ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ
ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ
ΤΟΜΈΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Έλεγχος ανοικτού βρόχου βηματικού κινητήρα μέσω του μικροελεγκτή Arduino.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σπύρος Σ. Φιορέτος

Επιβλέπων : Μαρία - Παρασκευή Ιωαννίδου
Καθηγητής Ε.Μ.Π

Αθήνα, Μάρτιος 2015

ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ
ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ
ΤΟΜΈΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Έλεγχος ανοικτού βρόχου βηματικού κινητήρα μέσω του μικροελεγκτή Arduino

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σπύρος Σ. Φιορέτος

Επιβλέπων : Μαρία – Παρασκευή Ιωαννίδου
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Μαρτίου 2015.

.....	.....	.....
Ιωαννίδου Μαρία - Παρασκευή Καθηγητής	Μέντζας Γρηγόριος Καθηγητής	Τσαραμπάρης Παναγιώτης Λέκτορας

Αθήνα, Μάρτιος 2015

.....
Σπύρος Σ. Φιορέτος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

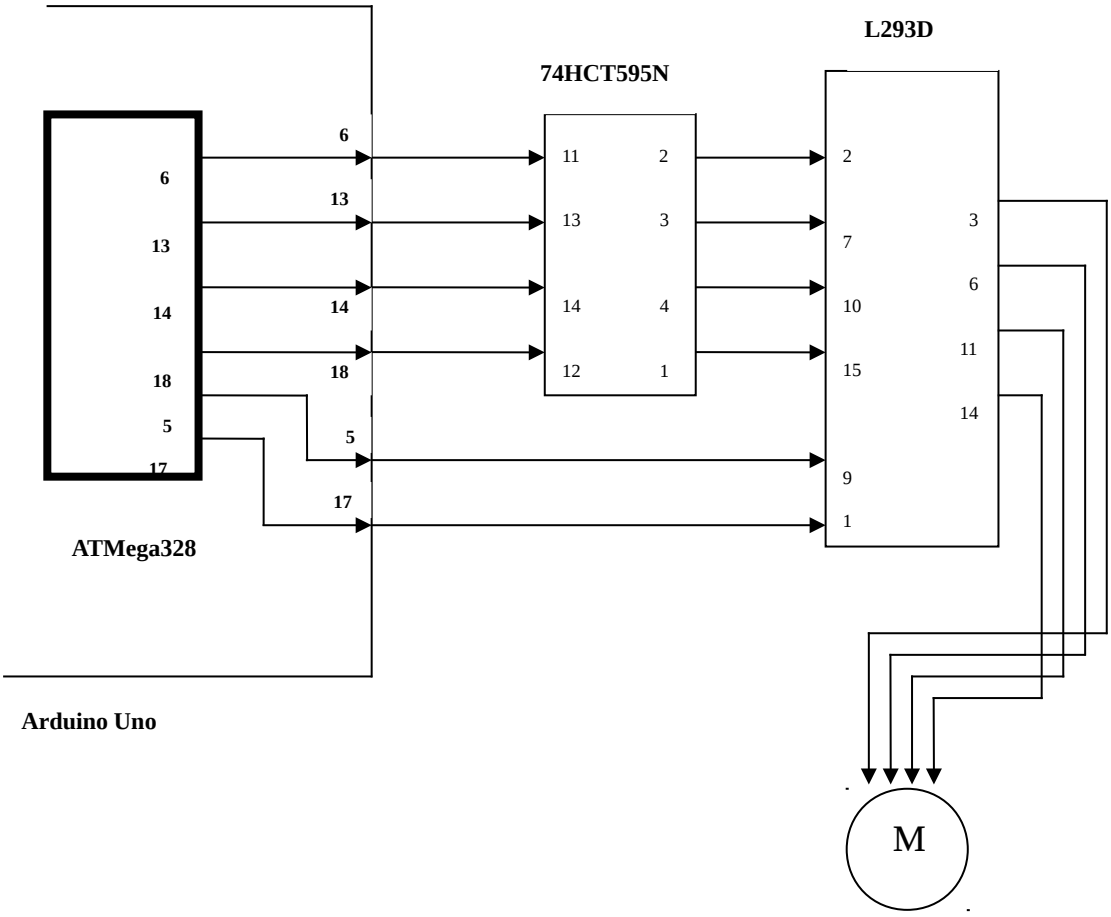
Copyright © Σπύρος Σ. Φιορέτος, 2015

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περιεχόμενα **Σελ.**


L293D
74HCT595N
6
13
14
18
5
17
ATMega328
Arduino Uno
11
13
14
12
2
3
4
1
2
7
10
15
9
1
3
6
11
14
M

Key words

βηματικός κινητήρας, σύστημα ελέγχου ανοικτού βρόχου, διέγερση φάσεων, full step, half step, microstepping, pulse width modulation, software intensive, hardware intensive, constant speed, acceleration, velocity profile, deceleration, διπολική οδήγηση, full H bridge, hybrid step motor, L293D, Arduino Uno, Adafruit, motor shield, sketches, AF_Motor.h, AccelStepper, μικρουπολογιστής, ATMega328, ενσωματωμένα συστήματα, application engineering

0. Αντί προλόγου - Abstract

Οι βηματικοί κινητήρες αν και προϋπήρχαν για αρκετά χρόνια, εντούτοις η χρησιμοποίηση τους για εμπορικούς σκοπούς και εφαρμογές ξεκίνησε κατά τη διάρκεια της δεκαετίας του 1960 όταν και κατέστη δυνατόν να κατασκευασθούν διακοπτικές διατάξεις από τρανζίστορ ισχύος τέτοιες ώστε να επιτρέπουν την όδευση και την διακοπή σχετικά υψηλής έντασης DC ρευμάτων στα τυλίγματα των κινητήρων. Αυτή η κατά τη βούλησή μας ροή και διακοπή του ρεύματος στον κινητήρα, μέσω των διατάξεων αυτών, έδωσε στον βηματικό κινητήρα την ικανότητα να μπορεί να προσαρμόζεται και να ενσωματώνεται πολύ εύκολα σε μεγαλύτερης κλίμακας ψηφιακά κυρίως συστήματα, έτσι ώστε να φτάσει στο σημείο να χαρακτηρίζεται σαν “ψηφιακή μηχανή”.
Η ραγδαία εξέλιξη των ψηφιακών ηλεκτρονικών και συστημάτων κατά τα αμέσως επόμενα χρόνια οδήγησε σε περαιτέρω χρήσεις των βηματικών κινητήρων, τάση που συνεχίζεται και στις μέρες μας με παγκόσμιο ενδιαφέρον τόσο για την εξέλιξή τους, όσο και για νέες εφαρμογές όπως για παράδειγμα την ανάπτυξη συστημάτων που ενσωματώνουν βηματικούς κινητήρες που ελέγχονται από μικροεπεξεργαστές. Οι μικροεπεξεργαστές εμπλέκονται με διάφορους τρόπους στην διαδικασία ελέγχου των βηματικών κινητήρων. Έτσι έχουμε συστήματα ελέγχου ανοικτού και κλειστού βρόχου στα οποία ο μικροεπεξεργαστής έχει τον κύριο ρόλο.
Περισσότερο χρησιμοποιούνται τα συστήματα ανοικτού βρόχου τα οποία μπορούν να ταξινομηθούν σε δυο βασικές κατηγορίες. Σε αυτά που η λειτουργία τους βασίζεται σε προγραμματιζόμενους μικροελεγκτές ή προγραμματιζόμενα ολοκληρωμένα κυκλώματα. Τα συστήματα αυτά ονομάζονται software – based ή software intensive και το χαρακτηριστικό τους είναι ότι μπορούν να εκτελούν όλες τις λειτουργίες ελέγχου είτε αμιγώς μέσα από το λογισμικό εφαρμογής που τρέχει ο μικροεπεξεργαστής είτε, ανάλογα με τη σχεδίαση, χρησιμοποιώντας διάφορα περιφερειακά ολοκληρωμένα κυκλώματα τα οποία αφού δεχτούν τα κατάλληλα σήματα εισόδου, εκτελούν συγκεκριμένες λειτουργίες ελέγχου. Σε απόλυτη αντίθεση βρίσκονται τα hardware based ή hardware intensive συστήματα στα οποία η πλειοψηφία των λειτουργιών ελέγχου γίνεται από ολοκληρωμένα (μη προγραμματιζόμενα) συνδυαστικά και ακολουθιακά κυκλώματα με τον μικροεπεξεργαστή να περιορίζεται απλά στο να παράγει τα βασικά σήματα χρονισμού και έναρξης / παύσης της λειτουργίας του συστήματος. Στο παρών παρουσιάζεται ένα σύστημα ελέγχου ανοικτού βρόχου που μπορεί να οδηγήσει μέχρι δυο βηματικούς κινητήρες. Οι περισσότερες λειτουργίες ελέγχου υλοποιούνται από λογισμικό ενώ κάποιες από υλικό. Έτσι το σύστημα ανήκει στην πρώτη κατηγορία.

Abstract

This essay is about controlling a hybrid two phase step motor via an open loop control system based on Arduino Uno microcontroller. We program the microcontroller so that the system can perform all the basic functions that a step motor is capable of. All the excitation modes are possible. We constract the motor’s speed profile through a numerical method so that we can monitor it’s behaviour in a step by step manner. Knowing the speed profile we can deside if a particular step motor is suitable for a specific application or not. We can examine if the motor can fullfill all the requirements imposed by the system designer. For example if the motor is capable of reaching it’s pull out rate, or to perform a specific number of steps in a certain amount of time. A more intresting requirement is if we want the motor to excecute a specific number of steps with a specific speed within a very strict time frame without missing the target possition. We can confront such requirements in robotics, CNC machines, plotters and other modern machinery.
Arduino Uno in coinjunction with the Adafruit motor shield has been proven a very reliable software – intensive solution for many small to mid – scale projects.
AF_Motor.h and AcceStepper.h libraries provide many functions that can be very usefull when it comes to programming the system.

