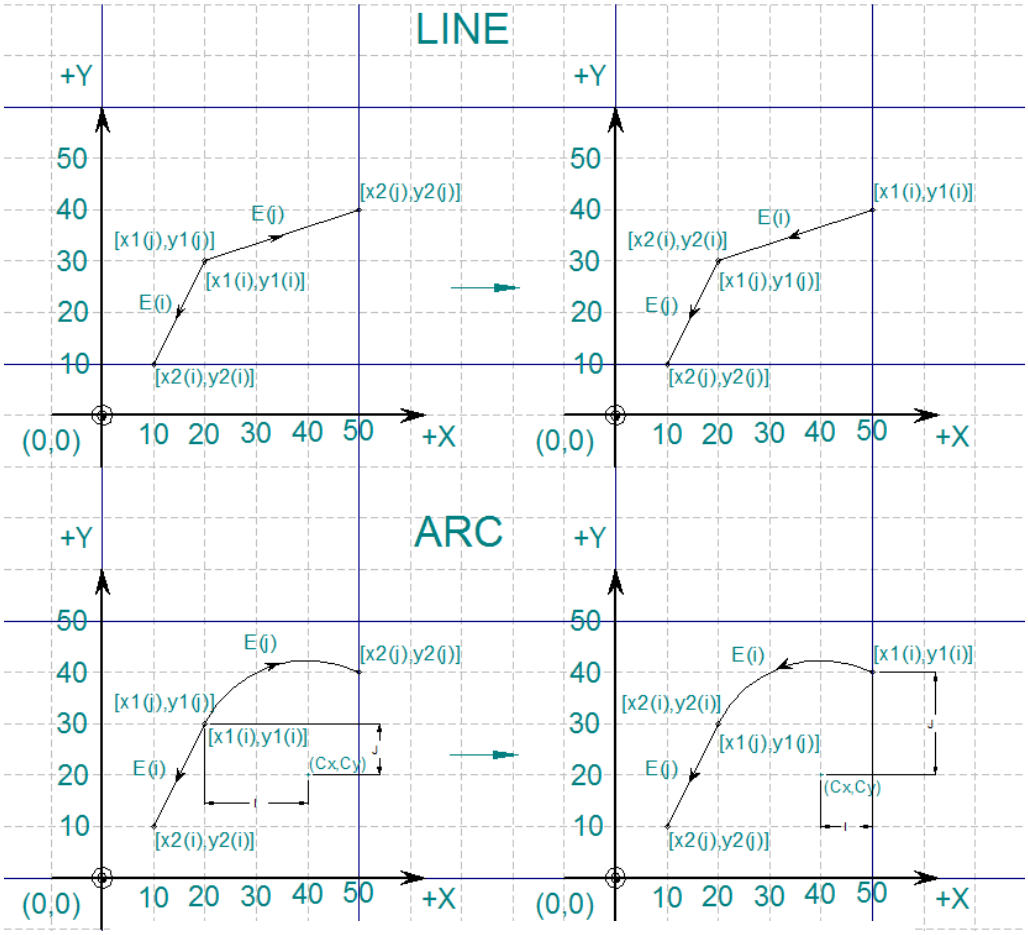
2) Γίνεται ανταλλαγή (swap) του αρχικού με το τελικού σημείου της οντότητας

3) Υπολογίζονται οι σχετικές συντεταγμένες του νέου αρχικού σημείου σε σχέση με το κέντρο του τόξου

$$I=Cx-x1$$

$$J=Cy-y1$$

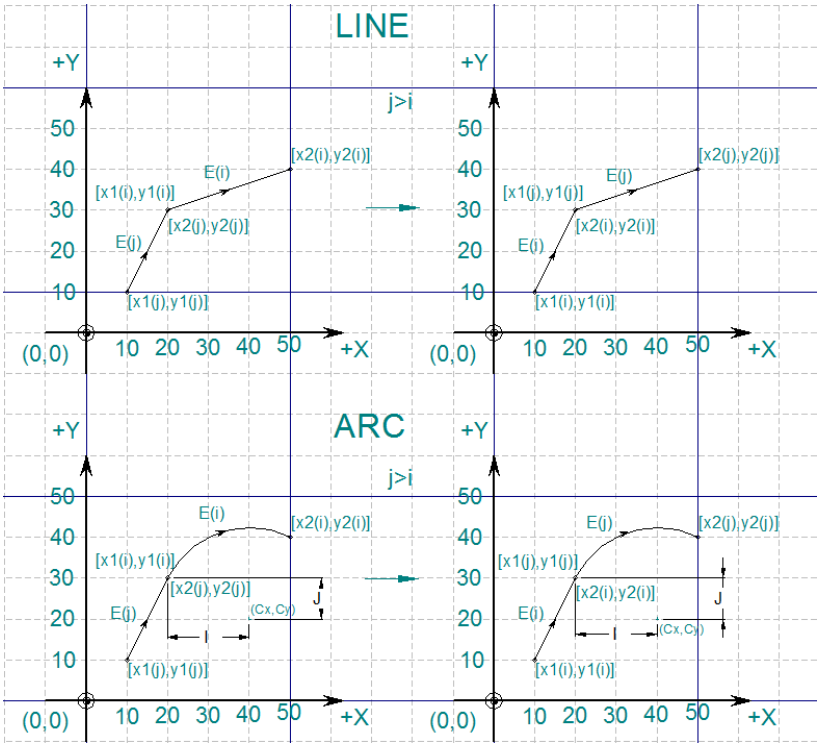
4) Γίνεται ανταλλαγή (swap) των οντοτήτων



Σχέδιο 5.5 - Case 2: Το αρχικό σημείο μιας οντότητας συμπίπτει με το αρχικό σημείο άλλης οντότητας

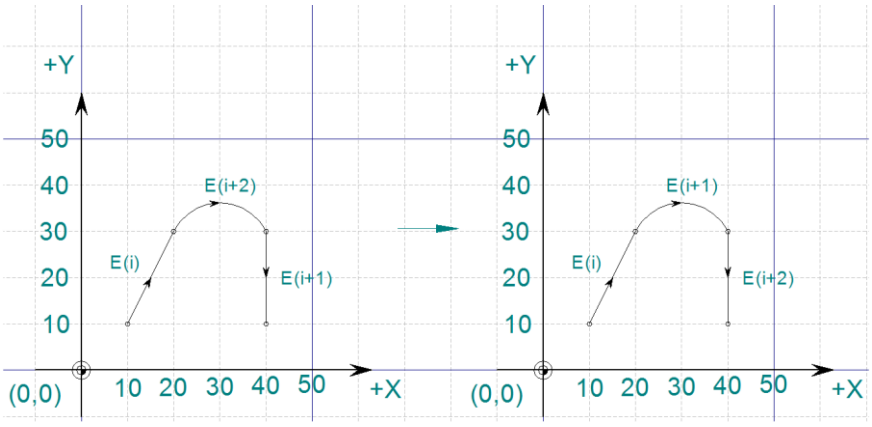
case3: το αρχικό σημείο μιας οντότητας E(i) συμπίπτει με το αρχικό σημείο μιας επόμενης οντότητας E(j) (j>i), σχέδιο 1.11.

1) γίνεται ανταλλαγή (swap) των οντοτήτων



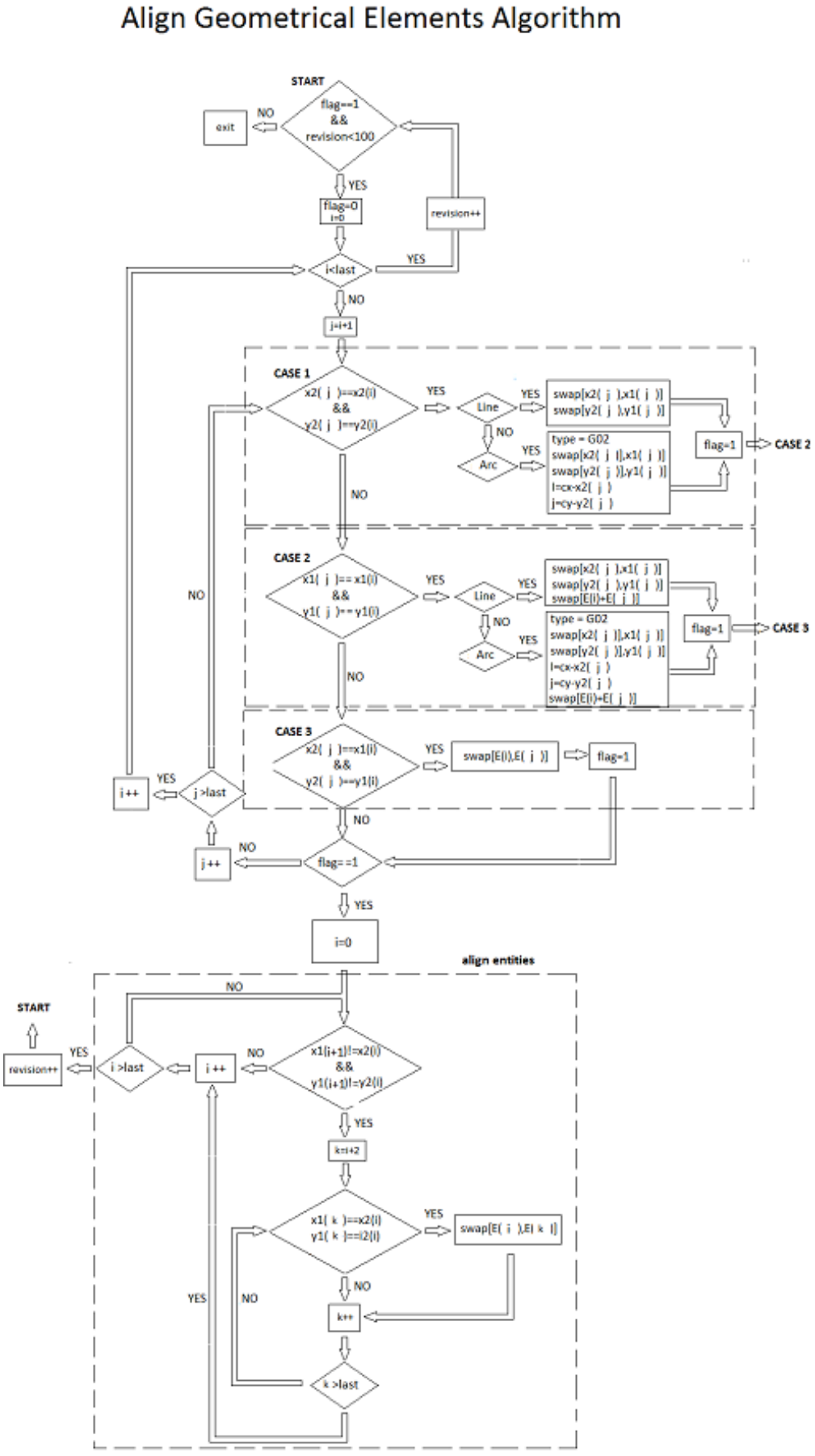
Σχέδιο 5.6 - Case 3: το αρχικό σημείο μιας οντότητας E(i) να συμπίπτει με το αρχικό σημείο μιας επόμενης οντότητας E(j) (j>i)οντότητας

στοίχιση: Αν κατά των έλεγχο των περιπτώσεων case1, case2 και case3 γίνει έστω και μια ανάλλαγη, ακολουθεί σάρωση και σύγκριση όλων των οντοτήτων μεταξύ τους και αναπροσαρμογή της θέσης τους, σχέδιο 1.12.



Σχέδιο 5.7 - Στοίχιση γεωμετρικών

Στην εικόνα 1.3 φαίνεται ο αλγόριθμος που περιλαμβάνει τις παραπάνω περιπτώσεις ολοκληρωμένος



Εικόνα 5.1 - Αλγόριθμος στοίχισης γεωμετρικών στοιχείων (Παράρτημα Α1)

5.1.3. Δημιουργία κώδικα G σχεδίου (αρχείου κειμένου .nc)

Μετά την εκτέλεση των συναρτήσεων στοίχισης των οντοτήτων, δημιουργείται το αρχείο κειμένου .nc το οποίο περιέχει το κώδικα G του σχεδίου που θα επεξεργαστούμε ακολουθώντας το πρότυπο ISO όπως αυτό περιεγράφηκε στο κεφάλαιο 1.

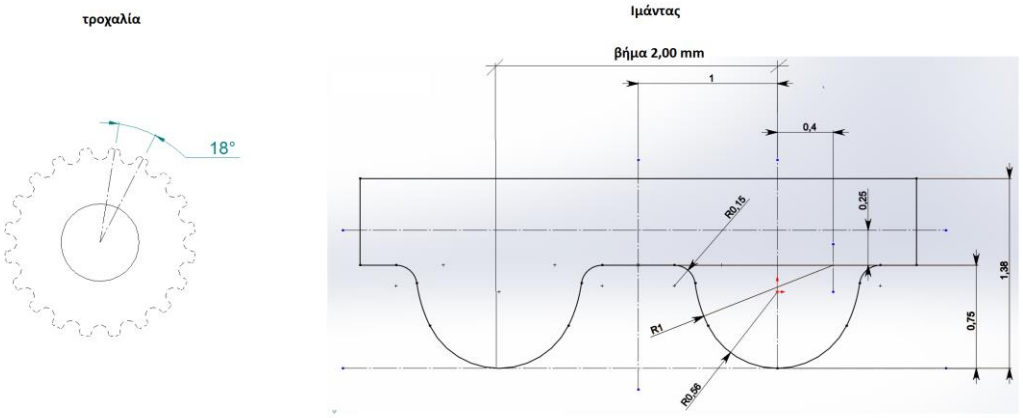
Ο αλγόριθμος είναι πολύ απλός:

Όταν το τελικό σημείο μιας οντότητας (i) δεν ταυτίζεται με το αρχικό σημείο της επόμενης οντότητας (i+1), τότε έχουμε κίνηση χωρίς κατεργασία (G00) με την μέγιστη ταχύτητα στο αρχικό σημείο της οντότητας (i+1). Ενώ αν υπάρχει ταύτιση τότε έχουμε κίνηση με κατεργασία (G01 ή G02 ή G03) , με ταχύτητα κατεργασίας στο τελικό σημείο της επόμενης (i+1).

5.2. Hardware

5.2.1. Μετάδοση κίνησης

Για τις ανάγκες της εργασίας χρησιμοποιήθηκε ιμάντας χρονισμού για λόγους απλοποίησης στον σχεδιασμό και μείωσης κόστους. Ο τύπος του ιμάντα είναι 2GT, και το βήμα είναι 2,00 mm



Σχέδιο 5.8 - Ιμάντας και τροχαλία 2GT

Με την τροχαλία να έχει 20 δόντια, η γωνία περιστροφής που αντιστοιχεί σε μετατόπιση ίση με ένα βήμα (2,00 mm), είναι φ=360/20=18^{o}. Η σχέση που εκφράζει την γραμμική μετατόπιση συναρτήσει της γωνίας περιστροφής a είναι:

$$dx = \frac{2a}{\varphi} = \frac{\alpha}{9} \quad (2.1)$$

Μια λογική σκέψη είναι: Να αυξήσουμε την ανάλυση (δηλαδή να μικρύνουμε το dx) επιλέγοντας τροχαλία μικρότερης διαμέτρου D. Η ακολουθία είναι D↓ → Αριθμός δοντιών↓ → φ↑ → dx↓. Μικρότερη τροχαλία όμως συνεπάγεται μικρότερη ροπή και άρα μεγαλύτερο μοτέρ.