1. Ο βηματικός κινητήρας

1.1 Εισαγωγή

Ένα σημαντικό ζητούμενο στα συστήματα βηματικών κινητήρων είναι ο κινητήρας να μπορεί να παράγει ικανοποιητική ροπή στρέψης παρά το σχετικά μικρό του μέγεθος. Για να το πετύχει αυτό χρειάζεται τόσο ο στάτης όσο και ο ρότορας του κινητήρα να έχουν μεγάλο αριθμό “δοντιών” έτσι ώστε να μπορεί να μεταφερθεί μεγάλη ποσότητα μαγνητικής ροής **B**. Αυτή η βασική απαίτηση καλύπτεται από δυο τύπους κινητήρων. Τους κινητήρες μεταβλητής μαγνητικής αντίστασης (variable reluctance) και τους υβριδικούς (hybrid).
Η πιο χαρακτηριστική ιδιότητα του βηματικού κινητήρα είναι η δυνατότητά του στο να “μεταφράζει” τις μεταβολές στην διέγερση (δηλαδή στην ροή του ρεύματος) των τυλιγμάτων του σε επακριβώς προκαθορισμένες μεταβολές στην θέση του άξονά του. Εδώ ένα σημαντικό ζητούμενο είναι η ακριβής τοποθέτηση του άξονα και κατ’ επέκταση του φορτίου του κινητήρα, η οποία γενικώς επιτυγχάνεται μέσω της μαγνητικής ευθυγράμμισης των δοντιών του στάτη και του ρότορα. Στην περίπτωση του υβριδικού κινητήρα η κύρια πηγή της μαγνητικής ροής **B** είναι ένας μόνιμος μαγνήτης ενώ DC ρεύματα τα οποία ρέουν σε ένα ή περισσότερα τυλίγματα κατευθύνουν την ροή αυτή κατά μήκος εναλλακτικών διαδρομών.

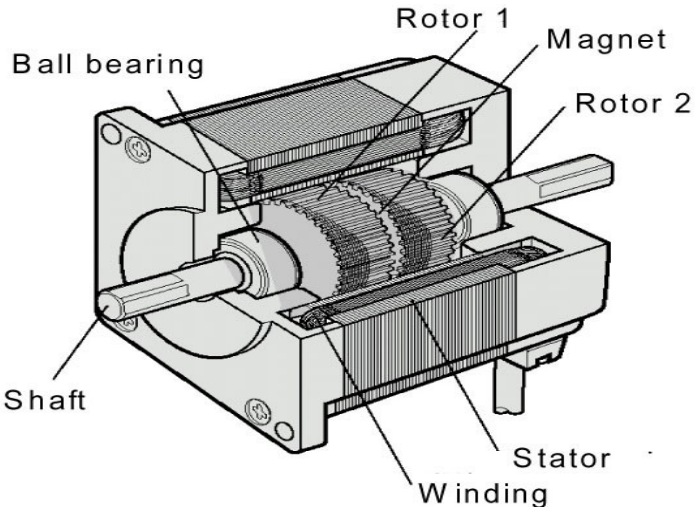
1.2 Ο υβριδικός βηματικός κινητήρας

1.2.1 Γενικά

Ο υβριδικός βηματικός κινητήρας συνδυάζει τα περισσότερα χαρακτηριστικά άλλων τύπων βηματικών κινητήρων όπως οι κινητήρες VR και μόνιμου μαγνήτη (PM).
Με τη χρήση του υβριδικού βηματικού κινητήρα πετυχαίνουμε υψηλή ταχύτητα, μικρή γωνία βήματος και υψηλή ροπή στρέψης από ένα σχετικά μικρό μέγεθος, πράγμα χρήσιμο στις εφαρμογές. Συνήθως μια πλήρης περιστροφή ενός υβριδικού κινητήρα περιλαμβάνει από 200 έως 400 βήματα που σημαίνει ότι έχουμε μια γωνία βήματος από 1.8 έως 0.9 μοίρες αντίστοιχα.

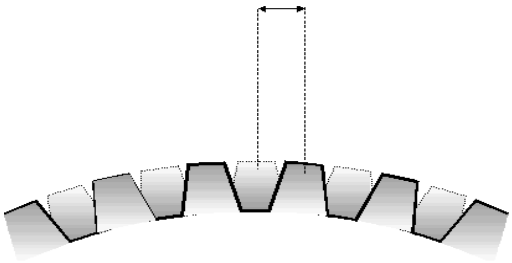
1.2.2 Κατασκευή και λειτουργία

Το σημαντικότερο κατασκευαστικό χαρακτηριστικό ενός υβριδικού κινητήρα είναι ο ρότοράς του στον οποίο βρίσκεται ενσωματωμένος ένας μόνιμος μαγνήτης κυλινδρικού σχήματος. Ο μόνιμος μαγνήτης αποτελεί ένα σημαντικό στοιχείο κάθε σύγχρονου τύπου κινητήρα αφού συμβάλλει στην υψηλή αποτελεσματικότητα και απόδοση του.
Ουσιαστικά αποτελεί το μέσο για την αποδοτικότερη μετατροπή της ηλεκτρικής ισχύος σε μηχανικό έργο.
Το παρακάτω σχήμα (σχήμα 1.2.2.1) απεικονίζει έναν υβριδικό κινητήρα γωνίας βήματος 1.8 μοιρών. Το ένα άκρο του μόνιμου μαγνήτη είναι ο βόρειος πόλος και το άλλο ο νότιος πόλος οι οποίοι και καλύπτονται από ξεχωριστούς οδοντωτούς τροχούς κατασκευασμένους από μαλακό σίδηρο (Rotor 1 και Rotor 2 αντίστοιχα). Στο σύνολό της η κατασκευή αυτή αποτελεί τον ρότορα του κινητήρα. Παρόμοια κατασκευή με οδοντωτές απολήξεις συναντάμε και στον στάτη. Η περιστροφή του κινητήρα οφείλεται στην ευθυγράμμιση των δοντιών του ρότορα και του στάτη.
Σε ένα κινητήρα με βήμα 1.8 μοιρών οι οδοντωτοί τροχοί του ρότορα έχουν 50 δόντια ο καθένας άρα το ένα δόντι απέχει από το άλλο απόσταση ίση προς 7.2 μοίρες. Ένα επίσης σημαντικό κατασκευαστικό στοιχείο είναι ότι τα δόντια στους δυο αυτούς τροχούς δεν είναι ευθυγραμμισμένα μεταξύ τους αλλά διαφέρουν κατά απόσταση ίση με το μισό της απόστασης μεταξύ δυο διαδοχικών δοντιών δηλαδή απόσταση ίση με 3.6 μοίρες (βλέπε σχήμα 1.2.2.2).

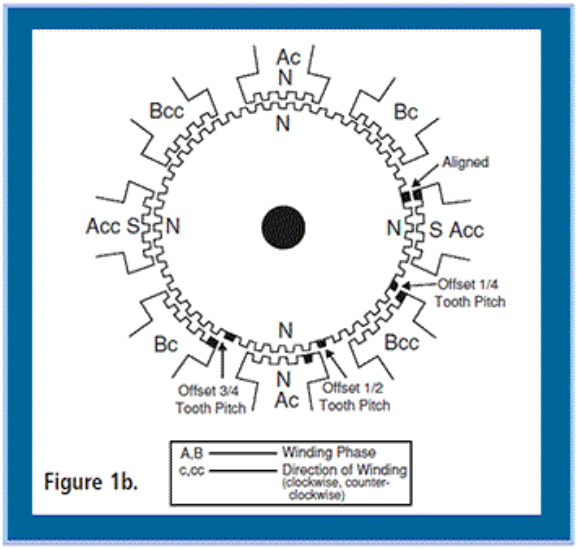


Σχήμα 1.2.2.1 – Τομή υβριδικού βηματικού κινητήρα 2 φάσεων, παράλληλη στον άξονα

Ο στάτης του κινητήρα συνήθως διαθέτει οκτώ πόλους με κάθε πόλο να έχει από δυο έως έξι δόντια. Επίσης ο στάτης διαθέτει και δυο ξεχωριστά τυλίγματα (φάσεις) τα οποία κατευθύνουν την μαγνητική ροή **B** διαμέσου συγκεκριμένων κάθε φορά πόλων ανάλογα με την επιθυμητή κάθε φορά τοποθέτηση του ρότορα. Στην περίπτωση των οκτώ πόλων η κάθε φάση αποτελείται από τις επιμέρους περιελίξεις τεσσάρων πόλων (οι οποίες συνδέονται μεταξύ τους) και συγκεκριμένα εκείνων των πόλων που απέχουν 90 μοίρες μεταξύ τους. Επιπλέον η περιέλιξη των πόλων γίνεται με τέτοιον τρόπο ώστε πόλοι που απέχουν 180 μοίρες να αποκτούν την ίδια πολικότητα, ενώ πόλοι που απέχουν κατά 90 μοίρες να αποκτούν την αντίθετη πολικότητα. Όταν αντιστραφεί η φορά του ρεύματος σε μια φάση τότε αντιστρέφεται και η πολικότητα των πόλων που σημαίνει ότι ο κάθε πόλος του στάτη μπορεί να έχει την έννοια του βόρειου ή του νότιου πόλου. Έστω ότι ονομάζουμε τις φάσεις ως φάση Α και ως φάση Β. Παρατηρώντας το σχήμα 1.2.2.3 βλέπουμε ότι όταν η φάση Α διαρρέετε από ηλεκτρικό ρεύμα τότε οι δύο πόλοι στην κορυφή και στην βάση του στάτη έχουν την έννοια του βόρειου πόλου ενώ οι δύο άλλοι πόλοι που απέχουν από αυτούς 90 μοίρες έχουν την έννοια του νότιου πόλου. Έτσι τώρα οι δυο βόρειοι πόλοι έλκουν τα δόντια του οδοντωτού τροχού του ρότορα που καλύπτει το νότιο πόλο του μόνιμου μαγνήτη (στο σχήμα δεν φαίνεται καθώς είναι πίσω από τον οδοντωτό τροχό που καλύπτει τον βόρειο πόλο του μόνιμου μαγνήτη) ενώ οι δυο νότιοι πόλοι έλκουν τα δόντια του τροχού που καλύπτει τον βόρειο πόλο του μαγνήτη. Αυτό που ουσιαστικά συμβαίνει είναι το ότι λόγω της “συγκέντρωσης” της μαγνητικής ροής **B** σε συγκεκριμένους κάθε φορά πόλους εξαιτίας της ανάλογης διέγερσης των τυλιγμάτων, ο ρότορας του κινητήρα τείνει να ευθυγραμμίσει τα δόντια του με εκείνα του στάτη έτσι ώστε να ελαχιστοποιηθεί η αντίσταση που προβάλλει το κενό αέρα στη κυκλοφορία της ροής **B** στο μαγνητικό κύκλωμα. Η επιθυμητή κατεύθυνση περιστροφής του άξονα του κινητήρα (clockwise ή counter clockwise) καθορίζει και το επόμενο σετ πόλων το τύλιγμα του οποίου θα διεγερθεί από ηλεκτρικό ρεύμα. Το ποία φάση θα διεγερθεί κάθε φορά αλλά και την φορά του ρεύματος στο τύλιγμα την καθορίζει το κύκλωμα οδήγησης του κινητήρα. Επειδή εδώ ο ρότορας έχει 50 δόντια, η απόσταση μεταξύ δυο διαδοχικών δοντιών είναι 7.2 μοίρες. Καθώς ο ρότορας προωθείται κάποια από τα δόντια του θα βρίσκονται ήδη σε ευθυγράμμιση με τα δόντια του στάτη. Τα υπόλοιπα δόντια θα απέχουν από την θέση ευθυγράμμισης κατά διάστημα ίσο προς 3 / 4 ή 1 / 2 ή 1 / 4 της απόστασης των 7.2 μοιρών. Όταν ο άξονας του κινητήρα προωθείται κατά ένα βήμα τότε βρίσκεται σε εκείνη την θέση στην οποία λαμβάνει χώρα η επόμενη ευθυγράμμιση δοντιών του ρότορα και του στάτη. Η θέση αυτή είναι το 1 / 4 της απόστασης των διαδοχικών δοντιών. Έτσι ο ρότορας θα προωθηθεί κατά το 1 / 4 των 7.2 μοιρών. Άρα ο άξονας προωθείται για 1.8 μοίρες. Αυτό είναι και το μήκος του κάθε βήματος. Ο κινητήρας συμπληρώνει μια πλήρη περιστροφή σε 200 βήματα