5.2.2. Μηχανολογικό Σχέδιο

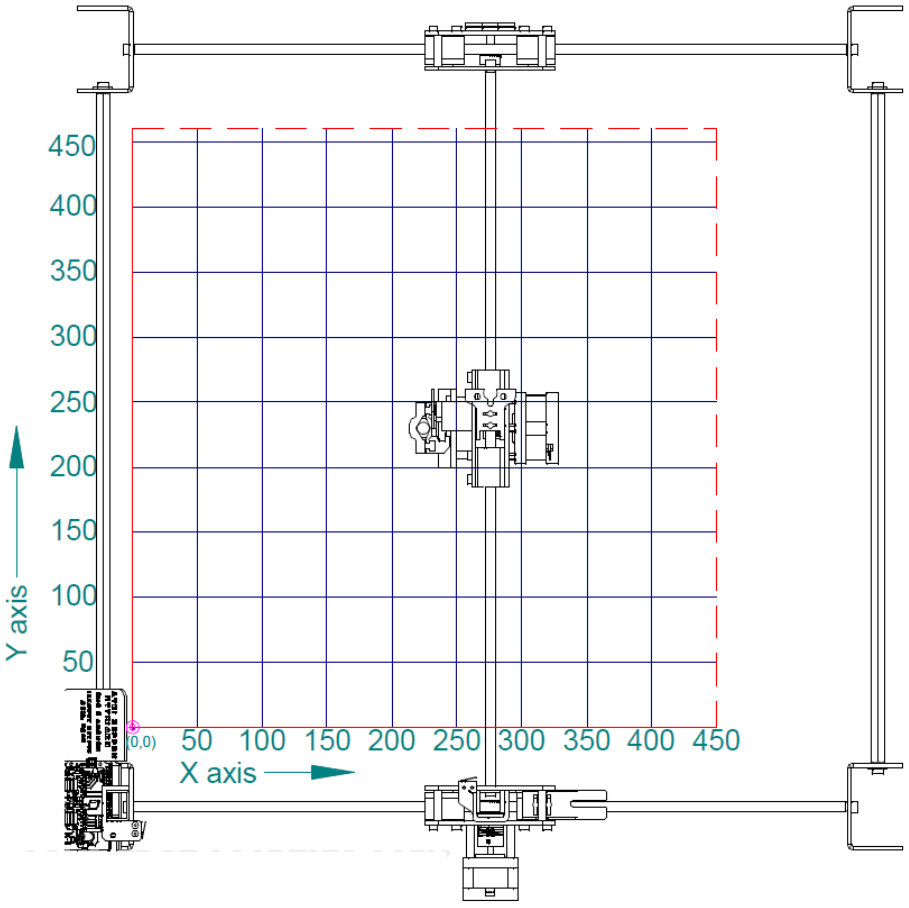
Έχοντας επιλέξει τον τρόπο μετάδοσης της κίνησης το επόμενο βήμα είναι η ολοκλήρωση του σχεδιασμού. Στο σχέδιο 2.4 φαίνονται οι βασικές εξωτερικές διαστάσεις του CNC καθώς και η ωφέλιμη σχεδιαστική επιφάνεια.

Γραφικό περιβάλλον εργασίας: Προκειμένου να υπάρξει δια-σύνδεση μεταξύ του σχεδίου και του CNC, δημιουργήθηκε ένα γραφικό περιβάλλον εργασίας όπως φαίνεται στην παρακάτω εικόνα.

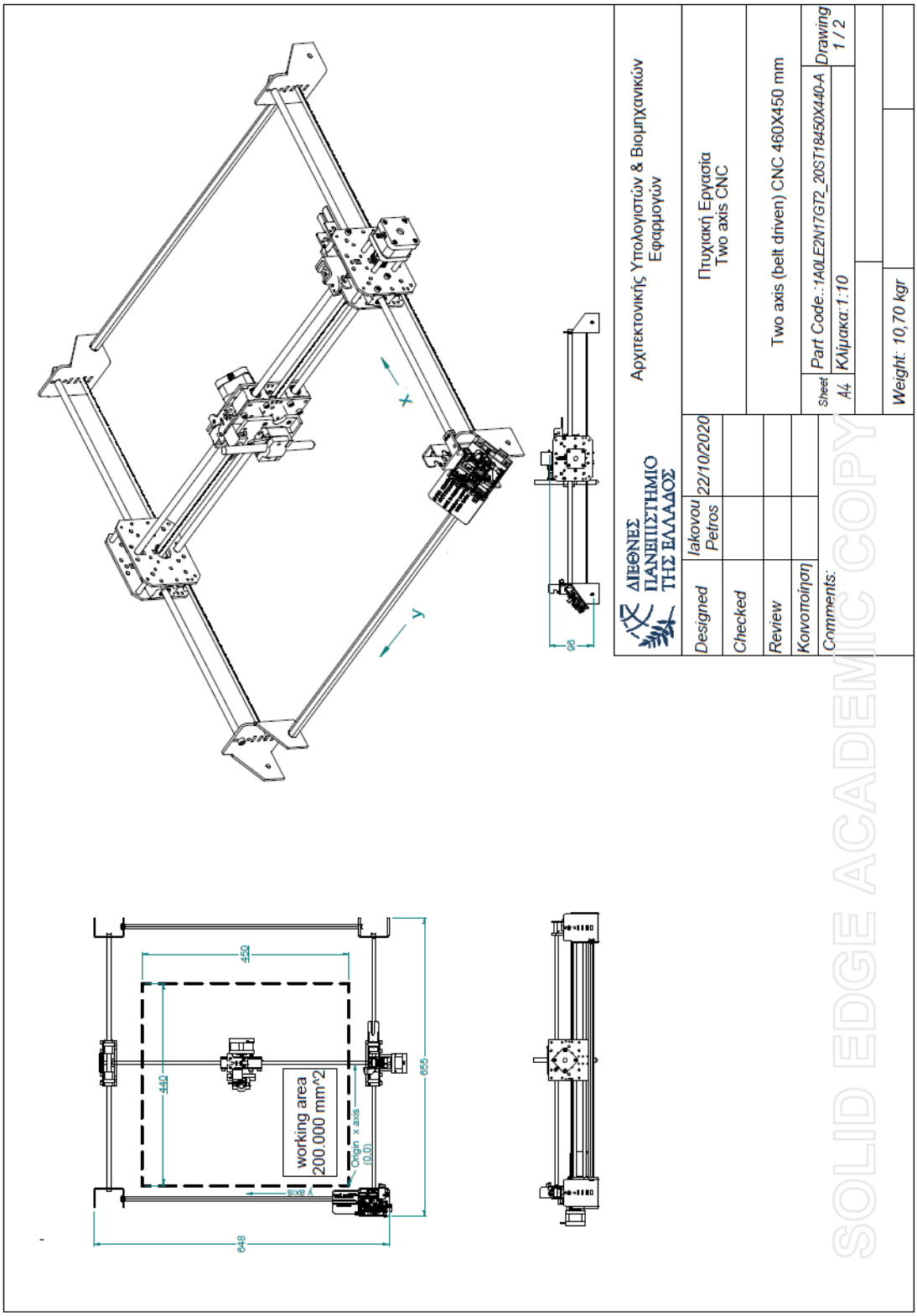
Απεικονίζεται το CNC και τα όρια της σχεδιαστικής επιφάνειας όπου μπορεί να γίνει κατεργασία. Φαίνεται το σημείο (0,0), και οι άξονες x και y βαθμονομημένοι. Με αυτόν τον τρόπο υπάρχει η δυνατότητα τοποθέτησης του σχεδίου σε συγκεκριμένη θέση.

Υπάρχει η δυνατότητα τοποθέτησης και επεξεργασίας ενός υπάρχοντος σχεδίου ή δημιουργίας ενός νέου σχεδίου από την αρχή. Εφόσον το σχέδιο έχει την τελική μορφή και είναι στην επιθυμητή θέση το αποθηκεύουμε σε μορφή κατάλληλη για ανάγνωση από το πρόγραμμα εφαρμογής (.dxf).

Μέσω λοιπόν αυτής της διαδικασίας γίνεται τοποθέτηση του σχέδιο σε συγκεκριμένη θέση.



Σχέδιο 5.9 - Γραφικό περιβάλλον διασύνδεσης



Σχέδιο 5.10 - CNC δύο αξόνων

5.2.3. Set up Driver

Η πλακέτα χρησιμοποιεί αντίσταση R_{ISEN}=0.1(Ohm). Σύμφωνα με την σχέση 2.2 ή τάση αναφοράς πρέπει να είναι:

$$V_{REF}=0.4(A)x5x0.1(Ohm)=0.2\ Volt$$

Το chipaki παρέχει δυνατότητα ρύθμισης της V_{REF} μέσω ενός ποτενσιόμετρου που έχει ενσωματωμένο πάνω του. Με το chipaki συνδεδεμένο περιστρέφουμε το ποτενσιόμετρο μέχρι να διαβάσουμε στο πολύμετρο 0.2 Volt. Εναλλακτικά μετράμε το ρεύμα που διαρρέει το τύλιγμα σε λειτουργία full step.

5.2.4. Set up Atmega 328p

Για την συγκεκριμένη εφαρμογή θα εκμεταλλευτούμε την ιδιότητα που έχει ο timer να κάνει toggle έναν συγκεκριμένο ακροδέκτη όταν ταυτιστεί με την τιμή που υπάρχει στο καταχωρητή σύγκρισης.

Παρακάτω γίνεται αναφορά στο set-up του timer0 (με παρόμοιο τρόπο γίνεται το set - up των άλλων timers). Χρησιμοποιούνται οι εξής καταχωρητές:

Καταχωρητής TCCR0A (TC0 Control Register A)

TCCRXA – **Control Register A**

Bit	7	6	5	4	3	2	1	0
	COM0A[1:0]		COM0B[1:0]				WGM0[1:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 7:6- COM0A[1:0] Compare Output Mode

Η τιμή στα δύο αυτά bit καθορίζουν την συμπεριφορά του pin OC0A (-PD6) (Output Compare pin A timer 0), σύμφωνα με τον παρακάτω πίνακα, όταν η το mode λειτουργίας έχει οριστεί σε CTC.

Table 19-3. Compare Output Mode, Non-PWM

COM0A[1]	COM0A[0]	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on compare match.
1	0	Clear OC0A on compare match.
1	1	Set OC0A on compare match.

Η λειτουργία που μας ενδιαφέρει είναι Toggle OC0A on compare match ("01"). Όταν ο timer γίνει ίσος με την τιμή σύγκρισης τότε ο ακροδέκτης OC0A (PD6) κάνει toggle την τιμή του. Ο ακροδέκτης αυτός είναι συνδεδεμένος με τον ακροδέκτη STEP του driver X. Οπότε όσο πιο μικρή είναι ή τιμή στον καταχωρητή σύγκρισης τόσο

μεγαλύτερη η συχνότητα του σήματος στον ακροδέκτη STEP και συνεπώς μεγαλύτερη ταχύτητα. Και αντίστροφα μεγαλύτερη τιμή στον καταχωρητή σύγκρισης → μικρότερη συχνότητα → μικρότερη ταχύτητα

Η συχνότητα με την οποία γίνεται εναλλαγή είναι:

$$f_{OC0A} = \frac{f_{clk\_I/O}}{2 * N * (1 + OCR0A)} \quad (2.3)$$

όπου:

$f_{clk\_i/o}$=16.000.000 hz είναι η συχνότητα ρολογιού για είσοδο - έξοδο

OCR0A είναι η τιμή που έχει ο καταχωρητής σύγκρισης

N ο prescaler που έχει επιλεγεί για τον timer και μπορεί να πάρει τιμές (1, 8,64 ,256 ή 1024)

Από την παραπάνω σχέση μπορούμε να υπολογίσουμε την τιμή που θα πρέπει να έχει ο καταχωρητής σύγκρισης για ορισμένη ταχύτητα v. Δεομένου ότι η κίνηση πρέπει να είναι ευθύγραμμη ομαλή (σταθερή ταχύτητα) κάθε χρονική στιγμή ισχύει:

$$s = v * t \rightarrow t = \frac{s}{v} \rightarrow \frac{1}{t} = \frac{v}{s} \rightarrow f_{OC0A} = \frac{v}{s} (2.4)$$

Η μετατόπιση που αντιστοιχεί σε ένα full step σύμφωνα με την σχέση 2.1 είναι:

$$Full\_Step = \frac{\frac{Pitch * Step\_Angle}{360}}{Pulley\_Teeth}$$

Όταν ο driver είναι σε mode micro-step τότε ή μετατόπιση για ένα micro-step είναι:

$$s = \frac{Full\_Step}{Step\_Mode} \quad (2.5)$$

Από τις 2.2,2.3 και 2.4 προκύπτει πως η σχέση που συνδέει ταχύτητα v με τον καταχωρητή σύγκρισης είναι:

$$OCR0A = \frac{f_{clk\_I/O}}{2 * N * \frac{v}{s}} - 1 \ (2.5)$$

Από την σχέδη 2.5 βλέπουμε πως η ταχύτητα είναι αντιστρόφως ανάλογη της τιμής του καταχωρητή σύγκρισης. Επίσης είναι αντιστρόφως ανάλογη του prescaller N. Για να έχουμε καλή ανάλυση, θέλουμε μικρή τιμή στο prescaller (ιδανικό N=1). Όσο πιο μικρό είναι το N όμως τόσο μεγαλύτερο το OCR0A. Ο καταχωρητής, όμως είναι OCR0A είναι μόνο 8 bit και σε μικρές ταχύτητες υπερχειλίζει. Όποτε αναγκαζόμαστε να μειώσουμε την ανάλυση αυξάνοντας τον prescaller. Στα συμπεράσματα θα δούμε λύσεις για να βελτιώσουμε την ανάλυση.

Ανάλογα με τον timer 0 γίνεται το set-up και στους άλλους timers.

Ο timer0 κάνει toggle τον ακροδέκτη PD6

Ο timer1 κάνει toggle τον ακροδέκτη PB1

Ο timer2 κάνει toggle τον ακροδέκτη PB3

Bits 1:0 – WGM0[1:0] Waveform Generation Mode

Τα bit αυτά σε συνδυασμό με το WGM02 bit στον καταχωρητή TCCR0B (που θα δούμε παρακάτω), καθορίζουν το mode λειτουργίας σύμφωνα με τον παρακάτω πίνακα.

Table 19-9. Waveform Generation Mode Bit Description

Mode	WGM0[2]	WGM0[1]	WGM0[0]	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on (1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCR0A	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCR0A	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCR0A	BOTTOM	TOP

Μας ενδιαφέρει η λειτουργία mode 2, CTC (**C**lear **T**imer on **C**ompare) "010". Όταν ο timer γίνει ίσος με την τιμή στο καταχωρητή σύγκρισης μηδενίζει την τιμή του και ξεκινάει να μετράει από την αρχή.

Καταχωρητής TCCR0B (TC0 Control Register B)

TCCR0B – **TC0 Control Register B**

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bits 2:0 – CS0[2:0] Clock Select 0

Ο συνδυασμός των τριών αυτών bit καθορίζει τον αριθμό τον χτύπων ρολογιού όπου θα γίνεται η προσαύξηση του counter.

Table 19-10. Clock Select Bit Description

CS0[2]	CS0[1]	CS0[0]	Description
0	0	0	No clock source (timer/counter stopped)
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	clkI/O/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Κατά την διάρκεια της λειτουργίας και την εκτέλεση του προγράμματος ο συνδυασμός αυτών των bit αλλάζει ανάλογα με την περίσταση.

Καταχωρητής TIMSK0 (TC0 Interrupt Mask Register)

TIMSK0 — **TC0 Interrupt Mask Register**

Bit	7	6	5	4	3	2	1	0
						OCIE0B	OCIE0A	TOIE0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 1 – OCIE0A Timer/Counter0, Output Compare A Match Interrupt Enable

Όταν το bit αυτό πάρει την τιμή "1" και το I-bit (Eneble Global Interrupts) του Status Register είναι "1", είναι ενεργοποιημένο το interrupt που σχετίζεται με την ταύτιση του timer και του καταχωρητή σύγκρισης.

Καταχωρητής TCNT0 (TC0 Counter Value Register)

Είναι ο καταχωρητής που αποθηκεύει την τιμή του ο timer . Όταν είναι 8 bit μπορεί να μετρήσει από το 00000000 μέχρι το 11111111

Καταχωρητής OCR0A (TC0 Output Compare Register A)

Είναι ο καταχωρητής σύγκρισης. Η τιμή σε αυτόν τον καταχωρητή συγκρίνεται συνεχώς με την τιμή στον μετρητή TCNT0.

Καταχωρητής TIFR0 (TC0 Interrupt Flag Register)

TIFR0 — **TC0 Interrupt Flag Register**

Bit	7	6	5	4	3	2	1	0
						OCF0B	OCF0A	TOV0
Access						R/W	R/W	R/W
Reset						0	0	0

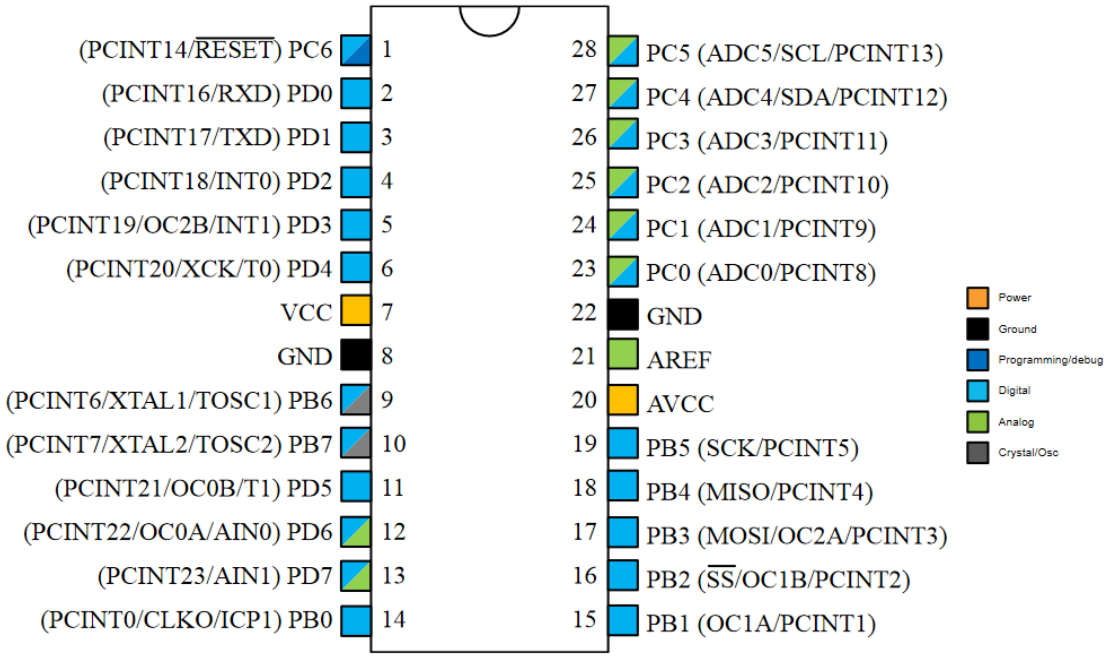
Bit 1 – OCF0A Timer/Counter 0, Output Compare A Match Flag

Το bit αυτό παίρνει την τιμή 1 όταν προκύπτει ταύτιση μεταξύ του μετρητή και του καταχωρητή σύγκρισης OCR0A

Οι ρυθμίσεις που πρέπει να γίνουν είναι παρόμοιές και για τους τρεις καταχωρητές. Η βασική διαφορά είναι στον timer1 όπου είναι 16 bit αντί για 8 bit και προσφέρει κάποιες παραπάνω λειτουργίες.

5.2.5. Χαρτογράφηση Pin m328p

Για την επικοινωνία με το εξωτερικό περιβάλλον ο μικροελεγκτής έχει 28 pin. Το κάθε ένα από αυτά τα pin μπορεί να έχει παραπάνω από μια λειτουργία ανάλογα με τις ρυθμίσεις που έχουν γίνει . Στην εικόνα φαίνονται τα pin του μικροελεγκτή και οι λειτουργίες τους.

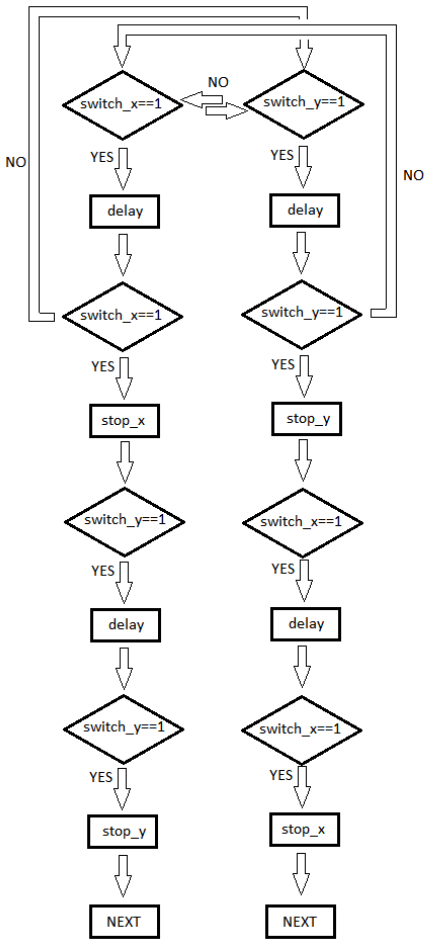


Εικόνα 5.2 - μικροελεγκτής m328p

PD2: Ο ακροδέκτης αυτός συνδέεται με έναν NO-NC διακόπτη (switch X_MSW). Η άλλη επαφή του διακόπτη είναι στα 5V. Κατά την εκκίνηση η κεφαλή οδηγείται στην αρχή των αξόνων (0,0). Όταν ό άξονας x, φθάσει στην αρχή των αξόνων, κλείνει η επαφή του διακόπτη, ο ακροδέκτης έρχεται σε λογικό "1" και ο μικροελεγκτής θέτει τον driver_x εκτός λειτουργίας (PB2→"1").

PD3: Ο ακροδέκτης αυτός συνδέεται με έναν NO-NC διακόπτη (switch Y_MSW). Η λειτουργία του είναι ίδια με τον διακόπτη switch X_MSW με την διαφορά ότι σχετίζεται με τον άξονα y (PB4→"1").

Στην εικόνα 2.13 φαίνεται ο κώδικάς (σε assembly) για τον έλεγχο ενεργοποίησης των τερματικών διακοπτών. Όταν πατηθεί ένας διακόπτης ο κώδικας επανελέγχει την τιμή μετά από μια μικρή καθυστέρηση. Με αυτό το είδος φίλτρου με κώδικα, γίνεται να προβλεφθεί η περίπτωση εσφαλμένου σήματος λόγω παρασίτων. Εναλλακτικά θα μπορούσε να τοποθετηθεί ένας μικρός ξηρός πυκνωτής πριν τον ακροδέκτη, για απόσβεση των παρασίτων. (Παρατηρήθηκε πως η απουσία φίλτρου κάποιες φορές έδινε εντολή στο μοτέρ να σταματήσει παρόλο που ο αντίστοιχος διακόπτης δεν είχε πατηθεί).




```
ORIGIN:
X_AXIS_ORIGIN:
                SBIS    PIND,2            //Skip if bit PD2 (switch_x) is set
                RJMP    Y_AXIS_ORIGIN
                RCALL   Delay
                SBIS    PIND,2            //Skip if bit PD2 (switch_x) is set
                RJMP    Y_AXIS_ORIGIN
                SBI     PORTB,PB2         //Stop x_axis motor (PB2=1)
        Y0:     SBIS    PIND,3           //Skip if bit PD3 (switch_y) is set
                RJMP    Y0
                RCALL   Delay
                SBIS    PIND,3          //Skip if bit PD3 (switch_y) is set
                RJMP    Y0
                SBI     PORTB,PB4       //Stop y_axis motor (PB4=1)

                RCALL   Delay 1s

                RJMP    Initialize_USART

Y_AXIS_ORIGIN:
                SBIS    PIND,3            //Skip if bit PD3 (switch_y) is set
                RJMP    X_AXIS_ORIGIN
                RCALL   Delay
                SBIS    PIND,3           //Skip if bit PD3 (switch_y) is set
                RJMP    X_AXIS_ORIGIN
                SBI     PORTB,PB4        //Stop y_axis motor (PB4=1)
        X0:     SBIS    PIND,2           //Skip if bit PD2 (switch_x) is set
                RJMP    X0
                RCALL   Delay
                SBIS    PIND,2           //Skip if bit PD2 (switch_x) is set
                RJMP    X0
                SBI     PORTB,PB2        //Stop x_axis motor (PB2=1)

                RCALL   Delay_1s

                RJMP    Initialize_USART
```

Εικόνα 5.3 – αλγόριθμός οδήγησης κεφαλής στην αρχή των αξόνων

PD6: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη STEP_X του driver X.Η δημιουργία παλμού σε αυτόν τον ακροδέκτη είναι η οδηγία στον driver_x για τον έλεγχο του μοτέρ στον άξονα x. Ο αριθμός τον παλμών είναι η μετατόπιση και η συχνότητα είναι η ταχύτητα. Για την δημιουργία παλμού χρησιμοποιείται η λειτουργία CTC (**C**lear **T**imer on **C**ompare) του timer/counter 0.

PB0: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη DIR_X του driver X. Όταν είναι σε λογικό 0 ο κινητήρας περιστρέφεται δεξιόστροφα. Σύμφωνα με τον σχεδιασμό, αυτό μεταφράζεται σε μεγαλύτερες τιμές κατά τον άξονα x. Όταν είναι σε λογικό 1 ο κινητήρας περιστρέφεται αριστερόστροφα (μικρότερες τιμές κατά τον x).

PB1: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη SIGNAL του σέρβο μοτέρ. Η έξοδος του ακροδέκτη είναι ένα σήμα PWM ορισμένης συχνότητας για τον έλεγχο του μοτέρ. Για την δημιουργία παλμού χρησιμοποιείται η λειτουργία CTC (**C**lear **T**imer on **C**ompare) του timer/counter 1. Ο κώδικας για τον έλεγχο του σέρβο μοτέρ είναι στο παράρτημα Α5.

Μέσω των Control Register A και B του timer1 ορίζουμε την λειτουργία σε mode 14 (Fats PWM), και τον ακροδέκτη OCR0A να έρχεται σε λογικό '1' όταν ο μετρητής είναι στην έναρξη (bottom) και να μηδενίζεται όταν υπάρχει ταύτιση με τον καταχωρητή σύγκρισης

TCCR1A – **TC1 Control Register A**

Bit	7	6	5	4	3	2	1	0
	COM1	COM1	COM1	COM1			WGM11	WGM10
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Compare Output Mode, Fast PWM

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM1[3:0] = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

Waveform Generation Mode Bit Description

Mode	WGM13	WGM12 (CTC1)[1]	WGM11 (PWM11)[1]	WGM10 (PWM10)[1]	Timer/ Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Θα πρέπει να βρούμε αρχικά την μέγιστη τιμή (TOP) που πρέπει να μετράει ο μετρητής ώστε να δημιουργηθεί παλμός συχνότητας f= 50 Hz (20 ms). Η τιμή αυτή αποθηκεύεται στον καταχωρητή ICR1 και για prescaler N=8 είναι.

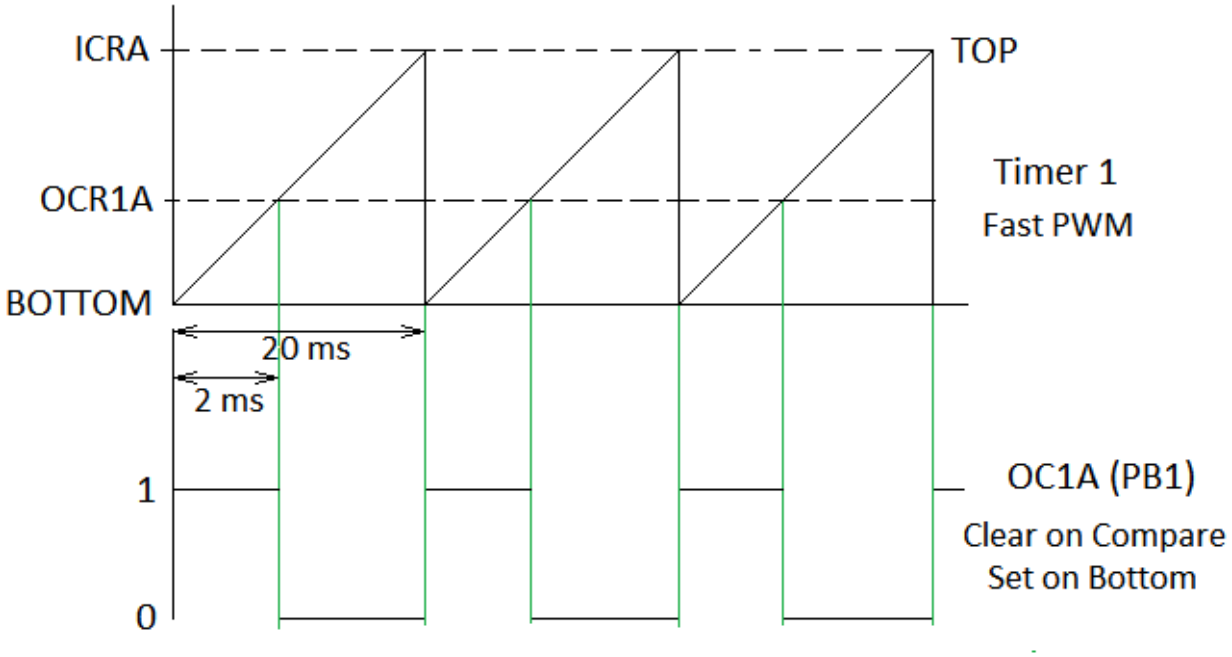
$$f = \frac{f_{clk\_I/O}}{2 * N * (1 + ICR1)} \rightarrow ICR1 = \frac{16{,}000{,}000\ Hz}{2 * 8 * 50\ Hz} - 1 \rightarrow ICR1 = 19{,}999$$

Και την τιμή που θα έχει ο καταχωρητής σύγκρισης $OCR1A_{(0)}$ ώστε το σήμα να έχει διάρκεια (duty cycle) 1,5 ms (f(0)=666,66 Hz) που αντιστοιχεί στην θέση 0^o για το σέρβο-μοτέρ.

$$OCR1A_{(0)} = \frac{16{,}000{,}000\ Hz}{2 * 8 * 666{,}66\ Hz} - 1 \rightarrow ICR1 = 1{,}499$$

Αντίστοιχα την τιμή που θα έχει ο καταχωρητής σύγκρισης $OCR1A_{(90)}$ ώστε το σήμα να έχει διάρκεια (duty cycle) 2 ms (f(0)=500 Hz) που αντιστοιχεί στην θέση 90^o για το σέρβο-μοτέρ. Στην εικόνα 2.15 παριστάνεται η λειτουργία του timer0.

$$OCR1A_{(90)} = \frac{16{,}000{,}000\ Hz}{2 * 8 * 500\ Hz} - 1 \rightarrow ICR1 = 2000$$



Εικόνα 5.4 – Απεικόνιση λειτουργίας timer1 για έλεγχο βηματικού κινητήρα

TCCR1B – **TC1 Control Register B**

Bit	7	6	5	4	3	2	1	0
	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)

PB2: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη EN_X του driver X. Όταν είναι σε λογικό "1" ο driver είναι απενεργοποιημένος, σε λογικό "0" ο driver ενεργοποιείται.

PB3: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη STEP_Y του driver Y. Η δημιουργία παλμού σε αυτόν τον ακροδέκτη είναι η οδηγία στον driver_y για τον έλεγχο του μοτέρ στον άξονα y. Ο αριθμός τον παλμών είναι η μετατόπιση και η συχνότητα είναι η ταχύτητα. Για την δημιουργία παλμού χρησιμοποιείται η λειτουργία CTC (**C**lear **T**imer on **C**ompare) του timer/counter 2.

PB4: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη EN_Y του driver Y. Όταν είναι σε λογικό "1" ο driver είναι απενεργοποιημένος, σε λογικό "0" ο driver ενεργοποιείται.

PB5: Ο ακροδέκτης αυτός συνδέεται με τον ακροδέκτη DIR_Y του driver Y. Όταν είναι σε λογικό 0 ο κινητήρας περιστρέφεται δεξιόστροφα. Σύμφωνα με τον σχεδιασμό αυτό μεταφράζεται σε μεγαλύτερες τιμές κατά τον άξονα y. Όταν είναι σε λογικό 1 ο κινητήρας περιστρέφεται αριστερόστροφα (μικρότερες τιμές κατά τον y).

PC0 / PC1 / PC2: Οι ακροδέκτες αυτοί συνδέονται αντίστοιχα με τους ακροδέκτες M0, M1, M2 του driver Y. Θέτουν το mode λειτουργίας του driver σύμφωνα με τον πίνακα 2.2.