Σχήμα 1.2.2.2 – Οι οδοντωτοί τροχοί του ρότορα διαφέρουν μεταξύ τους κατά το ήμισυ της απόστασης δυο διαδοχικών δοντιών



Σχήμα 1.2.2.3 – Τομή υβριδικού βηματικού κινητήρα 2 φάσεων, παράλληλη στον άξονα

1.3 Πλήρες βήμα, μισό βήμα, microstepping

Η συνεχόμενη περιστροφή του κινητήρα εξασφαλίζεται μέσω των διαδοχικών διεγέρσεων και από - διεγέρσεων των τυλιγμάτων των φάσεων. Προηγουμένως σημειώσαμε πως οι δυο φάσεις του κινητήρα είναι οι φάσεις Α και Β. Ανάλογα με την φορά του ρεύματος στα τυλίγματα των φάσεων πετυχαίνουμε την περιστροφή του κινητήρα clockwise και counterclockwise. Για περιστροφή σύμφωνα με την φορά των δεικτών του ρολογιού πρέπει η ροή του ρεύματος να είναι πρώτα κατά την θετική φορά και κατόπιν κατά την αρνητική φορά. Έτσι έχουμε την ακολουθία A+,B+,A-,B-. Περιστροφή κατά την αντίθετη φορά επιτυγχάνεται με την εξής ακολουθία στην διέγερση. A+, B-, A-, B+, A+,B-, …. Υπάρχουν τρεις ευρέως διαδεδομένες μέθοδοι διέγερσης των τυλιγμάτων. Αυτές είναι η διέγερση πλήρους βήματος, η διέγερση μισού βήματος και η microstepping. Στην περίπτωση της διέγερσης πλήρους βήματος ο κινητήρας προωθεί τον άξονά του για απόσταση ίση με τη γωνία βήματος. Για παράδειγμα κινητήρας με γωνία βήματος 1.8 μοίρες εκτελεί 200 συνολικά βήματα για μια πλήρη περιστροφή. Η διέγερση πλήρους βήματος μπορεί να επιτευχθεί με δυο τρόπους. Στον ένα τρόπο διεγείρεται μία μόνο φάση του κινητήρα (one phase on, full step) κάθε φορά. Ο τρόπος αυτός έχει το πλεονέκτημα της μικρότερης κατανάλωσης ισχύος. Στον δεύτερο τρόπο ο κινητήρας λειτουργεί με διεγερμένες και τις δυο φάσεις ταυτόχρονα (two phase on, full step). Το πλεονέκτημα αυτής της μεθόδου είναι ότι ο κινητήρας εμφανίζει μεγαλύτερη ροπή στρέψης (περίπου 30% - 40% περισσότερο σε σχέση με το one phase on). Το βασικότερο μειονέκτημα είναι ότι έχουμε διπλάσια κατανάλωση ισχύος.

Σχετικοί με τη μέθοδο είναι οι παρακάτω πίνακες:

# step	ΦΑΣΗ ΥΠΟ ΔΙΕΓΕΡΣΗ			
	A+	B+	A-	B-
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

διέγερση one phase on full step

# step	ΦΑΣΗ ΥΠΟ ΔΙΕΓΕΡΣΗ			
	A+	B+	A-	B-
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

διέγερση two phase on full step

Η διέγερση μισού βήματος (half step) προκύπτει από την εναλλαγή των διεγέρσεων one phase on και two phase on και έχει ως αποτέλεσμα τον υποδιπλασιασμό της γωνίας βήματος σε σχέση με τη γωνία που προκύπτει από τους προηγούμενους τρόπους διέγερσης. Τώρα ο κινητήρας θέλει τον διπλάσιο αριθμό βημάτων για μια πλήρη περιστροφή. Ως αποτέλεσμα έχουμε καλύτερη ποιότητα περιστροφής και ομαλότερη λειτουργία. Μειονέκτημα της μεθόδου αυτής είναι ότι έχουμε περίπου 15% λιγότερη ροπή στρέψης σε σχέση με τη μέθοδο two phase on. Ένας τρόπος για την απαλοιφή του μειονεκτήματος αυτού είναι να αυξήσουμε την ένταση του ρεύματος που διαρρέει τα τυλίγματα του κινητήρα. Για το λόγο αυτό χρησιμοποιούμε ειδικά κυκλώματα οδήγησης.
Σχετικός με αυτή τη μέθοδο διέγερσης είναι ο παρακάτω πίνακας:

# step	ΦΑΣΗ ΥΠΟ ΔΙΕΓΕΡΣΗ			
	A+	B+	A-	B-
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

διέγερση half step

Τέλος το microstepping είναι μια μορφή διέγερσης που αποσκοπεί στο ο κινητήρας να περιστρέφεται όσο το δυνατόν ομαλότερα. Λόγω της φύσης του βηματικού κινητήρα η περιστροφή δεν είναι τελείως ομαλή εξ αιτίας του ότι η κίνηση γίνεται “βήμα - βήμα” και αυτό μπορεί να μην είναι αποδεκτό σε ένα ορισμένο φάσμα εφαρμογών. Με το microstepping μπορούμε να διαιρέσουμε τη γωνία βήματος έως και 256 φορές με αποτέλεσμα να πετυχαίνουμε αρκετά πιο ομαλή περιστροφή και να εξαλείφουμε φαινόμενα συντονισμού που συνήθως λαμβάνουν χώρα στις χαμηλότερες ταχύτητες περιστροφής. Μειονέκτημα τις μεθόδου αυτής είναι η περίπου 30% χαμηλότερη ροπή στρέψης στον άξονα του κινητήρα.

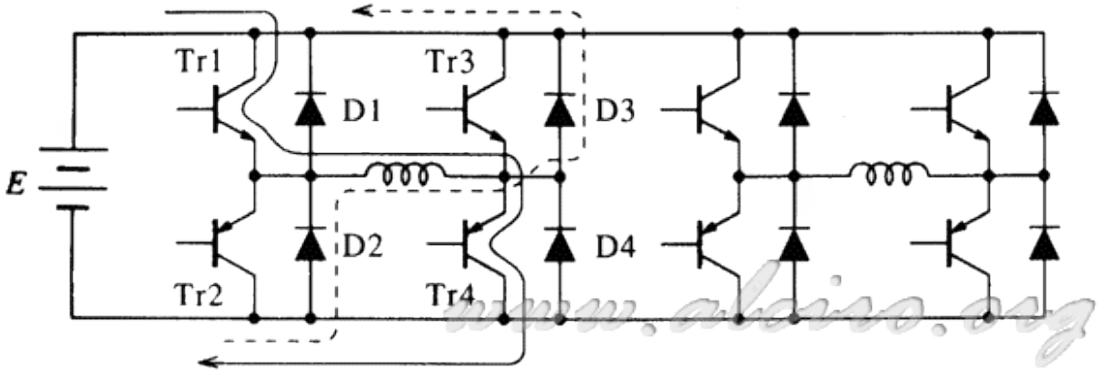
2. Το κύκλωμα οδήγησης

2.1 Εισαγωγή

Στην περίπτωση που έχουμε έναν DC κινητήρα η ταχύτητα περιστροφής του ελέγχεται από την τάση. Μεταβάλλοντας την τάση ελέγχουμε το πόσο γρήγορα ο κινητήρας θα φτάσει στην ταχύτητα που θέλουμε. Στην περίπτωση του βηματικού κινητήρα ωστόσο η μεταβολή της τάσης δεν έχει από μόνη της καμία επίδραση στην μεταβολή της ταχύτητας. Παρόλο που όσο μεγαλύτερη είναι η τάση τροφοδοσίας τόσο πιο γρήγορα το ρεύμα που διαρρέει τις φάσεις του κινητήρα παίρνει την ονομαστική τιμή του, αυτό έχει επίδραση μόνο στην τιμή της (μέγιστης) ταχύτητας περιστροφής. Δεν μεταβάλλει την ταχύτητα περιστροφής. Η ταχύτητα μεταβάλλεται από το ρυθμό που η κάθε φάση του κινητήρα διεγείρεται και από-διεγείρεται. Το κύκλωμα οδήγησης είναι υπεύθυνο για την παροχή του ονομαστικού ρεύματος στα τυλίγματα του κινητήρα στο συντομότερο δυνατό χρόνο. Ακόμα είναι υπεύθυνο και για την εναλλαγή του ρεύματος μεταξύ των φάσεων του κινητήρα. Η είσοδος του κυκλώματος οδήγησης είναι μια παλμοσειρά σταθερής ή μεταβλητής συχνότητας και η έξοδος του είναι τα ρεύματα που διαρρέουν τις φάσεις του κινητήρα με το ρυθμό εναλλαγής στην διέγερση να είναι απευθείας ανάλογος της συχνότητας της παλμοσειράς εισόδου με τον κάθε παλμό να αντιστοιχεί στην εκτέλεση ενός βήματος. Έτσι μεταβάλλοντας την συχνότητα των παλμών εισόδου μεταβάλλουμε ανάλογα και την ταχύτητα περιστροφής του κινητήρα.
Οι βασικότερες κατηγορίες κυκλωμάτων οδήγησης είναι τα κυκλώματα μονοπολικής οδήγησης (unipolar drive) και τα κυκλώματα διπολικής οδήγησης (bipolar drive circuit). Στην μονοπολική οδήγηση τα ρεύματα είναι μονοπολικά, διαρρέουν δηλαδή το τύλιγμα μόνο ως προς μια κατεύθυνση και δεν είναι δυνατόν να έχουμε αλλαγή στην κατεύθυνση αυτή άρα στην πολικότητα. Αντίθετα στην διπολική οδήγηση τα ρεύματα είναι διπολικά μπορούν δηλαδή να διαρρέουν τα τυλίγματα και ως προς τις δυο κατευθύνσεις. Στην περίπτωση του υβριδικού βηματικού κινητήρα η εναλλαγή στην πολικότητα του ρεύματος που διαρρέει τις φάσεις, είναι σημαντική για την σωστή λειτουργία του και έτσι απαιτείται κύκλωμα διπολικής οδήγησης.
Το πιο γνωστό κύκλωμα διπολικής οδήγησης είναι η γέφυρα με τρανζίστορ ισχύος.

2.2 Διπολική οδήγηση

Η απόδοση του κινητήρα είναι μεγαλύτερη εάν όλα τα τυλίγματα του είναι συνεχώς υπό διέγερση και μάλιστα με τέτοιο τρόπο ώστε να παράγεται περισσότερη ροπή στρέψης σε σχέση με την περίπτωση κατά την οποία το κάθε τύλιγμα διεγείρεται για περιορισμένο χρόνο. Για την οδήγηση του διφασικού υβριδικού βηματικού κινητήρα η διπολική οδήγηση είναι ο ιδανικός τρόπος οδήγησης αφού έτσι όλα τα τυλίγματα διαρρέοντα ταυτόχρονα από ρεύμα. Το κύκλωμα που υλοποιεί την διπολική οδήγηση φαίνεται στο παρακάτω σχήμα και είναι τύπου γέφυρας.



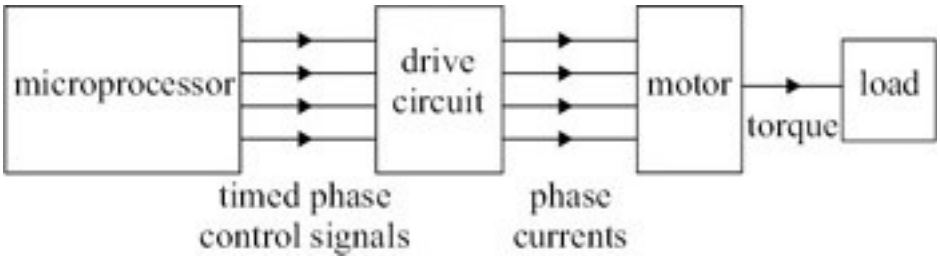
Σχήμα 2.2.1 – Κύκλωμα οδήγησης τύπου γέφυρας

Μάλιστα μερικοί κινητήρες είναι έτσι κατασκευασμένοι ώστε να στοχεύουν σε διπολική οδήγηση και για το λόγο αυτό έχουν τέσσερις ακροδέκτες ώστε να συνδέονται άμεσα με το κύκλωμα οδήγησης. Σε σχέση με την μονοπολική οδήγηση έχουμε αύξηση στη ροπή στρέψης σε ποσοστό περίπου 30%.
Στο κύκλωμα οδήγησης τύπου γέφυρας τα τρανζίστορ άγουν σε ζεύγη σύμφωνα με την επιθυμητή πολικότητα του ρεύματος. Για παράδειγμα για θετική πολικότητα άγουν τα Tr1 και Tr4 και το ρεύμα ξεκινώντας από την πηγή τροφοδοσίας διέρχεται διαμέσου του Tr1, στη συνέχεια διαρρέει το τύλιγμα της φάσης και επιστρέφει την πηγή διαμέσου του Tr4. Σε αντίθετη περίπτωση άγουν τα Tr2 και Tr3 και τώρα το ρεύμα διαρρέει τη φάση έχοντας την αντίθετη πολικότητα. Τα τρανζίστορ λειτουργούν ως διακόπτες και χρειάζονται ένα ξεχωριστό κύκλωμα στη βάση τους για να ενισχύει το σήμα ελέγχου που φτάνει στη βάση τους προερχόμενο από την έξοδο του μικροεπεξεργαστή.
Ένα πρόβλημα παρατηρείται την χρονική στιγμή που Tr1 παίρνει σήμα για να σβήσει και το Tr2 παίρνει σήμα για να αρχίσει να άγει. Εάν η σβέση του ενός τρανζίστορ δεν γίνει γρηγορότερα από την έναυση του άλλου, τα δυο τρανζίστορ θα βραχυκυκλώσουν μεταξύ τους. Για το λόγο αυτό χρησιμοποιούμε την γέφυρα με τις διόδους σε παράλληλη σύνδεση ως προς τα τρανζίστορ. Όταν τα Tr1 και Tr4 σβήσουν και τα Tr2 και Tr3 αρχίσουν να άγουν το ρεύμα στο τύλιγμα δεν έχει προλάβει να αναστρέψει την φορά του και έτσι υπάρχει ο κίνδυνος να περάσει μέσα από σβηστό τρανζίστορ. Όμως λόγω των διόδων περνάει μέσα από τις D2 και D3 και κατόπιν επιστρέφει στην πηγή. Οι δίοδοι παρέχουν δηλαδή ένα μονοπάτι “ελεύθερης διέλευσης” για το ρεύμα.
Με όρους ενέργειας, μαγνητική ενέργεια επιστρέφει πίσω στην πηγή χωρίς να καταναλώνεται σε τυλίγματα και εξωτερικές αντιστάσεις κάτι που αποτελεί πλεονέκτημα των κυκλωμάτων διπολικής οδήγησης.

3.Έλεγχος ανοικτού βρόχου

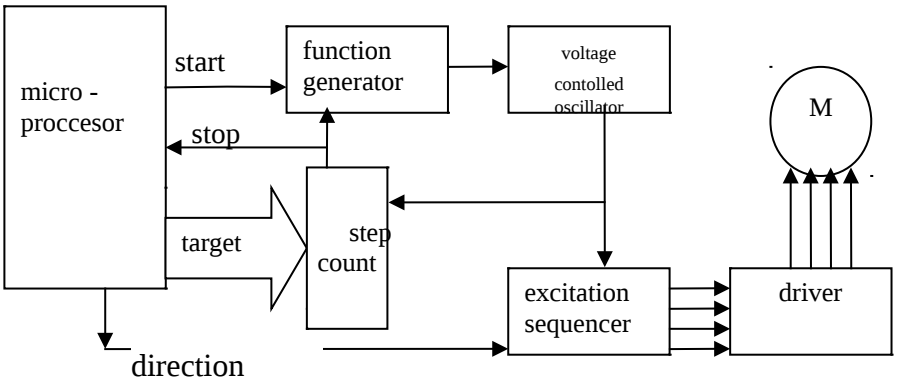
3.1 Εισαγωγή

Ένα από τα πρώτα πράγματα που μας απασχολεί κατά τη σχεδίαση ενός συστήματος βηματικού κινητήρα με έλεγχο ανοικτού βρόχου είναι η επίδοση και η απόδοση του συστήματος σε κατάσταση πραγματικής λειτουργίας. Οι προδιαγραφές που έχουμε θέσει τόσο ως προς το μέγιστα αποδεκτό σφάλμα στην τοποθέτηση του άξονα του κινητήρα όσο και ως προς τη μέγιστη ταχύτητα περιστροφής του κινητήρα καθορίζουν σε μεγάλο βαθμό την επιλογή κινητήρα και κυκλώματος οδήγησης. Κατόπιν αποφασίζουμε για το πώς θα ελέγχεται το κύκλωμα οδήγησης, τον τρόπο δηλαδή με τον οποίο θα τροφοδοτείται με τα απαραίτητα σήματα ελέγχου φάσεων ώστε να παράγει τα ρεύματα που τελικά διαρρέουν τα τυλίγματα (φάσεις) του κινητήρα και τέλος σχεδιάζουμε την ενσωμάτωση του κυκλώματος οδήγησης και του κινητήρα στο υπόλοιπο σύστημα. Ο σκοπός είναι να μεγιστοποιηθεί η απόδοση του συστήματος με το ελάχιστο δυνατό κόστος. Στο παρακάτω σχήμα απεικονίζεται το διάγραμμα βαθμίδων για ένα τυπικό σύστημα ελέγχου ανοικτού βρόχου.



Σχήμα 3.1.1 – Σύστημα ελέγχου ανοικτού βρόχου με μικροεπεξεργαστή

Ο μικροεπεξεργαστής, σύμφωνα με το πρόγραμμα που τρέχει, παράγει τα ψηφιακά σήματα ελέγχου των φάσεων του κινητήρα τα οποία εφαρμόζονται στο κύκλωμα οδήγησης (συνήθως μια γέφυρα) όπου και τελικώς παράγονται τα ρεύματα που ρέουν στα τυλίγματα του κινητήρα. Η διαμόρφωση του παραπάνω συστήματος είναι τέτοια ώστε να στηρίζεται αποκλειστικά (σχεδόν) στο μικροεπεξεργαστή για την υλοποίηση του ελέγχου. Ένα τέτοιο σύστημα ονομάζεται software based ή software intensive.
Βέβαια υπάρχουν και υλοποιήσεις που βασίζονται λιγότερο στον επεξεργαστή και πολύ περισσότερο σε περιφερειακά hardware συστήματα τα οποία συνδέονται και επικοινωνούν με τον επεξεργαστή και υλοποιούν το σύνολο σχεδόν των διεργασιών ελέγχου open loop.
Τα συστήματα αυτά λέγονται hardware based ή hardware intensive. Ένα παράδειγμα τέτοιου συστήματος δίνεται στο παρακάτω σχήμα (Σχήμα 3.1.2).



Σχήμα 3.1.2 – Σύστημα ελέγχου βασοζόμενο σε υλικό

Οι βασικότερες λειτουργίες που πρέπει να υλοποιεί ένα σύστημα ελέγχου βηματικού κινητήρα ανοικτού βρόχου είναι οι ακόλουθες:

- Εκκίνηση του κινητήρα και περιστροφή με σταθερή ταχύτητα.
- Επιλογή της κατεύθυνσης περιστροφής.
- Εκκίνηση του κινητήρα με εφαρμογή κατάλληλης επιτάχυνσης.
- Έλεγχος ότι ο κινητήρας έφτασε την μέγιστη ταχύτητα περιστροφής.
- Έλεγχος ότι ο κινητήρας έφτασε στο σημείο όπου πρέπει να ξεκινήσει η επιβράδυνση ανεξάρτητα από το εάν η μέγιστη ταχύτητα περιστροφής επιτεύχθηκε ή όχι.
- Έλεγχος ότι ο κινητήρας εκτέλεσε τον προβλεπόμενο αριθμό βημάτων οπότε πρέπει να σταματήσει η περιστροφή.