PC3 / PC4 / PC5: Οι ακροδέκτες αυτοί συνδέονται αντίστοιχα με τους ακροδέκτες M2, M1, M0 του driver X. Θέτουν το mode λειτουργίας του driver σύμφωνα με τον πίνακα 2.2.

Στο σχέδιο 2.5 φαίνεται το κύκλωμα όπου απεικονίζονται όλες οι διασυνδέσεις μεταξύ των ηλεκτρονικών στοιχείων. Ο μικροελεγκτής βρίσκεται πάνω στην αναπτυξιακή πλακέτα του arduino. Κάτω αριστερά στο σχέδιο υπάρχει μια χαρτογράφηση των pin του μικροελεγκτή και το πως αντιστοιχίζονται με τα pin του arduino.

5.2.6. Σχέδιο Κυκλώματος

Το σχέδιο 2.5 φαίνεται το κύκλωμα.

Η συνδεσμολογία του μικροελεγκτή με τους Drivers έχει γίνει κατά τέτοιο τρόπο ώστε με μια λέξη στην PORTC να καθορίζονται οι prescaler και των δύο drivers. Και επιπλέον με μία λέξη στην PORTB να ρυθμίζεται, η φορά περιστροφής, η ενεργοποίηση - απενεργοποίηση των moter και ενεργοποίηση - απενεργοποίηση της κεφαλής (Για αυτό τον λόγω την PORTB την ονομάζω byte εκκίνησης).

	PB7(IN)	**PB6(IN)**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**
PORTB	‘1’	’1’	Y DIR	Y EN	Y STP	X EN	HEADER	X DIR

	PC7(IN)	**PC6(IN)**	**PC5**	**PC4**	**PC3**	**PC2**	**PC1**	**PC0**
PORTC	‘1’	’1’	Y MS1	Y MS2	Y MS3	X MS3	X MS2	X MS1

	PD7(IN)	**PD6**	**PD5(IN)**	**PD4(IN)**	**PD3**	**PD2**	**PD1**	**PD0**
PORTD	‘1’	X STP	‘1’	‘1’	Y SW	X SW	TXD	RXD

- Αν dx<0 μέσω της PORTB ο ακροδέκτης ENABLE του driver x παίρνει τιμή '0' για ενεργοποίηση του driver και ο ακροδέκτης DIR παίρνει τιμή '1' για αριστερόστροφη περιστροφή.

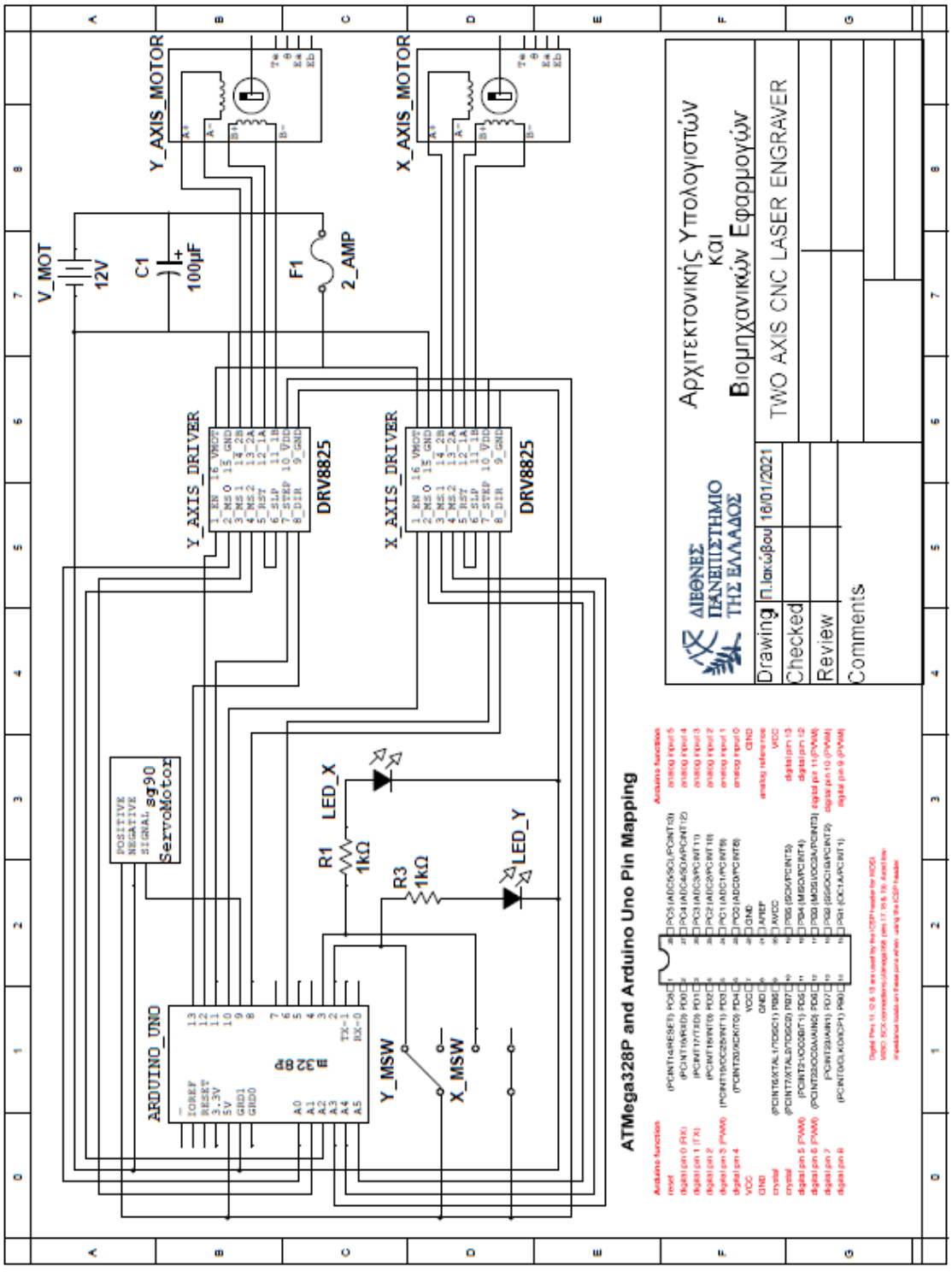
	PB7(IN)	**PB6(IN)**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**
PORTB	‘1’	’1’	Y DIR	Y EN	Y STP	‘0’	HEADER	‘1’

- Αν dx>0 μέσω της PORTB ο ακροδέκτης ENABLE του driver x παίρνει τιμή '0' για ενεργοποίηση του driver και ο ακροδέκτης DIR παίρνει τιμή '0' για αριστερόστροφη περιστροφή.

-Αν dx=0 μέσω της PORTB πάλι ο ακροδέκτης ENABLE γίνεται '1' και σταματάει η περιστροφή.

Με τον υπολογισμό των μετατοπίσεων και των ταχυτήτων ανά άξονα γίνεται η μετάδοση των δεδομένων και η αποθήκευση τους σε καταχωρητές. Τελευταία αποστέλλεται η τιμή της PORTB και τα moter περιστρέφονται αναλόγως.

Συνεπώς με μια δυαδική λέξη των 8-bit (μια επικοινωνία) μπορούμε μέσω της PORTB του μικροελεγκτή να ρυθμίσουμε την ενεργοποίηση-απενεργοποίηση των μοτέρ, την φορά περιστροφής. Και μέσω της PORTC τον prescaler των drivers



Σχέδιο 5.11 - Σχέδιο Κυκλώματος

5.3. Επικοινωνία προγράμματος εφαρμογής με τον m328p

Ο κώδικας G περιέχει την πληροφορία (συντεταγμένες) του σχεδίου που θέλουμε να επεξεργαστούμε. Η πληροφορία αυτή πρέπει να μεταφερθεί στον μικροελεγκτή. Θα χρησιμοποιηθεί UART επικοινωνία για την μεταφορά της πληροφορίας.

Κατά την εκκίνησης του CNC η κεφαλή οδηγείται στην αρχή των αξόνων (0,0). Στην συνέχεια γίνεται αρχικοποίηση της UART επικοινωνίας και το CNC αναμένει την οδηγία για να μεταφέρει την κεφαλή στο επιθυμητό σημείο.

5.3.1. UART επικοινωνία στον m328p

Αρχικοποίηση Επικοινωνίας

Για να υλοποιηθεί η επικοινωνία πρέπει να γίνει αρχικοποίηση της επικοινωνίας. Η αρχικοποίηση περιλαμβάνει τον ορισμό του ρυθμού μετάδοσης, του format του πλαισίου και την ενεργοποίηση πομπού και δέκτη.

Παρακάτω φαίνεται η ρουτίνα αρχικοποίησης της επικοινωνίας UART στον μικροελεγκτή

```
;Initialize_UART (8 bit data / no parity bit / 1 stop bit)
Initialize_UART:
          //set baud rate = 57600 bit/sec
          LDI   TEMP,16000000/16/57600-1
          STS   UBRR0L,TEMP
          //enable receiver and transmiter
          LDI   TEMP,1<<RXEN0 | 1<<TXEN0
          STS   UCSR0B,TEMP
          //8-bit communication
          LDI   TEMP,1<<UCSZ01 | 1<<UCSZ00
          STS   UCSR0C,TEMP
```

Οι καταχωρητές που χρησιμοποιήθηκαν:

UBRR0 – USART Baund Rate 0 Register

Είναι 12 bit και χρησιμοποιείται για το ορισμό του ρυθμού μετάδωσης (BAUD) της USART επικοινωνίας σήμφωνα με την σχέση.

$$BAUD = \frac{f_{osc}}{2(UBRRn + 1)}$$

UDR0 - USART I/O Data Register 0

Είναι ο ενδιάμεσος καταχωρητής (buffer) για την αποθήκευση των δεδομένων μετάδοσης και οι ενδιάμεσοι καταχωρητές για αποθήκευση των δεδομένων λήψης της επικοινωνίας «ακούν» στην ίδια διεύθυνση UDR0.

Ο ενδιάμεσος καταχωρητής μετάδοσης (TXB) είναι ο προορισμός των δεδομένων που έχουν εγγραφεί στον UDR0. Ενώ η ανάγνωση των δεδομένων του UDR0 επιστρέφει το περιεχόμενο του ενδιάμεσου καταχωρητή λήψης (RXB).

Όταν ο αριθμός των δεδομένων για μεταφορά έχει οριστεί μικρότερος του 8 (5-7) τότε τα μη χρησιμοποιημένα bit μεγαλύτερης αξίας αγνοούνται από τον εκπομπό ή γίνονται ‘0’ από τον δέκτη.

Στον buffer μετάδοσης μπορεί να γίνει εγγραφή δεδομένων όταν η σημαία UDRE0 στον καταχωρητή UCSR0A να είναι σε λογικό ‘1’. Αν το bit UDRE0 είναι σε λογικό ‘0’ οποιαδήποτε προσπάθεια για εγγραφή αγνοείται. Αν υπάρχουν δεδομένα στον UDR0 και είναι ενεργοποιημένη η μετάδοση δεδομένων, γίνεται μεταφορά των δεδομένων στον Shift Register (με την προϋπόθεση ότι είναι άδειος). Μετά από αυτό τα δεδομένα μεταφέρονται σειριακά στο TxD0 pin.

Ο buffer λήψης είναι δύο επιπέδων δομής FIFO.

UCSR0A - USART Control and Status Register 0 A

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC0: USART Receive Complete. Είναι ΄1΄ όταν υπάρχουν μη αναγνωσμένα δεδομένα στον buffer και ‘0’ όταν ο buffer είναι άδειος.

Bit 6 – TXC0: USART Transmit Complete: Είναι ‘1’ όταν ένα ολόκληρο πλαίσιο φεύγει από τον shit register και δεν υπάρχουν νέα δεδομένα στον buffer.

Bit 5 – UDRE0: USART Data Register Empty: Αν είναι ‘1’ ο buffer μετάδοσης είναι άδειος και άρα έτοιμος για να δεχτεί νέα δεδομένα.

UCSR0B - USART Control and Status Register 0 B

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 4 – RXEN0: Receiver Enable: Ο δέκτης USART ενεργοποιείται όταν το bit αυτό πάρει τιμή ΄1΄.

Bit 3 – TXEN0: Transmitter Enable: Ο πομπός USART ενεργοποιείται όταν αυτό το bit γίνει ‘1’.

Bit 2 – UCSZ02: Character Size: Σε συνδιασμό με το UCSZ0[1:0] bit του UCSR0C ορίζει το μέγεθος των bit των δεδομένων στο πλαίσιο.

UCSR0C - USART Control and Status Register 0 C

Bit	7	6	5	4	3	2	1	0
	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 / UDORD0	UCSZ00 / UCPHA0	UCPOL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bits 7:6 – UMSEL0n: USART Mode Select: Ορίζει το mode λειτουργίας της USART επικοινωνίας σύμφωνα με τον παρακάτω πίνακα:

USART Mode Selection

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

Bits 5:4 – UPM0n: USART Parity Mode: Ενεργοποιεί και ορίζει το είδος του ελέγχου ισοτιμίας σύμφωνα με τον παρακάτω πίνακα:

USART Mode Selection

UPM0[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

Bit 3 – USBS0: USART Stop Bit Select 0: Ορίζει τον αριθμό τον bit τερματισμού.

Stop Bit Settings

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

Bit 2 – UCSZ01 / UDORD0: USART Character Size / Data Order: Σε συνδυασμό με UCSZ02 bit in UCSR0B ορίζει το μέγεθος των bit των δεδομένων σε ένα πλαίσιο σύμφωνα με τον παρακάτω πίνακα.

Character Size Settings

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

Πομπός:

Παρακάτω φαίνεται η συνάρτηση που χρησιμοποιήθηκε για την μετάδοση δεδομένων στο πρόγραμμα εφαρμογής:

```
Transmit:
          LDS   TEMP, UCSR0A  //copy UCSR0A to TEMP
          SBRS  TEMP, UDRE0  //if UDRE0='1' skip next instr.
          RJMP  Transmit     //loop until UDRE0='1'
          STS   UDR0,'Data to be sent' //copy data to UDR0
          RET
```

Η συνάρτηση περιμένει να αδειάσει ο buffer του πομπού ελέγχοντας την UDRE0 σημαία. Όταν η σημαία γίνει '1' ο buffer φορτώνεται με νέα δεδομένα.

Δέκτης:

Παρακάτω φαίνεται η συνάρτηση που χρησιμοποιήθηκε για την λήψη δεδομένων από το πρόγραμμα εφαρμογής:

```
Receive:
          LDS   TEMP,UCSR0A
          SBRS  TEMP,RXC0   //if RXCO='1' skip next instr.
          RJMP  Receive    // loop until buffer empty
          LDS   TEMP,UDR0  // take data from buffer
          RET
```

Η συνάρτηση περιμένει δεδομένα να εμφανιστούν στον buffer του δέκτη ελέγχοντας την RXC σημαία. Όταν η σημαία γίνει ‘1’ διαβάζονται τα δεδομένα από τον buffer.

5.3.2. UART επικοινωνία στο πρόγραμμα εφαρμογής [3]

Διαμόρφωση σειριακής θύρας

Η διαμόρφωση περιλαμβάνει τα εξής βήματα:

1. Συνάρτηση CreateFile() για το άνοιγμα μια νέας σειριακής θύρας με τις προκαθορισμένες ρυθμίσεις.

2. Αρχικοποίηση του DCBLength μέλους της DCB δομής για το μέγεθος της δομής. Η αρχικοποίηση χρειάζεται πριν το πρόγραμμα περάσει το DCBLength ως μεταβλητή σε κάθε συνάρτηση.

3. Συνάρτηση GetCommState() για ανάκτηση των προκαθορισμένων ρυθμίσεων της CreateFile().

4. Ανάθεση τιμών στα μέλη της DCB δομής.