Στα software intensive συστήματα οι παραπάνω λειτουργίες ελέγχου μπορούν να υλοποιηθούν με ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου (π.χ C, C++) ενώ στα hardware intensive συστήματα χρειάζεται προγραμματισμός σε χαμηλότερο επίπεδο, για όσες μονάδες είναι προγραμματιζόμενες, ενώ σε συστήματα που δεν είναι προγραμματιζόμενα ουσιαστικά χρειάζεται υλοποίηση των παραπάνω ελέγχων σε hardware με χρήση ακολουθιακών ολοκληρωμένων κυκλωμάτων LSI και MSI.

3.2 Λειτουργία με σταθερή ταχύτητα

Με τον όρο “ταχύτητα start / stop” εννοούμε την μέγιστη ταχύτητα περιστροφής στη οποία ο κινητήρας μπορεί να εκκινεί αλλά και να σταματάει χωρίς να χάνει βήματα. Με τον όρο “Maximum pull out rate” εννοούμε την μέγιστη ταχύτητα στην οποία μπορεί να περιστρέφεται ο κινητήρας, εν κενώ, χωρίς να χάνει βήματα. Όταν ο κινητήρας είναι υπό φορτίο η μέγιστη ταχύτητα περιστροφής αναφέρεται με τον όρο “Pull out rate”.
Η απλούστερη λειτουργία ελέγχου σε σύστημα ανοικτού βρόχου είναι η οδήγηση του κινητήρα υπό σταθερή ταχύτητα μέχρις ότου ο άξονας του φτάσει στην προκαθορισμένη θέση (στόχος). Η έξοδος του κυκλώματος οδήγησης βρίσκεται σε ευθεία αναλογία με μια παλμοσειρά σταθερής και συγκεκριμένης συχνότητας. Κάθε παλμός της σειράς αυτής αντιστοιχεί σε προώθηση του άξονα του κινητήρα κατά ένα βήμα. Έτσι μεταβάλλοντας τη συχνότητα της παλμοσειράς μεταβάλλουμε αναλόγως και την ταχύτητα περιστροφής του κινητήρα.
Για να γίνει η εκκίνηση χωρίς προβλήματα η συχνότητα των παλμών θα πρέπει να είναι μικρότερη από την συχνότητα (ταχύτητα) start / stop. Εάν η συχνότητα είναι υψηλότερη τότε ο κινητήρας δεν θα μπορέσει να επιταχύνει την αδράνεια του φορτίου και έτσι είτε δεν θα εκκινήσει καθόλου ή θα εκκινήσει χάνοντας βήματα. Αυτό είναι πρόβλημα όταν θέλουμε ο

κινητήρας να λειτουργήσει σε ταχύτητα μεγαλύτερη από την ταχύτητα start / stop. Η λύση που εφαρμόζουμε είναι η εκκίνηση του κινητήρα να γίνει σε ταχύτητα μικρότερη από την ταχύτητα start / stop και κατόπιν ο κινητήρας να επιταχύνει έως ότου αποκτήσει την επιθυμητή ταχύτητα.
Το ίδιο ισχύει και για το σταμάτημα. Αν ο κινητήρας σταματήσει με ταχύτητα μεγαλύτερη από την ταχύτητα start/stop θα χάσει βήματα και άρα και τον στόχο. Για να αποφύγουμε κάτι τέτοιο δεν θα πρέπει να σταματάμε τον κινητήρα απότομα και ενώ αυτός κινείται με ταχύτητα μεγαλύτερη από την ταχύτητα start / stop αλλά θα πρέπει να ελαττώνουμε την ταχύτητά του σταδιακά προς μια ταχύτητα μικρότερη από την ταχύτητα start / stop έτσι ώστε ο κινητήρας να σταματήσει με ασφάλεια χωρίς να εκτελέσει περισσότερα βήματα από αυτά που έχουμε καθορίσει. Εφαρμόζουμε δηλαδή μια σταδιακή επιβράδυνση. Με αυτόν τον τρόπο μπορούμε να πετύχουμε σωστό έλεγχο της κίνησης του κινητήρα και να είμαστε σίγουροι ότι θα εκτελεστεί μόνο ο προκαθορισμένος αριθμός βημάτων.
Βέβαια εάν η επιθυμητή ταχύτητα είναι χαμηλότερη από την ταχύτητα start / stop τότε μπορούμε να εκκινήσουμε και να σταματήσουμε τον κινητήρα με μια απλή εντολή, δηλαδή με ένα σήμα START και ένα STOP χωρίς να εφαρμόσουμε την παραπάνω διαδικασία.
Η υλοποίηση της κίνησης υπό σταθερή ταχύτητα, στην περίπτωση που η επιθυμητή ταχύτητα είναι μικρότερη από την ταχύτητα start / stop μέσω μικροεπεξεργαστή απαιτεί τη χρήση ενός ειδικού χρονιστή – συγκριτή (timer - comparator) ο οποίος παράγει την παλμοσειρά σταθερής συχνότητας. Θα ονομάσουμε τον χρονιστή αυτόν χρονιστή Α.
Έστω SPS (StepsPerSecond) η επιθυμητή ταχύτητα στην οποία θέλουμε να τρέξουμε τον κινητήρα. Τότε ο χρονιστής Α θα πρέπει να προγραμματισθεί έτσι ώστε να παράγει παλμούς που να αντιστοιχούν στην συγκεκριμένη ταχύτητα περιστροφής. Λαμβάνοντας υπ όψιν και την συχνότητα του master clock του μικροεπεξεργαστή έχουμε την εξής σχέση:

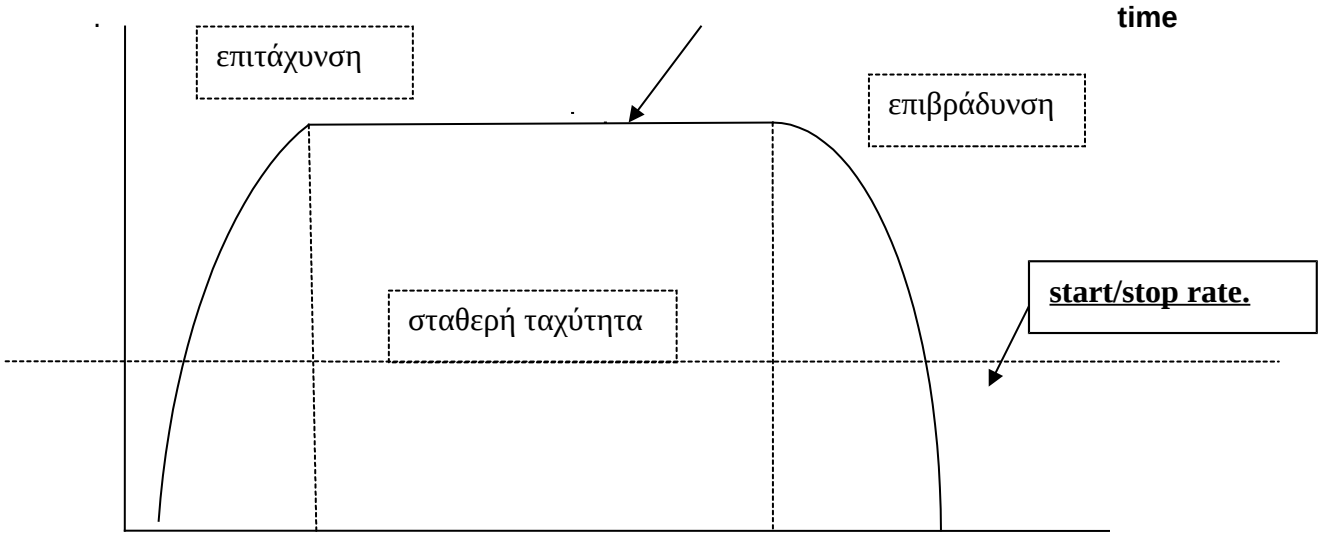
SPS_timer_register = master_clock / SPS.

Όπου SPS_timer_register είναι ένας καταχωρητής 16 bit το περιεχόμενο του οποίου είναι ο αριθμός των παλμών του master clock που μεσολαβεί μέχρι τον επόμενο παλμό στην παλμοσειρά.
Έτσι για παράδειγμα αν το master clock είναι στα 16MHz και θέλουμε SPS = 400 τότε ο SPS_timer_register θα πάρει τιμή ίση με 40000. Αυτό σημαίνει ότι για κάθε 40000 παλμούς του master clock θα παράγεται ένας παλμός από τον χρονιστή Α .
Άρα θα έχουμε SPS = 400Hz ή 400 βήματα το δευτερόλεπτο και αυτός θα είναι τελικά ο αριθμός των βημάτων που θα μετακινηθεί ο άξονας του κινητήρα. Αν το master clock είναι στο 1MHz και θέλουμε SPS = 650 τότε ο SPS_time_register θα πάρει τιμή ίση προς 1504. Δηλαδή για κάθε 1504 παλμούς του master clock θα παράγεται ένας παλμός που αντιστοιχεί σε ένα βήμα.
Στα επόμενα η τιμή του καταχωρητή SPS_time_register θα συμβολίζεται ως ***c***.
Άρα ***c = SPS_time_register***.
Το χρονικό διάστημα **δt** που αντιστοιχεί στην τιμή **c** είναι ίσο προς **$\delta t = c / f_t$** όπου **f_t = 1MHz**.
Άρα έχουμε ότι **δt = 1504 / 1000000 = 0,001504 sec** και αυτό είναι το χρονικό διάστημα που μεσολαβεί μεταξύ δυο διαδοχικών διεγέρσεων των φάσεων ή μεταξύ δυο διαδοχικών βημάτων του κινητήρα. Κάθε φορά που έχουμε την εκτέλεση ενός βήματος γίνεται κλήση μιας ρουτίνας εξυπηρέτησης διακοπών (interrupt service routine - ISR) , το περιεχόμενο του καταχωρητή SPS_time_register μηδενίζεται και ξεκινάει ο επόμενος κύκλος. Το χρονικό διάστημα που διαρκεί η εκτέλεση της ρουτίνας εξυπηρέτησης διακοπής συνήθως κυμαίνεται στα 35μsec και επηρεάζει την μέγιστη ταχύτητα που μπορεί να πιάσει ο κινητήρας. Για παράδειγμα αν έχουμε βηματικό κινητήρα με γωνία βήματος 1,8 μοιρών ή 200 βήματα ανά περιστροφή η μέγιστη ταχύτητα περιορίζεται σε ω_{max} = 2 * π / 200 * 35 μsec = 8500 rpm. Βέβαια για να έχει νόημα ο περιορισμός θα πρέπει η ταχύτητα αυτή να είναι και μικρότερη από την ταχύτητα start / stop του κινητήρα άρα και κατά πολύ μικρότερη από την ταχύτητα pull out. Άρα για την περίπτωση της κίνησης υπό σταθερή ταχύτητα περιορισμός της ταχύτητας λόγω της εκτέλεσης της ρουτίνας εξυπηρέτησης διακοπής δεν υφίσταται στην πραγματικότητα.

3.3 Επιτάχυνση και επιβράδυνση

Είδαμε στην προηγούμενη παράγραφο ότι ο βηματικός κινητήρας δεν μπορεί να εκκινήσει σε οποιαδήποτε ταχύτητα. Στη γενική περίπτωση η εκκίνηση του βηματικού κινητήρα γίνεται σε ταχύτητα το πολύ ίση με την ταχύτητα start / stop και πάντως αρκετά χαμηλότερη από την ταχύτητα pull out. Για το λόγο αυτό μπορούμε να επιταχύνουμε τον κινητήρα έως ότου αυτός αποκτήσει την ταχύτητα αυτή.
Αυτό έχει σαν αποτέλεσμα την δραστική μείωση του χρόνου που απαιτείται για τις τοποθετήσεις του άξονα του κινητήρα στις προκαθορισμένες θέσεις. Όσο όμως πλησιάζουμε σε μια προκαθορισμένη θέση η ταχύτητα του κινητήρα πρέπει να μειώνεται τόσο ώστε, ιδανικά, να φτάσει την ταχύτητα start/stop. Αυτό σημαίνει αφενός ότι η κίνηση θα ολοκληρωθεί το ταχύτερο δυνατόν, αφετέρου (και το σημαντικότερο) ότι ο κινητήρας θα φτάσει ακριβώς στην καθορισμένη θέση χωρίς την απώλεια βημάτων. Μπορούμε να παραστήσουμε γραφικά τη χρονική μεταβολή της ταχύτητας καθώς ο κινητήρας μετακινεί τον άξονά του μεταξύ αρχικής και τελικής θέσης. Ένα τέτοιο γράφημα λέγεται “προφίλ ταχύτητας” και ένα τυπικό παράδειγμα βλέπουμε στο παρακάτω σχήμα. Μπορούμε να διακρίνουμε τρεις περιοχές. Την περιοχή όπου ο κινητήρας επιταχύνει ξεκινώντας ιδανικά με την ταχύτητα start/stop και φτάνοντας την ταχύτητα pull out, την περιοχή όπου η κίνηση γίνεται με σταθερή ταχύτητα (πάντα στην ταχύτητα pull out), και την περιοχή όπου ο κινητήρας επιβραδύνει από την ταχύτητα pull out στην ταχύτητα start/stop.

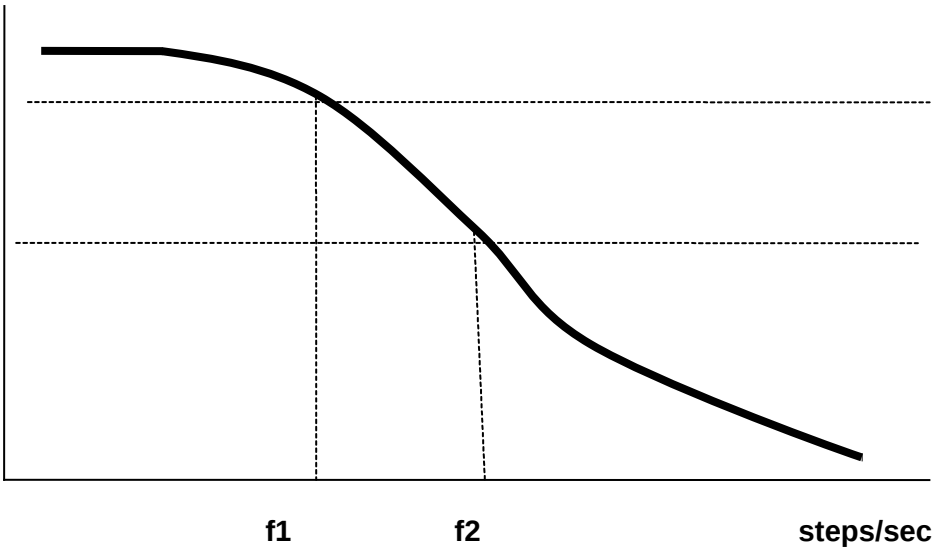
Velocity **pull out rate**



Σχήμα 3.3.1 – Τυπικό προφίλ ταχύτητας βηματικού κινητήρα

Κανονικά το χρονικό διάστημα που απαιτείται για την επιβράδυνση είναι μικρότερο από το διάστημα που απαιτείται για την επιτάχυνση λόγω της ροπής του φορτίου η οποία τείνει να επιβραδύνει το όλο σύστημα.
Στην συντριπτική πλειοψηφία των εφαρμογών ο κινητήρας χρησιμοποιείται για να μεταβάλλει την θέση ενός μηχανικού φορτίου για αρκετά βήματα. Για να σχεδιάσουμε σωστά ένα σύστημα χρειάζεται να γνωρίζουμε πόση ροπή στρέψης είναι σε θέση να παράγει ο κινητήρας κατά την επιτάχυνση, την κίνηση με σταθερή ταχύτητα και την επιβράδυνση έτσι ώστε να μπορεί να περιστρέψει το φορτίο. Επίσης χρειάζεται να γνωρίζουμε και ποια είναι η μέγιστη ταχύτητα στην οποία μπορεί να περιστρέφεται ο κινητήρας όντας φορτισμένος. Τις πληροφορίες αυτές μας τις παρέχει ένα άλλο γράφημα που δείχνει την μέγιστη ροπή στρέψης που μπορεί να αναπτύξει ο κινητήρας σε σχέση με την ταχύτητα περιστροφής. Την ροπή αυτή την ονομάζουμε και ροπή “pull out” για το λόγο του ότι εάν η ροπή του φορτίου ξεπεράσει την ροπή pull out, τότε ο ρότορας χάνει το συγχρονισμό του με το μαγνητικό πεδίο και ο κινητήρας σταματά την λειτουργία του. Στο παρακάτω σχήμα βλέπουμε μια τυπική χαρακτηριστική καμπύλη ροπής pull out – ταχύτητας από όπου παρατηρούμε πως η μέγιστη ροπή του κινητήρα είναι διαθέσιμη σε χαμηλές ταχύτητες ενώ η ροπή ελαττώνεται καθώς η ταχύτητα αυξάνει. Η ροπή μειώνεται και σε κάποιες άλλες ταχύτητες τις λεγόμενες “ταχύτητες συντονισμού ” οι οποίες εξαρτώνται τόσο από τον τρόπο οδήγησης του κινητήρα όσο και από το φορτίο του.

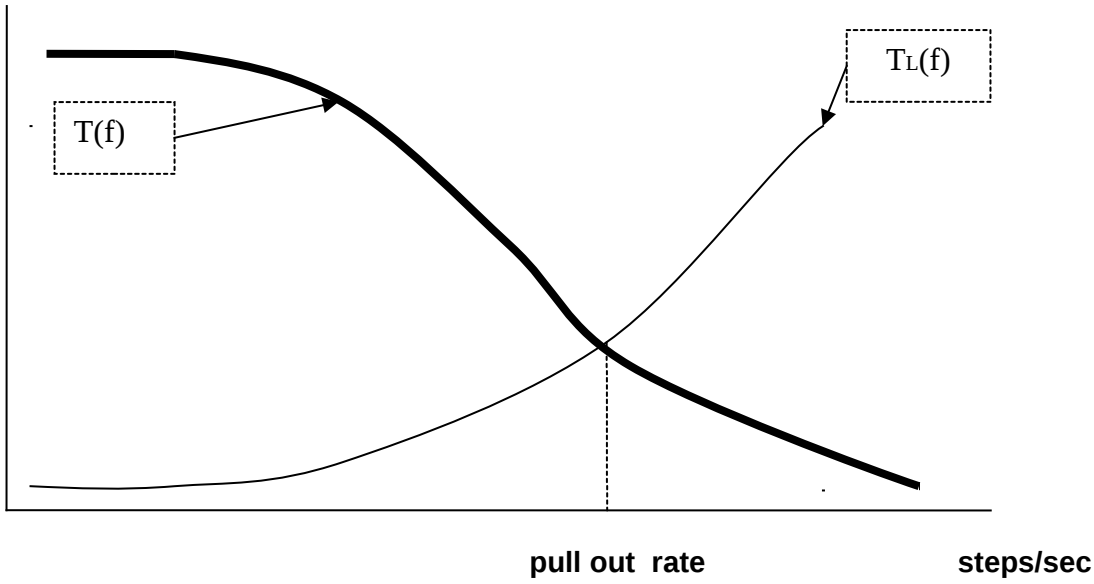
Pull out torque(N.m)



Σχήμα 3.3.2 – Καμπύλη ροπής pull out - ταχύτητας

Στο επόμενο σχήμα βλέπουμε την χαρακτηριστική ροπής pull out σε συνδυασμό με την χαρακτηριστική της ροπής του φορτίου σε σχέση με την ταχύτητα. Για ταχύτητα **f** η ροπή pull out είναι **T(f)** και η ροπή του φορτίου **Tʟ(f)**. Εάν θέλουμε ο κινητήρας να επιταχύνει όσο το δυνατόν γρηγορότερα τότε η ροπή pull out πρέπει να είναι διαθέσιμη σε όλο το φάσμα ταχυτήτων.