5. Συνάρτηση SetCommState() για ορισμό νεών ρυθμίσεων σειριακής θύρας.

```
SerialPort::SerialPort(const char *portName)
{
    this->connected = false;

    this->handler = CreateFileA(static_cast<LPCSTR>(portName),        // pointer to name of the file
                                GENERIC_READ | GENERIC_WRITE,         // Specify mode that open device.
                                0,                                   // the devide isn't shared.
                                NULL,                               // the object gets a default security.
                                OPEN_EXISTING,                     // Specify which action to take on file.
                                FILE_ATTRIBUTE_NORMAL,            // Port attributes
                                NULL);                           // Handle to port with attribute to copy
    // If it fails to open the port, return error.
    if (this->handler == INVALID_HANDLE_VALUE)
    {
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
        {
            std::cerr << "ERROR: Handle was not attached.Reason : " << portName << " not available\n";
        }
        else
        {
            std::cerr << "ERROR!!!\n";
        }
    }
    else
    {
        DCB dcbSerialParameters = {0};

        if (!GetCommState(this->handler, &dcbSerialParameters))  // Get the default port setting information
        {
            std::cerr << "Failed to get current serial parameters\n";
        }
        else
        {
            //  dcbSerialParameters.BaudRate = CBR_9600;
            dcbSerialParameters.BaudRate = CBR_57600;    //  baud rate
            dcbSerialParameters.ByteSize = 8;            // Number of bits/bytes, 4-8
            dcbSerialParameters.StopBits = ONESTOPBIT;  //0,1,2 = 1, 1.5, 2
            dcbSerialParameters.Parity = NOPARITY;      // 0-4=no,odd,even,mark,space
            dcbSerialParameters.fDtrControl = DTR_CONTROL_ENABLE;
            // Configure the port according to the specifications of the DCB structure.
            if (!SetCommState(handler, &dcbSerialParameters))
            {
                std::cout << "ALERT: could not set serial port parameters\n";
            }
            else
            {
                this->connected = true;
                PurgeComm(this->handler, PURGE_RXCLEAR | PURGE_TXCLEAR);
                Sleep(ARDUINO_WAIT_TIME);
            }
        }
    }
}
```

Εικόνα 5.5 – Αρχικοποίηση σειριακής θύρας στα Windows8

Η λαβή (handler) hPort είναι μια παράμετρος για το άνοιγμα της σειριακής θύρας. Είναι ένας μοναδικός αριθμός της νέας θύρας και όλες οι ρυθμίσεις της νέας σειριακή θύρας βασίζονται σε αυτή τη λαβή.

Εγγραφή δεδομένων στην σειριακή θύρα

Η συνάρτηση WriteFile() χρησιμοποιείται για πρόσβαση στην θύρα που έχει ανοιχτεί με την CreateFile() και την αποστολή νέων δεδομένων στην θύρα. Πιο συγκεκριμένα αποστολή των δεδομένων σε μια θέση μνήμης στον buffer της UART και μετά μεταφορά στην θύρα. Τα ορίσματα της συνάρτησης WriteFile() είναι:

1. Η λαβή (handler) που δημιουργήθηκε από την CreateFile() κατά το άνοιγμα της θύρας.

2. Δείκτης σε θέση μνήμης (buffer) που πρόκειται να εγγραφούν τα δεδομένα. Πρόκειται για δεδομένα σε δυαδική μορφή ή πίνακας χαρακτήρων.

3. Ο αριθμός των byte που πρόκειται να εγγραφούν.

4. Μεταβλητή για αποθήκευση του αριθμού των byte που πραγματικά εγγράφηκαν.

5. Δείκτης σε μια δομή δεδομένων σχετική με σύγχρονη επικοινωνία. Σε περίπτωση ασύγχρονης επικοινωνίας έχει τιμή NULL

```
// Sending provided buffer to serial port;  // SEND BYTE

bool SerialPort::writeSerialPort(byte *buffer, unsigned int buf_size)
{
    DWORD bytesSend;


    if (!WriteFile(this->handler, (void*)  buffer, 1, &bytesSend, 0))
    {
        ClearCommError(this->handler, &this->errors, &this->status);
        return false;
    }

    return true;
}
```

Εικόνα 5.6 – Εγγραφή δεδομένων σε σειριακή θύρα στα Windows8

Διάβασμα δεδομένων από σειριακή θύρα

Το διάβασμα εκτελείται από ένα νήμα έτοιμο να επεξεργαστεί δεδομένα που καταφθάνουν στον buffer. Ένα σήμα ενημερώνει την διεργασία για δεδομένα στην σειριακή θύρα. Η διεργασία διαβάζει ένα byte κάθε φορά. Μια ReadFile() συνάρτηση για κάθε

byte που είναι να διαβαστεί μέχρι να διαβαστούν όλα τα byte. Αφού ολοκληρωθεί το διάβασμα δεδομένων, η διεργασία αναμένει για νέο σήμα.

1. Η λαβή (handler) που δημιουργήθηκε από την CreateFile() κατά το άνοιγμα της θύρας.

2. Δείκτης σε θέση μνήμης (buffer) που πρόκειται να διαβαστοπύν τα δεδομένα. Πρόκειται για δεδομένα σε δυαδική μορφή ή πίνακας χαρακτήρων.

3. Ο αριθμός των byte που πρόκειται να διαβαστούν.

4. Μεταβλητή για αποθήκευση του αριθμού των byte που πραγματικά διαβάστηκαν.

5. Δείκτης σε μια δομή δεδομένων σχετική με σύγχρονη επικοινωνία. Σε περίπτωση ασύγχρονης επικοινωνίας έχει τιμή NULL

```
int SerialPort::readSerialPort(byte *buffer, unsigned int buf_size)
{
    DWORD bytesRead{};
    unsigned int toRead = 0;

    ClearCommError(this->handler, &this->errors, &this->status);

    if (this->status.cbInQue > 0)
    {
        if (this->status.cbInQue > buf_size)
        {
            toRead = buf_size;
        }
        else
        {
            toRead = this->status.cbInQue;
        }
    }

    memset((void*) buffer, 0, buf_size);

    if (ReadFile(this->handler, (void*) buffer, toRead, &bytesRead, NULL))
    {
        return bytesRead;
    }

    return 0;
}
```

Εικόνα 5.7 – Διάβασμα δεδομένων από σειριακή θύρα στα Windows8

5.4 Ανάπτυξη Αλγορίθμου γραμμικής και κυκλικής παρεμβολής

Το πρόγραμμα εφαρμογής διαβάζει τον κώδικα G από το αρχείο κειμένου .nc,και ανάλογα με την εντολή στέλνει στον μικροελεγκτή έναν αναγνωριστικό κωδικό των 8 bit. Στην συνέχεια καλείται η κατάλληλη συνάρτηση για την εκτέλεση του αλγόριθμου παρεμβολής (εξισώσεις επίλυσης ευθύγραμμης ομαλής κίνησης). Ο μικροελεγκτής χρησιμοποιεί τον αναγνωριστικό κωδικό για να καλέσει την ρουτίνα που θα εξυπηρετήσει τον αλγόριθμο παρεμβολής.

Εντολή	8-bit δυαδικός κωδικός	Αλγόριθμος Παρεμβολής	Ρουτίνα Εξυπηρέτησης Μικροελεγκτή
G00	'00000000'	Ευθύγραμμη κίνηση με μέγιστη ταχύτητα	Ευθύγραμμη κίνηση με μέγιστη ταχύτητα
G01	'00000001'	Ευθύγραμμη κίνηση με ταχύτητα κατεργασίας	Ευθύγραμμη κίνηση με ταχύτητα κατεργασίας
G02	'00000010'	Δεξιόστροφο Τόξο	Κυκλική Κίνηση
G03	'00000011'	Αριστερόστροφο Τόξο	
M30	'00111011'	Τερματισμός	Τερματισμός

Πίνακας 5.1 – Δυαδικός κωδικός Εντολών δρομολόγησης κώδικα G

Ο αλγόριθμος παρεμβολής επεξεργάζεται τα δεδομένα και δημιουργεί λέξεις του ενός byte για την μετατόπιση και την ταχύτητα. Οι λέξεις αυτές στέλνονται στον μικροελεγκτή και αποθηκεύονται σε συγκεκριμένους καταχωρητές που χρησιμοποιούνται από την ρουτίνα εξυπηρέτησης. Χρησιμοποιούνται οι εξής αλγόριθμοι παρεμβολής.

5.4.1. Ευθύγραμμη κίνηση με μέγιστη ταχύτητα στους δύο άξονες, Εντολή -'G00', (Παράρτημα Α2)

Η προσέγγιση που γίνεται είναι μετακίνηση με μέγιστη ταχύτητα (Vmax) και στους δύο άξονες. Ο άξονας με την μικρότερη μετατόπιση σταματάει πρώτος και ακολουθεί ο άξονας με την μεγαλύτερη μετατόπιση.

Η μετατόπιση ανά άξονα δίνεται από την σχέση:

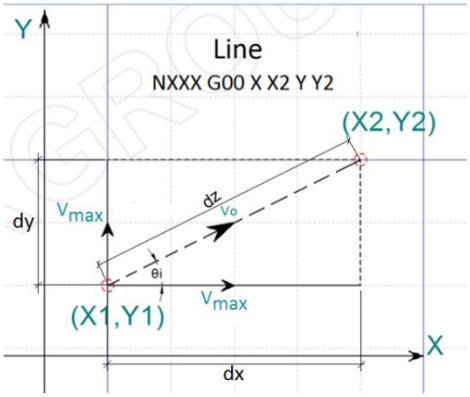
$$dx = x2 - x1 \ (4.71\ a)$$

$$dy = y2 - y1 \ (4.71\ \beta)$$

οι τιμές των dx και dy είναι σε mm και πρέπει να μετατραπούν σε steps σύμφωνα με την σχέση:

$$stepx = \frac{|dx|}{microstep} \ (4.72\ \alpha)$$

$$stepy = \frac{|dy|}{microstep} (4.72\ \beta)$$



Εικόνα 5.8 – Ευθύγραμμη κίνηση χωρίς κατεργασία

Η ταχύτητα μετατρέπεται στο δυαδικό σύμφωνα με την σχέση:

$$OCRnA = \frac{f}{2N(\frac{v_{max}}{micro\_step})} - 1 \quad (4.73)$$

Οι τιμές για την μετατόπιση dx και dy και οι αντίστοιχες ταχύτητες OCR0A και 0CR2A αποθηκεύονται σε καθορισμένους καταχωρητές στον μικροελεγκτή. Τελευταίο στέλνεται το byte εκκίνησης στην PORTB, που καθορίζει την φορά περιστροφής, την απενεργοποίηση της κεφαλής και την ενεργοποίηση των drivers.

Ρουτίνα εξυπηρέτησης εντολής G00 στον μικροελεγκτή

Η αποστολή του byte εκκίνησης στην PORTB προκαλεί την έναρξη της κίνησης. Ο μικροελεγκτής διαχειρίζεται με διαφορετικό τρόπο τον έλεγχο της θέσης στον άξονα x σε σχέση με τον άξονα y.

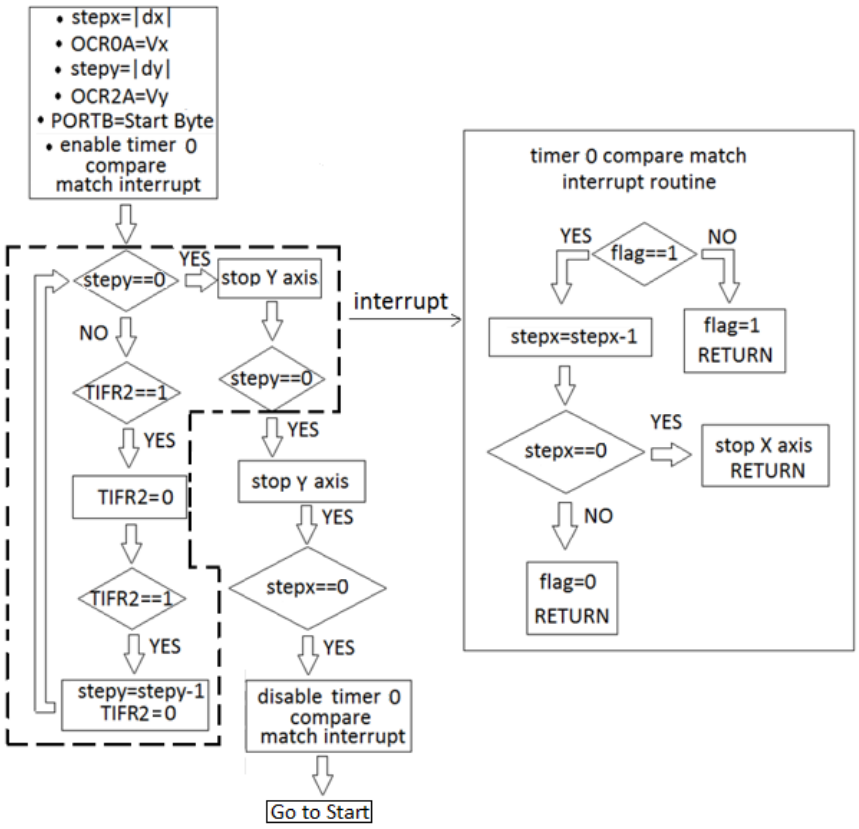
Έλεγχος θέση κατά y:

Κάθε φορά που ο timer 2 ταυτιστεί με την τιμή στον καταχωρητή σύγκρισης OCR0A η σημαία TIFR2 στον OCF2A γίνεται 1. Ένας βρόγχος επανάληψης ελέγχει συνεχώς την σημαία και σε κάθε δεύτερη ταύτιση ο μετρητής stepy μειώνεται κατά 1. Ο έλεγχος επαναλαμβάνεται και όταν ο μετρητής stepy γίνει μηδέν σταματάει η κίνηση στο άξονα y (ακροδέκτης Enable του driver→ '1').

Ο έλεγχος θέσης κατά y διακόπτεται προσωρινά από το interrupt ταύτισης του timer0 με τον 0CR0A το οποίο καλεί την ρουτίνα για έλεγχο θέσης κατά y.

Έλεγχος θέση κατά y:

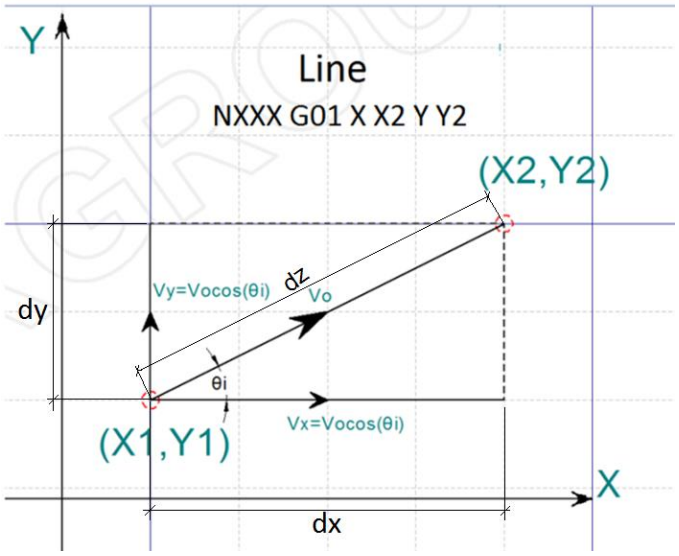
Κάθε φορά που ο timer 0 ταυτιστεί με την τιμή OCR0A ενεργοποιείται ένα interrupt το οποίο ελέγχει μια προκαθορισμένη σημαία. Αν η σημαία είναι '0' την κάνει ένα και επιστρέφει στο πρόγραμμα (έλεγχος θέσης κατά y). Αν η σημαία είναι '1' μειώνει τον μετρητή step x και επιστρέφει στο πρόγραμμα (έλεγχος θέσης κατά y). Όταν ο μετρητής γίνει μηδέν σταματάει η κίνηση στον άξονα y.