Torque



Σχήμα 3.3.3 – Καμπύλη ροπής pull out – φορτίου

Η ροπή αυτή καλύπτει τη ροπή του φορτίου και ακόμη επιταχύνει την αδράνεια του συστήματος σύμφωνα με την εξίσωση :

$$T(f) = T_L(f) + J(d^2\theta/dt^2)$$

Για κινητήρα με *n* φάσεις και *p* δόντια στο ρότορα το μήκος του βήματος είναι $2\pi/np$ και έτσι η ταχύτητα *f* (steps/sec) συνδέεται με την ταχύτητα $d\theta/dt$ ως εξής:

$$d\theta/dt = 2\pi f/np$$

Έτσι έχουμε:

$$T(f) = T_L(f) + (2\pi J /np) \times (df/dt)$$ ή

$$df/dt = [T(f) - T_L(f)] \times np/(2\pi J)$$

Με ολοκλήρωση της παραπάνω εξίσωσης παίρνουμε το χρόνο t που χρειάζεται ο κινητήρας για να φτάσει στην ταχύτητα f καθώς επιταχύνει από στάση.

$$np / (2\pi J) \int_0^t dt = npt / (2\pi J) = \int_0^f df / [T(f) - T_L(f)]$$

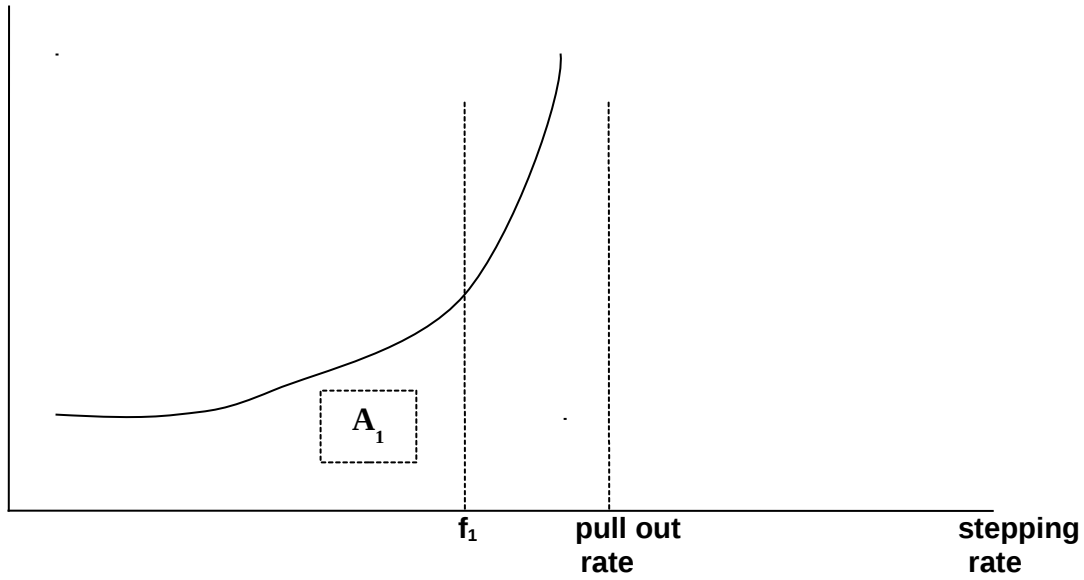
Το ολοκλήρωμα αυτό υπολογίζεται γραφικά και η λύση του ισοδυναμεί με το εμβαδόν της επιφανείας A = A(f) οπότε υπολογίζουμε το χρόνο t για να φτάσουμε σε ταχύτητα f. Στο σχήμα A_1 είναι η επιφάνεια που περικλείεται από την γραφική παράσταση της συνάρτησης $1 /[T(f)-T_L(f)]$ και του άξονα των ταχυτήτων για ταχύτητες από 0 έως την ταχύτητα f_1.

Επομένως ο χρόνος t_1 τον οποίο χρειάζεται ο κινητήρας για την ταχύτητα f_1 είναι:

$$t_1 = 2\pi J A_1 / np$$

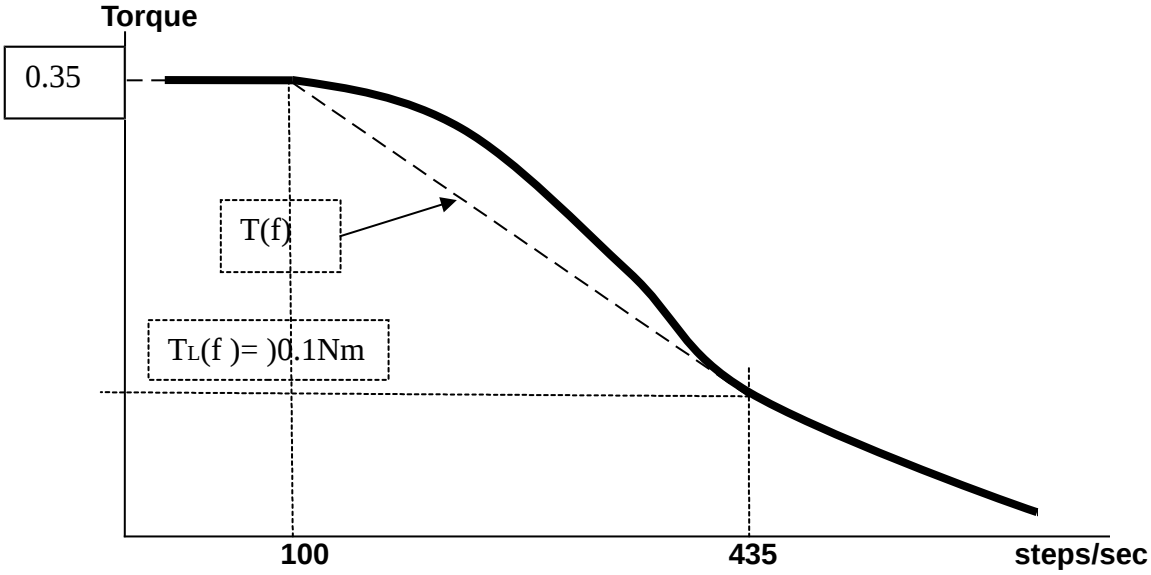
Συνεπώς μπορούμε να κατασκευάσουμε το προφίλ ταχύτητας για την επιτάχυνση εάν επαναλάβουμε την διαδικασία αυτή για διάφορες ταχύτητες ξεκινώντας από την ταχύτητα start/stop και φτάνοντας έως την ταχύτητα pull out.

1/T(f)-TL(f)



Σχήμα 3.3.4 – Γραφική παράσταση συνάρτησης 1/[T(f)-TL(f)]

Η διαδικασία μπορεί να απλοποιηθεί εάν προσεγγίσουμε τις συναρτήσεις T(f) και TL(f) με γνωστές συναρτήσεις. Έτσι λοιπόν μπορούμε την χαρακτηριστική καμπύλη να την προσεγγίσουμε με δυο ευθείες όπως φαίνεται στο σχήμα.



Σχήμα 3.3.5 – Γραφική προσέγγιση καμπύλης ροπής pull out

Έστω λοιπόν ότι η παραπάνω χαρακτηριστική ανήκει σε ένα βηματικό κινητήρα με βήμα 15 μοιρών ο οποίος οδηγεί φορτίο αδρανείας $2x10^4$ Kgm^2 και ροπής 0.1Nm. Η ταχύτητα pull out είναι 435 steps/sec αφού στην ταχύτητα αυτή η ροπή του κινητήρα είναι ίση με τη ροπή του φορτίου. Με βάση τις προσεγγίσεις για την ροπή pull out T(f) ισχύουν τα εξής:

$T(f) = 0.35Nm$ για $0 < f < 100$ steps/sec

και

$T(f) = 0.426 – 0.00075x\ f$ για $100 < f < 435$ steps/sec ενώ

$T_L(f) = 0.1Nm$

Για ταχύτητες έως τα 100 steps/sec έχουμε ότι:

$$A = \int_{}^{f} df / [T(f) – T_L(f)] = \int_{}^{f} df /0.25 = 4f\ [Nms]^{-1}$$

Το μήκος του βήματος είναι 15 μοίρες = 2π / np και άρα έχουμε ότι

$$t = 2\pi JA/np = 0.21\ f\ [ms].$$

Άρα η ταχύτητα αυξάνει γραμμικά με το χρόνο μέχρι την ταχύτητα των 100 steps/sec και την ταχύτητα αυτή ο κινητήρας την “πιάνει” μετά από 21ms.

Για ταχύτητες από 100 έως 435 steps / sec έχουμε:

$$A = \int_{0}^{f} df\ / [T(f) – T_L(f)] = \int_{0}^{100} df /0.25 + \int_{100}^{f} df/(0.426 – 0.00075\ f\ ) =$$

$$= 400 + [(-1333,3\ ln(0,426 – 0,00075\ f\ )]_{100}^{f}$$

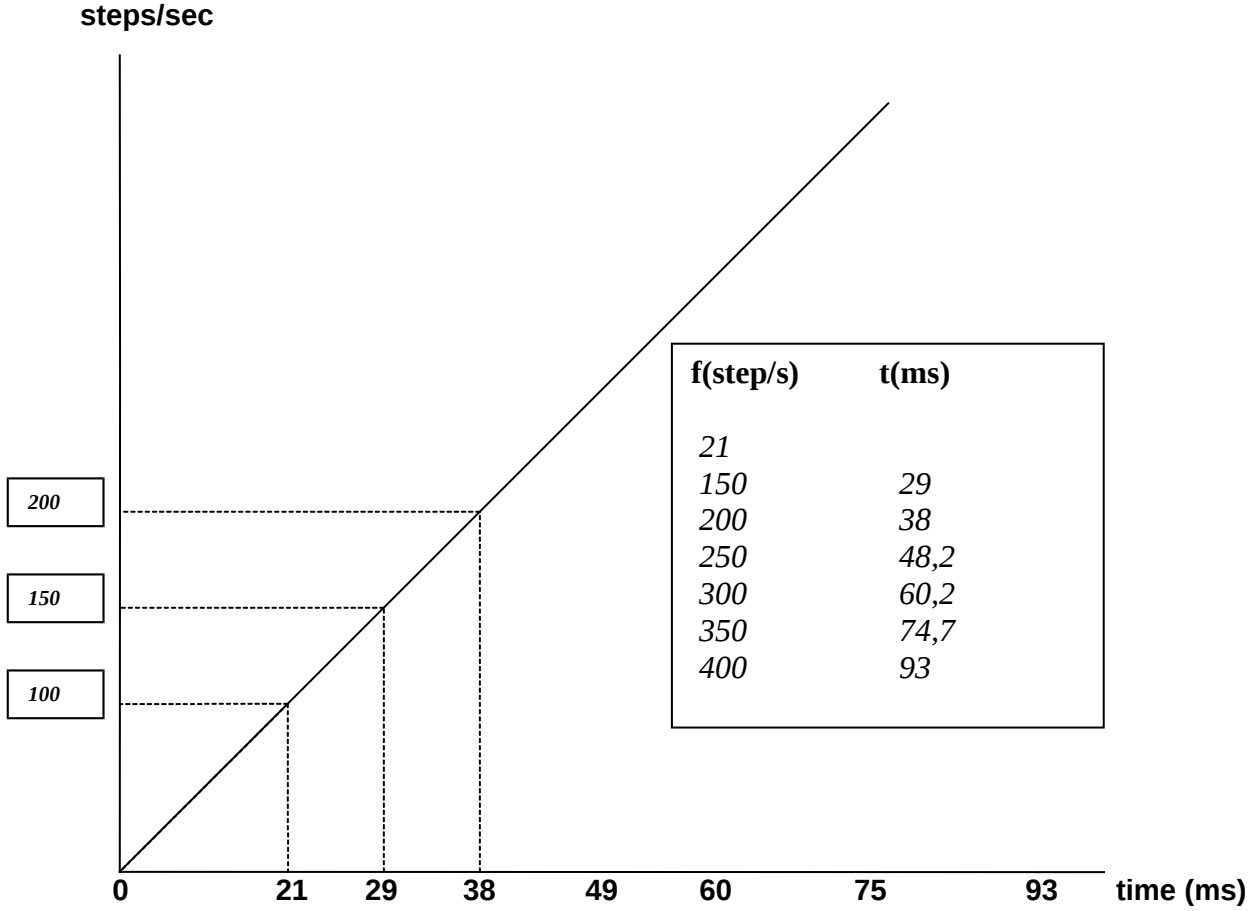
Έτσι από την εξίσωση t = 2πJA/ np προκύπτει ότι ο χρόνος που χρειάζεται ο βηματικός κινητήρας για να φτάσει μια ταχύτητα στο διάστημα 100 – 435 βημάτων ανά λεπτό δύνεται από την σχέση :

t = 2πJA / nρ

όπου A το εμβαδόν που κάθε φορά προκύπτει για διάφορές τιμές του f.