Εικόνα 5.9 – Ρουτίνα εξυπηρέτησης εντολής G00

5.4.2. Ευθύγραμμη κίνηση με ταχύτητα κατεργασίας - Εντολή 'G01'. Παράρτημα Α3

Η κίνηση και στους δύο άξονες είναι ευθύγραμμη ομαλή (σταθερή ταχύτητα). Το μέτρο της ταχύτητας του κάθε άξονα υπολογίζεται ώστε να προκύπτει συνισταμένη κίνηση με ταχύτητα ίση με την επιθυμητή ταχύτητα κατεργασίας



Εικόνα 5.10 – Ευθύγραμμη κίνηση με ταχύτητα κατεργα-

Υπολογισμός σχετικής μετατόπισης ανά άξονα όπως στην 7.71:

$$dx = x2 - x1 \ (4.71\ \alpha)$$

$$dy = y2 - y1 \ (4.71\ \beta)$$

και από το πυθαγόρειο θεώρημα η συνισταμένη μετατόπιση είναι:

$$dz = \sqrt{dx^2 + dy^2} \ (4.74)$$

Προκειμένου :

1. να ελαχιστοποιήσω την επικοινωνία μεταξύ προγράμματος εφαρμογής και μικροελεγκτή και

2. Να περιορίσω το χρονικό διάστημα που είναι απασχολημένος ο μικροελεγκτής,

ως μετρητή θα χρησιμοποιήσω μόνο την μετατόπιση στον έναν άξονα (στον x). Εφόσον η κίνηση είναι ευθύγραμμη ομαλή αυτό που χρειάζομαι είναι οι σωστές ταχύτητες στον κάθε άξονα. Όταν ολοκληρωθεί η μετατόπιση στον έναν άξονα αρκεί να σταματήσω την κίνηση και στους δύο άξονες και εφόσον οι ταχύτητες έχουν σωστή τιμή θα έχω φθάσει στο σωστό σημείο (X2,Y2).Την μετατόπιση dy στην σχέση 4.71 β την χρειάζομαι για να υπολογίσω την ταχύτητα στον y.[1]

Δεδομένου ότι η κίνηση είναι ευθύγραμμη ομαλή κάθε χρονική στιγμή t ισχύει:

$$dz = v_0 \cdot dt \rightarrow \ t = \frac{dz}{v_0} \ (4.75) \text{ και}$$

$$dx = v_x \cdot t \rightarrow \ v_x = \frac{dx}{t} \overset{4.}{\rightarrow} \ v_x = \frac{dx}{dz} v_o \ (4.76\ \alpha)$$

$$dy = v_y \cdot t \rightarrow \ v_y = \frac{dy}{t} \overset{4.}{\rightarrow} \ v_y = \frac{dy}{dz} v_o \ (4.76\ \beta)$$

οι τιμή των dx είναι σε mm και μετατρέπεται σε steps σύμφωνα με την σχέση 4.72α. Μετατροπή της ταχύτητας vx και vy σε δυαδικό σύμφωνα με τις σχέσεις:

$$OCR0A = \frac{f}{2N(\frac{v_x}{micro\_step})} - 1 \ (4.77\ \alpha)$$

$$OCR2A = \frac{f}{2N(\frac{v_y}{micro\_step})} - 1 \ (4.77\ \beta)$$

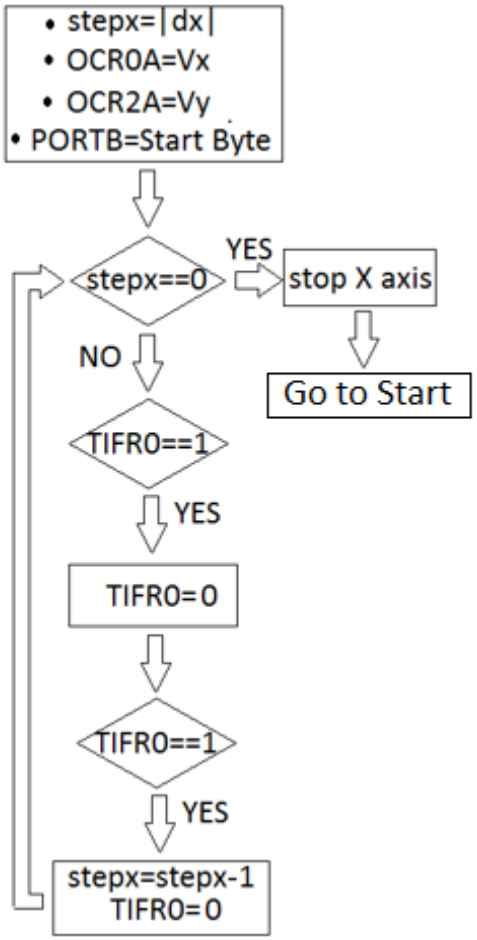
Η τιμή στον καταχωρητή OCR0A(8 - bit)ρυθμίζει την συχνότητα εναλλαγής του σήματος στον ακροδέκτη STEP του driver. Η σχέση είναι αντιστρόφως ανάλογη, μεγάλη τιμή σημαίνει μικρή ταχύτητα και μικρή τιμή σημαίνει μεγάλη ταχύτητα.

[1] Η περίπτωση που περιγράφηκε παραπάνω για χρήση μόνο της μετατόπιση στον έναν άξονα δημιουργεί πρόβλημα όταν ή μετατόπιση στον άξονα x είναι μηδέν διότι dx=0 →vx=0→OCR0A=0. Για να ξεπεραστεί αυτό το πρόβλημα όταν dx=0 τότε stepx=stepy και vx=vy.

Οι τιμές για την μετατόπιση dx και οι ταχύτητες OCR0A(κατά x) και 0CR2A (κατά y) αποθηκεύονται σε συγκεκριμένους καταχωρητές στον μικροελεγκτή. Τελευταίο στέλνεται το byte εκκίνησης στην PORTB, που καθορίζει την φορά περιστροφής, την ενεργοποίηση της κεφαλής και την ενεργοποίηση των drivers.

Ρουτίνα εξυπηρέτησης εντολής G01 στον μικροελεγκτή

Η αποστολή του byte εκκίνησης στην PORTB προκαλεί την έναρξη της κίνησης. Ο μικροελεγκτής ελέγχει μόνο την μετατόπιση κατά x, με τον ίδιο τρόπο που γίνεται στην εντολή G00. Όταν ο μετρητής θέσης κατά x μηδενιστεί τερματίζεται η κίνηση και στους δύο άξονες.



Εικόνα 5.11 – Ρουτίνα εξυπηρέτησης εντολής G01

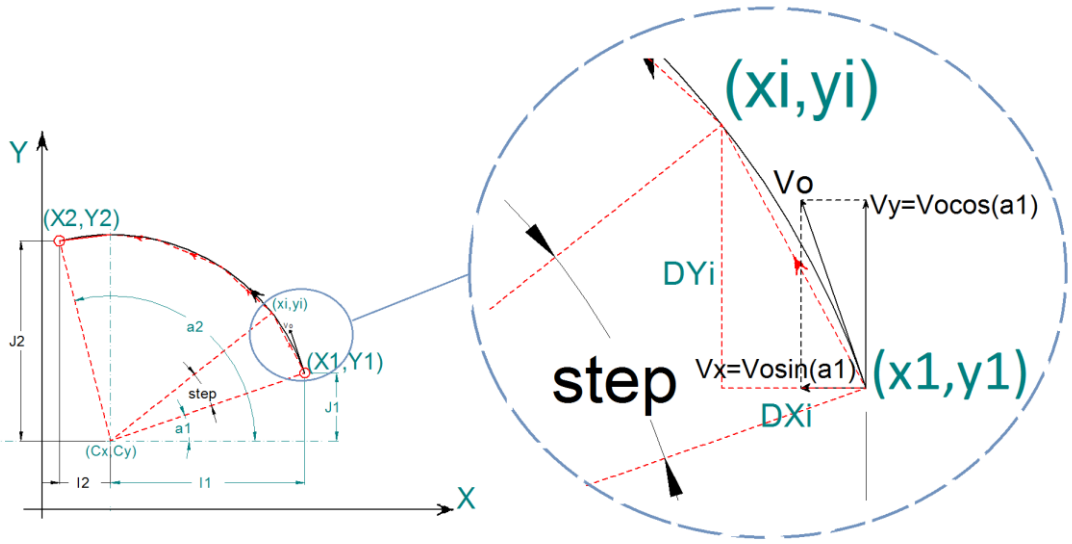
5.4.3. Αριστερόστροφο τόξο με ταχύτητα κατεργασίας - Εντολή 'G03'. **Παράρτημα Α4**

Το τόξο προσεγγίζεται ως τμήμα πολυγώνου εγγεγραμμένου σε αυτό. Κατά αυτό τον τρόπο η κίνηση αναλύεται σε πολύ μικρά ευθύγραμμα τμήματα (πλευρές του πολυγώνου). Η κίνηση για την διαγραφή του κάθε τμήματος είναι ευθύγραμμη ομαλή και συνεπώς οι εξισώσεις που την ικανοποιούν είναι ίδιες όπως περιγράφονται στην 4.3.2.

Οι επιπλέον υπολογισμοί που πρέπει να εκτελέσει ο αλγόριθμος στην περίπτωση αυτή είναι ο διαχωρισμός του τόξου σε μικρά ευθύγραμμα τμήματα. Για το σκοπό αυτό γίνεται επιλογή μιας γωνίας α που θα χρησιμοποιηθεί ως βήμα για την εύρεση του επόμενου σημείου.

Όσο μικρότερο το α , τόσο μικρότερο το ευθύγραμμο τμήμα και άρα τόσο μικρότερο το σφάλμα χορδής και συνεπώς το πολύγωνο προσεγγίζει καλύτερα το τόξο. Για μεγαλύτερο α το σφάλμα χορδής αυξάνεται και μεγαλώνει η απόκλιση από το πραγματικό τόξο, και το αποτέλεσμα προσεγγίζει οπτικά το πολύγωνο.

Συνεπώς θέλουμε όσο το δυνατόν μικρότερο α. Το πόσο μικρό θα είναι το α εξαρτάται από τον χρόνο που απαιτείται προκειμένου να γίνει η επίλυση των εξισώσεων και η μεταφορά των δεδομένων.



Εικόνα 5.12 – Κυκλική αριστερόστροφη κίνηση G03

Αρχικά γίνεται ο υπολογισμός της ακτίνας του τόξου από την σχέση:

$$R = \sqrt{I_1^2 + J_1^2} (4.78)$$

Υπολογίζονται οι συντεταγμένες του κέντρου του τόξου:

$$C_x = X_1 + I_1 \ (4.79\ \alpha)$$

$$C_y = Y_1 + J_1 \ (4.79\ \beta)$$

Και οι συντεταγμένες του τελικού σημείου σε σχέση με το κέντρο του τόξου:

$$I_2 = C_X - X_2 \ (4.80\ \alpha)$$

$$J_2 = C_Y - Y_2 \ (4.80\ \beta)$$

Με τα παραπάνω δεδομένα μπορούμε να υπολογίσουμε τις γωνίες που σχηματίζει το αρχικό και τελικό σημείο με το οριζόντιο άξονα (X):

$$\alpha_1 = atan\left[\left(\frac{J_1}{I_1}\right) * \frac{180}{\pi}\right] \ (4.81\ \alpha\ )$$

$$\alpha_2 = atan\left[\left(\frac{J_2}{I_2}\right) * \frac{180}{\pi}\right] \ (4.81\ \beta)$$

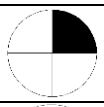
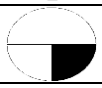
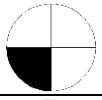
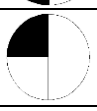
Σε κάθε επανάληψη η γωνία αυξάνεται α1=α1+stepangle. Οι συντεταγμένες του επόμενου σημείου i που αντιστοιχεί σε γωνία ίση με +1 step angle μπορούν να βρεθούν από την σχέση:

$$x_i = cos\left[\frac{a1 \cdot \pi}{180}\right] R \ (4.82\ \alpha\ )$$

$$y_i = sin\left[\frac{a1 \cdot \pi}{180}\right] R (4.82\ \beta)$$

Με τον τρόπο αυτό έχουμε υπολογίζει αρχικό και τελικό σημείο και οι εξισώσεις κίνησης είναι όπως στην 4.3.2.

Η τιμή των bit που καθορίζουν την φορά περιστροφής στο byte εκκίνησης εξαρτάται από το τεταρτημόριο και παίρνει τιμές από τον παρακάτω πίνακα:

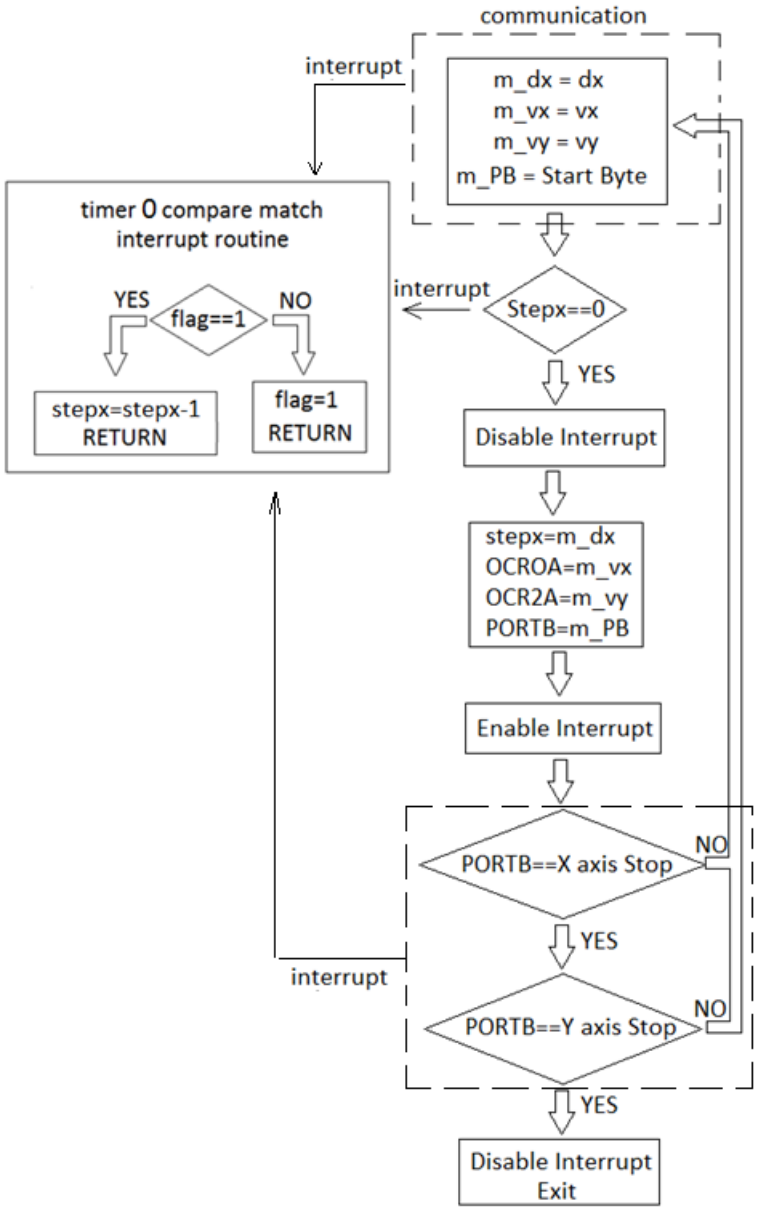
Φορά κίνησης ανάλογα με το τεταρτημόριο για αριστερόστροφο τόξο (G03)			
Τεταρτημόριο Κύκλου	Γωνία α1	X	Y
	$0^o<\alpha\leq 90^o$	-	+
	$90^o<\alpha\leq 180^o$	-	-
	$180^o<\alpha\leq 270^o$	+	-
	$270^o<\alpha\leq 360^o$	+	+

Πίνακας 5.2 – Κατεύθυνση κίνησης αξόνων ανάλογα με το τεταρτημόριο για G03

Ο αλγόριθμος επαναλαμβάνεται όσο ικανοποιείται η σχέση α1≤α2. Όταν γίνει α1>α2 τα bit του byte εκκίνησης απενεργοποιούν τους drivers (λογικό '1').

Ρουτίνα εξυπηρέτησης εντολής G03 στον μικροελεγκτή

Προκειμένου να πετύχουμε όσο το δυνατόν μικρότερο step angle,ο έλεγχος θέσης εξελίσσεται παράλληλα με την μεταφορά των δεδομένων για την επόμενη κίνηση. Τα δεδομένα για το πρώτο ευθύγραμμο τμήμα του πολυγώνου μεταφέρονται στους καταχωρητές και στην συνέχεια εκτελείται ο παρακάτω βρόγχος.



Εικόνα 5.13 – Ρουτίνα εξυπηρέτησης εντολής G03

Τα αποτελέσματα του αλγορίθμου (μετατόπιση στον x (dx), ταχύτητα στον x (vx), ταχύτητα στον y (vy) και byte εκκίνησης (PORTB)) στέλνονται στον μικροελεγκτή και αποθηκεύονται σε θέσεις μνήμης και όχι στους καταχωρητές που χρησιμοποιούνται από την ρουτίνα εξυπηρέτησης.

Όταν ο timer0 ταυτιστεί με την τιμή σύγκρισης OCR0A ενεργοποιείται ένα interrupt το οποίο καλεί την ρουτίνα μείωσης μετατόπισης. Η ρουτίνα ελέγχει μια σημαία και σε κάθε δεύτερη ταύτιση μειώνει τον μετρητή θέση κατά 1.

Ο έλεγχος θέσης εξελίσσεται παράλληλα με την μεταφορά δεδομένων που είναι σχετικά χρονοβόρα. Όταν ολοκληρωθεί η μετατόπιση απενεργοποιείται το interrupt (το step x δεν μειώνεται) και τα δεδομένα για την επόμενη μετακίνηση μεταφέρονται στους κατάλληλους καταχωρητές.

Η μεταφορά αυτή γίνεται πολύ γρήγορα και τα interrupt απενεργοποιούνται.

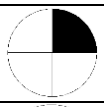
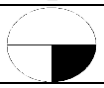
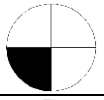
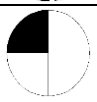
Σημείωση! (Η απενεργοποίηση αυτή ίσως προκαλεί μικρή απώλεια θέσης, καθώς η κίνηση συνεχίζεται)

Θα πρέπει σε κάθε περίπτωση η επικοινωνία να προλάβει να ολοκληρωθεί προτού ολοκληρωθεί η μετατόπιση. Όταν ολοκληρωθεί η μετατόπιση γίνεται αντιγραφή των παραμέτρων από την θέση μνήμης στους καταχωρητές της ρουτίνας εξυπηρέτησης και η κίνηση συνεχίζεται με τα νέα δεδομένα.

Πριν την έναρξη κάθε ευθύγραμμου τμήματος γίνεται έλεγχος στην τιμή του byte εκκίνησης μέσω της PORTB και αν οι τιμές των bit ενεργοποίησης των drivers είναι '1' (driver απενεργοποιημένοι) ο βρόγχος επανάληψης τερματίζεται.

5.4.4. Δεξιόστροφο τόξο με ταχύτητα κατεργασίας - Εντολή 'G02'

Είναι η ίδια φιλοσοφία με το αριστερόστροφο τόξο με την διαφορά ότι ο βρόγχος επαναλαμβάνεται όσο ισχύει η σχέση $\alpha 2<\alpha 1$. Και η τιμή των bit που καθορίζουν την φορά περιστροφής στο byte εκκίνησης παίρνει τιμές από τον παρακάτω πίνακα:

Φορά κίνησης ανάλογα με το τεταρτημόριο για δεξιόστροφο τόξο (G02)			
Τεταρτημόριο Κύκλου	Γωνία α1	X	Y
	$0^o<\alpha\leq 90^o$	+	-
	$90^o<\alpha\leq 180^o$	+	+
	$180^o<\alpha\leq 270^o$	-	+
	$270^o<\alpha\leq 360^o$	-	-

Πίνακας 5.3 – Κατεύθυνση κίνησης αξόνων ανάλογα με το τεταρτημόριο για G02

5.4.5. Τερματισμός - Εντολή 'M30'

Η Εντολή M30 τερματίζει το πρόγραμμα εφαρμογής, απενεργοποιεί την κεφαλή και βάζει τον μικροελεγκτή σε sleep mode.

5.5 Συμπεράσματα:

1) Ανάλογα με τον prescaler που θα επιλέξουμε στον driver οι τιμές που μπορεί να πάρει το micro step είναι:

Prescaler	Microstep (mm)
1	0.2
1/2	0.1
1/4	0,05
1/8	0,025
1/16	0,0125
1/32	0,00625

Πίνακας 5.4 – Prescaler Driver – Micro step

Για καλή ακρίβεια θέσης θέλουμε όσο το δυνατόν μικρό microstep. Όσο μικρότερο το micro step όμως τόσο μεγαλύτερη η τιμή stepx που θα πρέπει να αποθηκευτεί στον καταχωρητή. Για τον λόγο αυτό για την μετατόπιση θα χρησιμοποιήσω δύο καταχωρητές των 8 bit (=16 bit). Με 16 bit, η μέγιστη μετατόπιση που μπορεί να αποθηκευτεί είναι:

$$ds=2^{16}xmicrostep$$

Prescaler	ds_{max} (mm)
1	13107,2
1/2	6553,6
1/4	3276,8
1/8	1638,4
1/16	819,2
1/32	409,6

Πίνακας 5.4 – Prescaler Driver – Μέγιστη μετατόπιση

Στην συγκεκριμένη κατασκευή η μέγιστη μετατόπιση που θα μπορούσαμε να έχουμε είναι ds_{max}=460 mm. Για prescaler = 1/32 θα υπάρχει πρόβλημα στην περίπτωση που η μετατόπιση ξεπεράσει τα 409,60 mm καθώς θα έχουμε υπερχείλιση του μετρητή θέσης.

2) Η σχέση 4.73 μεταφράζει την ταχύτητα σε δυαδικό αριθμό στον καταχωρητή σύγκρισης. Ο παράγοντας N είναι ο prescaler του timer του μικροελεγκτή και η τιμή του επηρεάζει την ακρίβεια στην ταχύτητα. Με prescaler N=1 σημαίνει ότι κάθε χτύπος ρολογιού προσθέτει ή αφαιρεί από την timer ένα bit. Για prescaler N=8 χρειάζονται 8 χτύποι ρολογιού για να αλλάξει τιμή ο timer.

Για παράδειγμα έστω ότι η επιθυμητή ταχύτητα είναι v=10,45 mm/sec και ο ότι ο prescaler των drivers είναι 1/32 (micro step=0,00625). Αν ο prescaler του timer είναι N=1024 ο καταχωρητής σύγκρισης OCR0A σύμφωνα με την 4.73:

$$OCR0A = \frac{16000000}{2 \cdot 1024 \cdot (\frac{10,45}{0,00625})} - 1 = 3,67 \quad (5.1)$$

Στην μετατροπή του δεκαδικού 3,67 σε δυαδικό το δεκαδικό μέρος (0,67) χάνεται και τελικά η τιμή του OCR0A είναι:

$$OCR0A = (3)_{10} = (11)(2\ bit)$$

Και η πραγματική ταχύτητα θα είναι:

$$v_R = \frac{f\ \cdot microstep}{2N(OCR0A+1)} = \frac{16000000 \cdot 0{,}00625}{2 \cdot 1024 \cdot (3+1)} = 12{,}20\ mm/\sec \quad (5.2)$$

Προκύπτει μια σημαντική απόκλιση της πραγματική ταχύτητας με την επιθυμητή:

$$dv = v_R - v = 12{,}20 - 10{,}45 = 1{,}75\ mm/\sec\ (5.3)$$

Οι ίδιοι υπολογισμοί με τον prescaler του timer 0 να είναι N=1:

$$OCR0A = \frac{16000000}{2 \cdot 1 \cdot \left(\frac{10{,}45}{0{,}00625}\right)} - 1 = 4784{,}68 = (4784)_{10} = (1001010110000)(13bit)(5.3)$$

Η πραγματική ταχύτητα:

$$v_R = \frac{f\ \cdot microstep}{2N(OCR0A+1)} = \frac{16000000 \cdot 0{,}00625}{2 \cdot 1 \cdot (4784+1)} = 10{,}449\ mm/\sec \quad (5.4)$$

Και η απόκλιση, σχεδόν μηδέν:

$$dv = v_R - v = 10{,}449 - 10{,}45 = -0{,}001\ mm/\sec \quad (5.5)$$

Παρόλο που το ιδανικό θα ήταν ο prescaler στον timer να έχει την μικρότερη δυνατή τιμή N=1, είναι πρακτικά μη εφικτό (με τον σχεδιασμό του hardware όπως έχει γίνει). Αυτό διότι ο καταχωρητής OCR0A (8 bit) θα πρέπει να πάρει πολύ μεγάλη τιμή.

Αυτό που θα μπορούσε να γίνει για αντιστάθμιση είναι να μειώσουμε λίγο την ανάλυση στην μετατόπιση διαλέγοντας μεγαλύτερο prescaler στον driver (π.χ. 1/8), το οποίο θα βοηθήσει να μειώσουμε λίγο τον prescaler του timer(π.χ. N=256). Με αυτόν τον τρόπο ο λόγος N/micro_step διατηρείται σταθερός (1024/32=256/8) .

5.6. Σημεία Βελτίωσης:

Λογισμικό CAM:

1) Εντοπισμός οντοτήτων, που αλληλοκαλύπτονται και αντικατάσταση τους με μια οντότητα.

2)Εμφάνιση μηνύματος σφάλματος όταν μέρος του σχεδίου είναι εκτός ορίων σχεδίασης.

Αλγόριθμοι Παρεμβολής:

1)Στρογγυλοποίηση της τιμής στον καταχωρητή OCR0A (ταχύτητα στον x) στον πλησιέστερο ακέραιο. Κατά αυτόν τον τρόπο η δεκαδική τιμή X,99 γίνεται X+1 (απόκλιση 0,01) αντί για X (απόκλιση 0,99). Η πραγματική ταχύτητα προσεγγίζεται καλύτερα καθώς το μέγιστο σφάλμα περιορίζεται στο 0,5 αντί για 0,99. Φυσικά υπάρχει μικρή απόκλιση από την επιθυμητή ταχύτητα κατεργασίας.

2)Επαναπροσδιορισμός της πραγματικής ταχύτητας στον άξονα x με την στρογγυλοποιημένη τιμή του OCR0A και υπολογισμός της ταχύτητας στον άξονα y λαμβάνοντας υπόψη την πραγματική ταχύτητα στον x σύμφωνα με την σχέση:

$$v_y = \frac{dy}{dx} v_x$$

Η έκφραση της ταχύτητας στον άξονα y, συναρτήσει της ταχύτητας στον άξονα x φαίνεται να οδηγεί σε καλύτερη ακρίβεια.

3) Στρογγυλοποίηση του καταχωρητή OCR2A (ταχύτητα στον y)

Hardware:

1) Προστασία μέσω τερματικών διακοπτών στο τέλος της διαδρομής του κάθε άξονα.

2) Η ταχύτητα στον άξονα y να ρυθμίζεται από τον timer 1. Ο καταχωρητής σύγκρισης OCR1A είναι 16bit. Ο prescaler του timer θα μπορεί να πάρει τιμή N=1 με αποτέλεσμα να αυξηθεί πολύ η ανάλυση της ταχύτητας στον άξονα y.

Για να γίνει αυτό θα πρέπει να αλλάξει η πλακέτα. Το σήμα για την ρύθμιση του servo moter να γίνεται από τον timer2, και το σήμα για τον έλεγχο του βηματικού κινητήρα στον άξονα y να γίνεται από τον timer 1.

5.7 Επίλογος:

Αναζητώντας στο διαδίκτυο για το που προσανατολίζεται η έρευνα σχετικά με τα CNC, διαπίστωσα πως υπάρχει αυξημένο ενδιαφέρον για τους 3D εκτυπωτές. Χαρακτηριστικό είναι η δημιουργία πολλών Startup εταιριών που στοχεύουν στην ανάπτυξη οικονομικών οικιακών 3D desktop εκτυπωτών.

Το φάσμα εφαρμογών είναι ευρύτατο και αφορά τομείς όπως οδοντιατρική, βίο-ιατρική, μοντελισμό, κατασκευές, και πολλά άλλα. Τα επόμενα χρόνια σε κάθε σπίτι θα υπάρχει ένας 3D εκτυπωτής (όπως υπάρχει προσωπικός υπολογιστής, πλυντήριο ρούχων κ.α.). Αν υπάρχει ανάγκη για π.χ. ένα ανταλλακτικό για το ψυγείο, θήκη για το κινητό, τεχνητό προσθετικό μέλος και πολλά άλλα το μόνο που χρειάζεται είναι η αγορά του 3D μοντέλου από την εταιρία.

Προσπαθώντας να καταλάβω την φιλοσοφία που λειτουργούν οι desktop 3D Printers που κυκλοφορούν σήμερα στην αγορά, έκανα μια έρευνα στο διαδίκτυο. Σε όλες τις περιπτώσεις που εξέτασα η επεξεργασία του 3D μοντέλου για την δημιουργία του κώδικα G μπορεί να γίνει από δωρεάν λογισμικά ανοικτού κώδικα (π.χ. Cura).

Χρησιμοποίησα ένα τέτοιο λογισμικό για να επεξεργαστώ ένα απλό 3D μοντέλο και δημιούργησα το αρχείο .nc που περιέχει τον κώδικα G. Στην συνέχεια άνοιξα το αρχείο με το notepad για να το διαβάσω. Το συμπέρασμα που κατέληξα είναι πως

1. Χρησιμοποιούνται αλγόριθμοι παλμού και

2. Ο αλγόριθμος παρεμβολής εκτελείται από το πρόγραμμα CAM και ο κώδικας G που δημιουργείται περιέχει μια γραμμή για κάθε παλμό. Οπότε δεν υπάρχει εντολή G03 ή G02 αλλά μόνο G00 και G01.Ο κώδικάς περιλαμβάνει υπερβολικά μεγάλο αριθμό εντολών (της τάξης 10^6 γραμμές κώδικα).

Η έρευνα σήμερα επικεντρώνονται κυρίως:

- Στην ανάπτυξη διαφόρων υλικών με συγκεκριμένες ιδιότητες (πυραντοχή, ελαστικότητα, ανθεκτικότητα)

- 3D εκτύπωση μεταλλικών αντικειμένων

- Αλγόριθμοι παρεμβολής για διερμηνεία καμπύλων επιφανειών μεταβλητής ακτίνας

Βιβλιογραφία

[1] ISO 6983-1:2009: Automation systems and integration - Numerical control of machines - Program format and definitions of address words - Part 1: Data format for positioning, line motion and contouring control systems. International Organization for Standardization.

[2] Atmel, "8-bit AVR Microcontrollers",ATmega328/P_Datasheet_Complete,[Rev. Dec. 2016]

[3] Ying Bai. Windows Serial Port Programming Handbook. CRC Press LLC 1st edition,2005

[4] Texas Instrument, DRV8825 Stepper Motor Controller IC datasheet [Rev. July 2014]

[5] Open Pulse, 42BYGHW208 Stepper Motor Datasheet

[6] Tower Pro, Servo Motor SG90 Datasheet

Α1. Αλγόριθμος Στοίχισης γεωμετρικών στοιχείων

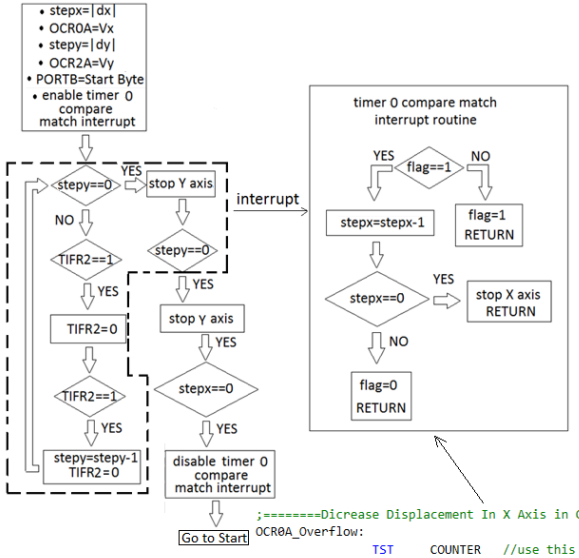
```
//Align Entities Algorithm
//Written in Dev-C++ 5.11
//Author : Petros Iakovou
//Date: 02/02/2021

flag=1;
int revisions=0;                // to count how many loop will make
while(flag==1 && revisions<100) // normally the revisions<100 does not need. but i will use it for safe is for some reason will have continious loop
{
    flag=0;                     // make the flag =0 if no any changes happens flag will remain zero and will not repeat the loop
    for (j=0;j<=i_E;j++)
    {
        for (k=j+1;k<=i_E;k++)  //compare with another (k) entity (line or arc)
        {

//============CASE 1 : if the end point of the current entity k is the same with end point of another k entity===========

            if(E[k].Get_x2()==E[j].Get_x2() && E[k].Get_y2()==E[j].Get_y2())
            {

                s1=E[k].Get_type();      // take the type of (k) entitity

                if(s1=="G01")    // if it is line
                {
                E[k].SetLine((char*)"G01",E[k].Get_x2(),E[k].Get_y2(),E[k].Get_x1(),E[k].Get_y1());     //reverse line's coordinates
                }
                if(s1=="G03")      // if it is arc
                {
                    rx=E[k].Get_x1()+E[k].Get_I1();      // find arc's center x coordinate
                    ry=E[k].Get_y1()+E[k].Get_J1();      // find arc's center y coordinate
                    I=rx-E[k].Get_x2();                 //  find I2
                    J=ry-E[k].Get_y2();                 // find J2
                    E[k].SetArc((char*)"G02",E[k].Get_x2(),E[k].Get_y2(),E[k].Get_x1(),E[k].Get_y1(),I,J);   //reverse arc's coordinates
                }
                flag=1;      // raise flag
            }
```

```
//============CASE 2: if the start point of the current entity j is the same with start point of another k entity===========

        if(E[k].Get_x1()==E[j].Get_x1() && E[k].Get_y1()==E[j].Get_y1())
        {
            s1=E[k].Get_type();

            if(s1=="G01") // if it is line
            {
                E[k].SetLine((char*)"G01",E[k].Get_x2(),E[k].Get_y2(),E[k].Get_x1(),E[k].Get_y1());   //reverse line's coordinates

            //swap entities
                temp=E[j];
                E[j]=E[k];
                E[k]=temp;
            }
            if(s1=="G03")   // if it is arc
            {
                rx=E[k].Get_x1()+E[k].Get_I1();      // find arc's center x coordinate
                ry=E[k].Get_y1()+E[k].Get_J1();      // find arc's center y coordinate
                I=rx-E[k].Get_x2();                 //  find I2
                J=ry-E[k].Get_y2();                 // find J2

                E[k].SetArc((char*)"G02",E[k].Get_x2(),E[k].Get_y2(),E[k].Get_x1(),E[k].Get_y1(),I,J);  //reverse arc's coordinates

            //swap entities
                temp=E[j];
                E[j]=E[k];
                E[k]=temp;
            }
            flag=1;    // if it is line
        }

//============CASE 3: if the start point of the current entity j is the same with the end point of another k entity===========

        if(E[j].Get_x1()==E[k].Get_x2() && E[j].Get_y1()==E[k].Get_y2())
        {
            //swap entities
                temp=E[j];
                E[j]=E[k];
                E[k]=temp;

            flag=1;
        }
```

```
//=============swap entities algorithm====================
//-----if one swap in coordinates happens in previous step we should rearrange the lines order to avoid a continious loop in swap in coordinates (journal 01/01/2021)
        if(flag==1)    //while flag =1 it means that in previous step we make one revesion in entities coordinate
        {
            for (j=0;j<=i_E;j++)
            {
            //if the next (j+1) entity is continious with the current (j) entity move to the next j+1 line
                if(E[j+1].Get_x1()!=E[j].Get_x2() || E[j+1].Get_y1()!=E[j].Get_y2())
                {
                    for(k=j+2;k<=i_E;k++)           //  start the comparison with the (k=j+2) entity
                    {
                        if(E[k].Get_x1()==E[j].Get_x2() && E[k].Get_y1()==E[j].Get_y2())  // if k line is continious with the first swap j+1 with k
                        {
                          //swap entities
                            temp=E[j+1];
                            E[j+1]=E[k];
                            E[k]=temp;
                        }
                    }
                }
            }
        }
//===============end of swap entities algorithm===============
    }
}
revisions++;            // how many revision will need to allign entities?
}
```

A2. Ρουτίνα Εξυπηρέτησης Εντολής G00 Κώδικα G

```
G_00:                                          Dicrease Displacement  in G00
RCALL    Pen_U                                 Date:06/05/2021
Communication to Give                          Written in AtmelStudio
   Data to Registers                           Author:Petros Iakovou
                                               Version: V38
//6th    PORTB                                 C++ executable: CAD2G V42
RCALL    Transmit       //confirm
RCALL    Receive        //receive  motion parameters
OUT      PORTB,TEMP     //start motion

//Enable Timer 0 Compare Match OCR0A Interrupt
LDI      TEMP,0b00000010
STS      TIMSK0,TEMP

SEI //Enable Global Interrupts

//RCALL  Accelarate

RCALL    Count_Steps2 // Call Function Dicrease Displacement In Y Axis in G00

SBI      PORTB,PB4   //Disable Driver in Y Axis

//if Y axis stop motion before x axis wait in Decrease_x axis loop:
Decrease_x:
          L1: CPI     STEPH,0
              BRNE    L1      // if STEPH!=0 goto L1 else move to L2
          L2: CPI     STEPL,0 // if STEPL!=0 goto L2 else continue to the programm
              BRNE    L2

Escape:       CLI     // Disable Global Interrupts

              RJMP    Start

              ;=====================Dicrease Displacement In Y Axis in G00 by function call=======================
              Count_Steps2:
              //Check for displacement in Y axis (STEPY>0)
                              CPI     STEPYH,0b00000000    // compare STEPYH with 0
                              BRNE    half_pulse2     // //if STEPYH ==0 execute function
                              CPI     STEPYL,0b00000000   // compare STEPYL with 0
                              BREQ    exit            // if STEPY==0 return to main programm
              //loop for the first compare match
              half_pulse2:
                              SBIS    TIFR2,OCF2A   // skip if TIFR2 flag == 1 (compare match)
                              RJMP    half_pulse2   // loop if TIFR2 flag == 0
                              SBI     TIFR2,1       // make TIFR2 == 0
              //loop for the second compare match
              one_pulse2:
                              SBIS    TIFR2,OCF2A    // skip if TIFR2 flag == 1 (compare match)
                              RJMP    one_pulse2     // loop if TIFR2 flag == 0
                              SBIW    STEPYL,1       // reduce STEPYL
                              SBI     TIFR2,1        // make TIFR2 == 0
                              BRNE    Count_Steps2   // if STEPY !=0 repeat function else return to main programm
              //Return to main programm
              exit:           RET
              ;=============================================================================================
```

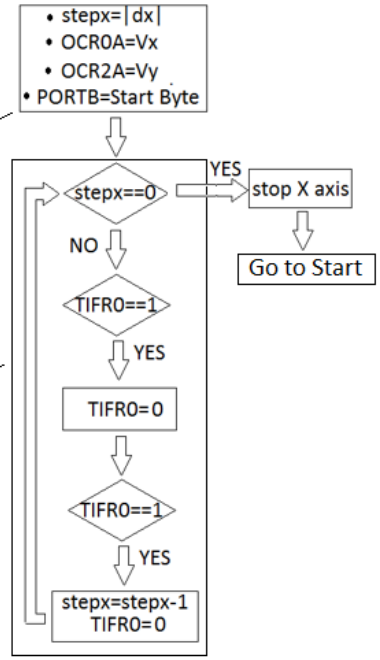



```
;========Dicrease Displacement In X Axis in G00 using Timer 0 Compare Match Interrupt ===================================
OCR0A_Overflow:
                TST     COUNTER   //use this flag to see if there is the first or second compare match after 1 step reduction
                BRNE    Decrease_Dx  //if COUNTER!=0 one pulse have been generated go to Decrease_Dx
                INC     COUNTER     // if COUNTER==0 half palse have been generated

                RETI              // Return to main programm

Decrease_Dx:    SBIW    STEPL,1 // decrease displacement in x axis 1 step
                BREQ    Dis_Int_T0 //if displacement in x == 0 go to Dis_Int_T0
                LDI     COUNTER,0b00000000  // make flag COUNTER ==0

                RETI   //Return to main programm

 Dis_Int_T0:    SBI     PORTB,PB2   //disable driver x
                LDI     TEMP,0b00000000
                STS     TIMSK0,TEMP //disable interrupt Timer 0 Compare Match
//Return from Interrupt
                RETI
```

Α3. Ρουτίνα Εξυπηρέτησης Εντολής G01 Κώδικα G

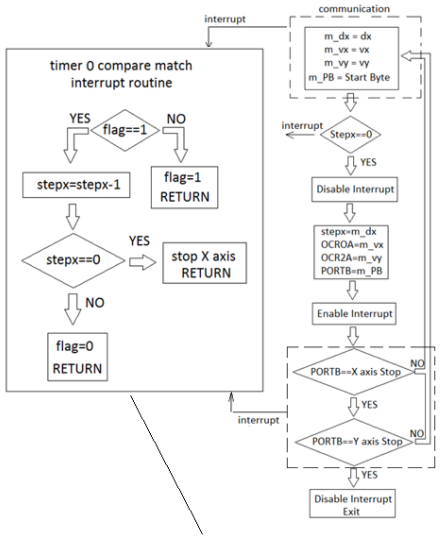
```
 Dicrease Displacement  in G01
 Date:06/05/2021
 Written in AtmelStudio
 Author:Petros Iakovou
 Version: V38
 C++ executable: CAD2G V42
;=============================================================================================================
G_01:
        // i can not reduce precsaller of timer0 because i use the displasement in x axis to control movment
                LDI     TEMP,0b000000101  // prescaller timer0 1024
                OUT     TCCR0B,TEMP
        // but i can reduce precsaller of timer0 in combination with prescaller in driver to achive better
        // accurancy as i do not check for displacement in y axis
                LDI     TEMP,0b00000110  // prescaller timer2 128
                STS     TCCR2B,TEMP

                //PEN DOWN
                RCALL    Pen_D

                RCALL    Count_Steps
                RJMP     Start
;=================Dicrease Displacement In X Axis in G01 by function call=====================================
Count_Steps:
//loop for the first compare match
        half_pulse:
                SBIS    TIFR0,OCF0A  // skip if TIFR0 flag == 1 (compare match)
                RJMP    half_pulse   // loop if TIFR0 flag == 0
                SBI     TIFR0,1      // make TIFR0 == 0
//loop for the second compare match
        one_pulse:
                SBIS    TIFR0,OCF0A   // skip if TIFR0 flag == 1 (compare match)
                RJMP    one_pulse     // loop if TIFR0 flag == 0
                SBIW    STEPL,1       // reduce STEPX one step
                SBI     TIFR0,1       // make TIFR0 == 0
                BRNE    Count_Steps   // if displacement in x !=0 repeat function else return to main programm

                RET
;=============================================================================================================
```

Communication to Give Data to Registers



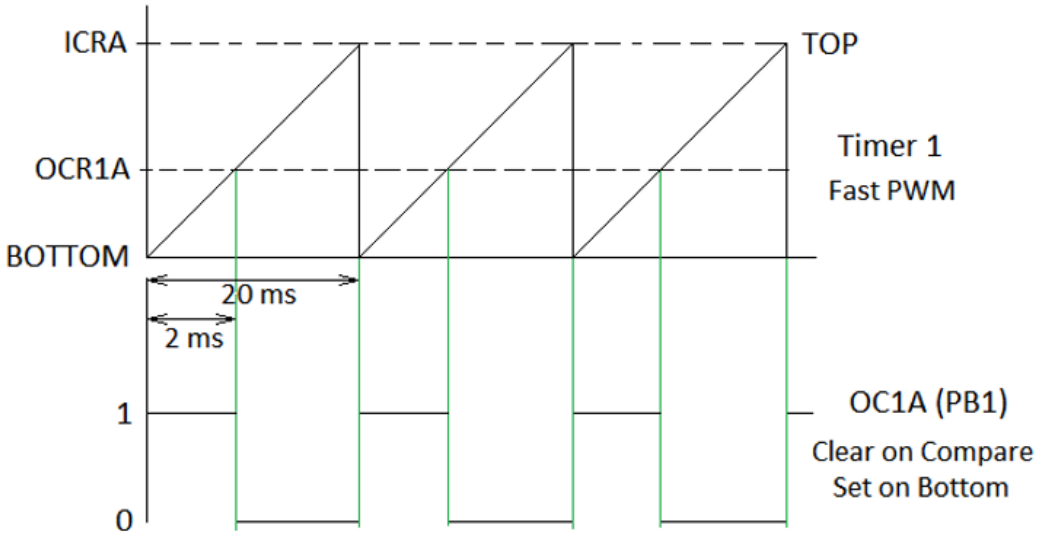
Α4. Ρουτίνα Εξυπηρέτησης Εντολής G03 Κώδικα G

```
Dicrease Displacement  in G03
Date:06/06/2021
Written in AtmelStudio
Author:Petros Iakovou
Version: V38
C++ executable: CAD2G V42
G_03:
                RCALL    Pen_D   //   enable header
                    Communication
                    Store Data to memory
                    not to working registers
                MOV     DIR,TEMP        //start or stop motion*/

            loop:
                CPI     STEPL,0     //if displacement in x axis!=0 go to loop
                BRNE    loop

                // Enable Timer 0 Compare Match OCR0A Interrupt
                LDI     TEMP,0b00000010
                STS     TIMSK0,TEMP

                //Disable Global Interrupts
                CLI

                //Give Data to Working Registers
                MOV     STEPL,DXL       // displasement in x axis
                OUT     OCR0A,V_X       // velocity in x axis
                STS     OCR2A,V_Y       // velocity in y axis
                OUT     PORTB,DIR       // start bit

                //Enable Global Interrupts
                SEI

                //if motors in both axis are disable exit the Count_Steps_G03 loop
                //it can work by check only one motor (x or y) but although it is
                //faster i think that is better to check both axis
                SBIC    PORTB,PB4       //if driver Y is enable go to Count_Steps_G03
                SBIS    PORTB,PB2       //if driver X is disable jump next instruction

                RJMP    G_03

                CLI
                RJMP     Start
```




```
;========Dicrease Displacement In X Axis in G00 using Timer 0 Compare Match Interrupt ====================================
OCR0A_Overflow:
                TST     COUNTER   //use this flag to see if there is the first or second compare match after 1 step reduction
                BRNE    Decrease_Dx  //if COUNTER!=0 one pulse have been generated go to Decrease_Dx
                INC     COUNTER      // if COUNTER==0 half palse have been generated

                RETI             // Return to main programm

Decrease_Dx:    SBIW    STEPL,1 // decrease displacement in x axis 1 step
                BREQ    Dis_Int_T0 //if displacement in x == 0 go to Dis_Int_T0
                LDI     COUNTER,0b00000000  // make flag COUNTER ==0

                RETI   //Return to main programm

 Dis_Int_T0:    SBI     PORTB,PB2   //disable driver x
                LDI     TEMP,0b00000000
                STS     TIMSK0,TEMP //disable interrupt Timer 0 Compare Match
//Return from Interrupt
                RETI
```

Α5. Έλεγχος Σέρβο μοτέρ




```
.equ    Pen_Down            =   1500
.equ    Pen_UP              =   2000
;====================================
//SET UP TIMER 1 . PEN_Header
LDI     TEMP,0b01_0000_00     //toggle OC1A DP->9 PB1 ON COMPARE MATCH
STS     TCCR1A,TEMP
LDI     TEMP,0b000_01_010   // prescaller 8
STS     TCCR1B,TEMP
;=========================================Pen_Up=2000=============
//By giving the value 2000 to OCR1AH a 2 ms pulse produce in PB1 pin
//and the servo moter rotates to +90 degrees position
Pen_U:
                LDI XH,HIGH(Pen_Up)
                STS OCR1AH,XH
                LDI XL,LOW(Pen_Up)
                STS OCR1AL,XL

                RET
;=========================================Pen_Down=1500================
//By giving the value 1500 to OCR1AH a 1.5 ms pulse produce in PB1 pin
// and the servo moter rotates to +00 degrees position
Pen_D:
                LDI XH,HIGH(Pen_Down)
                STS OCR1AH,XH
                LDI XL,LOW(Pen_Down)
                STS OCR1AL,XL

                RET
;=============================================================================
```