Έτσι μπορούμε τώρα να κατασκευάσουμε το προφίλ ταχύτητας αντικαθιστώντας στις παραπάνω εκφράσεις διάφορες τιμές για την ταχύτητα f.
Για παράδειγμα για την τιμή f = 200 υπολογίζουμε την τιμή του ολοκληρώματος με όρια από 100 έως 200, στην τιμή που θα βρούμε προσθέτουμε 400 και αυτή είναι η τελική τιμή του A.
Μόλις ο κινητήρας φτάσει την ταχύτητα pull out δεν μπορεί να συνεχίσει να επιταχύνει περαιτέρω (τουλάχιστον υπό το συγκεκριμένο φορτίο) και έτσι συνεχίζει την κίνηση του έχοντας σταθερή ταχύτητα, την ταχύτητα pull out.
Για να σταματήσουμε τον κινητήρα θα πρέπει να μειώνουμε την ταχύτητά του από την pull out σε μια χαμηλότερη ταχύτητα στην οποία ο κινητήρας να μπορεί να σταματήσει ακριβώς εκεί που θέλουμε χωρίς επιπλέον βήματα λόγω ροπής αδράνειας.
Από τα παραπάνω συμπεραίνουμε ότι ο βηματικός κινητήρας είναι ο κινητήρας που, περισσότερο από οποιονδήποτε άλλο τύπο κινητήρα, χρειάζεται ο έλεγχος της κίνησής του να γίνεται με βάση το προφίλ ταχύτητας. Γνωρίζοντας το προφίλ ταχύτητας η ενσωμάτωση του κινητήρα σε ένα σύστημα γίνεται με τον βέλτιστο τρόπο. Μπορούμε να υπολογίσουμε πόσο χρόνο χρειάζεται ο κινητήρας για να φτάσει την ταχύτητα που θέλουμε, σε πόσα βήματα θα το κάνει αυτό και σε ποια ταχύτητα είναι ασφαλές να σταματήσουμε τον κινητήρα χωρίς επιπλέον

βήματα. Όλα τα παραπάνω παίζουν σημαντικό ρόλο σε εφαρμογές που χρησιμοποιούμε βηματικούς κινητήρες, και οι οποίες απαιτούν αυστηρό έλεγχο θέσης, ταχύτητας και φορτίου. Στη συνέχεια με βάση τα δεδομένα της παραπάνω γραφικής μεθόδου μπορούμε να κατασκευάσουμε το προφίλ ταχύτητας κατά τη φάση της επιτάχυνσης. Στην επόμενη παράγραφο θα δούμε μια αριθμητική μέθοδο για την παραγωγή του προφίλ ταχύτητας που είναι και η μέθοδος που χρησιμοποιείται κατά τον έλεγχο από μικρουπολογιστικά συστήματα.

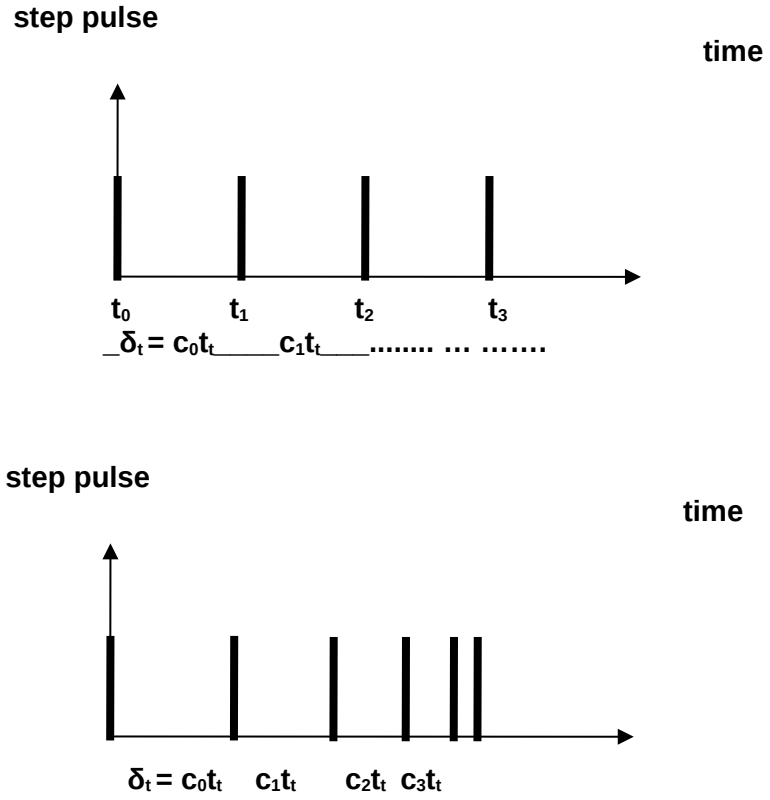


Σχήμα 3.3.6 – Προφίλ ταχύτητας ως γραφική παράσταση της t = 2πJA / nρ

Τα παραπάνω ισχύουν για κινητήρα του οποίου είναι γνωστή η χαρακτηριστική καμπύλη ταχύτητας – ροπής pull out. Εάν η καμπύλη δεν είναι διαθέσιμη μπορούμε να την κατασκευάσουμε ως εξής: Περιστρέφουμε τον κινητήρα σε μια γνωστή ταχύτητα και κατόπιν τον φορτίζουμε σταδιακά έως ότου ο κινητήρας σταματήσει λόγω απώλειας συγχρονισμού. Τη στιγμή εκείνη καταγράφουμε την τιμή της ροπής φορτίου. Η διαδικασία αυτή επαναλαμβάνεται για τρεις φορές (στην ίδια ταχύτητα) και από τις τρεις μετρήσεις λαμβάνουμε το μέσο όρο. Η διαδικασία επαναλαμβάνεται για διάφορες ταχύτητες περιστροφής και έτσι τελικά προκύπτει η καμπύλη.

3.4 Γραμμικό προφίλ ταχύτητας

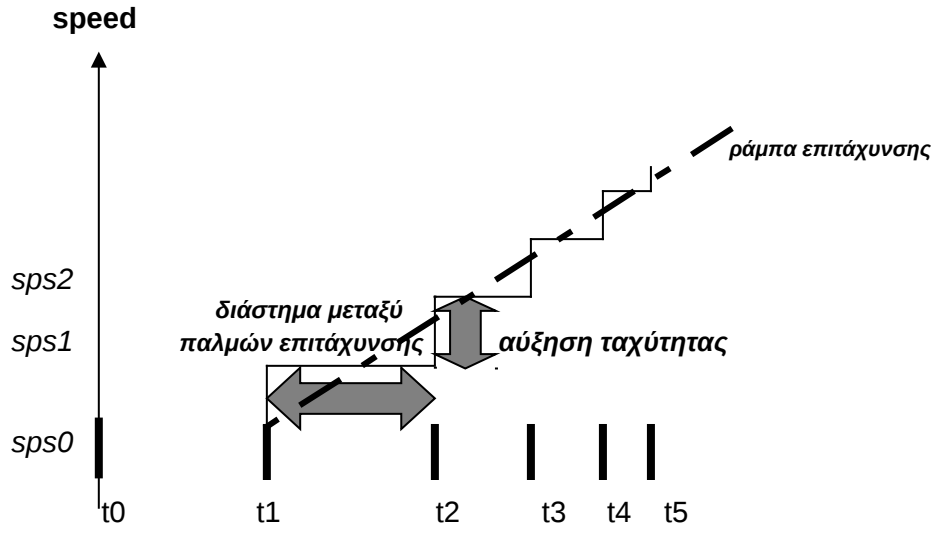
Για να έχουμε την περιστροφή του κινητήρα θα πρέπει να εφαρμοστεί μια παλμοσειρά στην είσοδο της μονάδας που παράγει τα σήματα ελέγχου των φάσεων με κάθε παλμό να αντιστοιχεί σε κίνηση του άξονα για ένα βήμα. Εάν η παλμοσειρά έχει σταθερή συχνότητα ο κινητήρας περιστρέφεται με σταθερή ταχύτητα. Εάν η συχνότητα μεταβάλλεται τότε έχουμε την επιτάχυνση / επιβράδυνση. Στο παρακάτω σχήμα 3.4.1 βλέπουμε μια παλμοσειρά σταθερής συχνότητας και μια μεταβλητής.



Σχήμα 3.4.1 – Παλμοσειρά σταθερής και μεταβλητής συχνότητας

Οι παλμοί αυτοί παράγονται από ένα “ρολόι”σταθερής συχνότητας f_t [Hz],όπου $f_t = 1/t_t$ και το χρονικό διάστημα που μεσολαβεί μεταξύ διαδοχικών παλμών είναι το δ_t .
Το ρολόι αυτό μπορούμε να το πάρουμε υποδιαιρώντας την συχνότητα του master clock του μικροεπεξεργαστή, αλλά μπορεί και να υλοποιηθεί ως μια υπομονάδα ενός hardware συστήματος που συνεργάζεται με τον μικροεπεξεργαστή (περιφερειακό).
Ένα παράδειγμα τέτοιου περιφερειακού συστήματος είναι μια μονάδα FPGA. Υπάρχουν και άλλες πολλές hardwarwe υλοποιήσεις, με το κυριότερο πλεονέκτημά τους να είναι το ότι δεν απασχολούν σχεδόν καθόλου τον μικροεπεξεργαστή σπαταλώντας έτσι υπολογιστικούς πόρους
Το περιφερειακό σύστημα είναι αυτό που ουσιαστικά υλοποιεί τον έλεγχο του κινητήρα και συγχρονίζεται με το master clock του μικροεπεξεργαστή. Μια υπομονάδα του περιφερειακού συστήματος (CCP – capture / compare / pwm) παρέχει τον κατάλληλο χρονισμό στην περιφερειακή μονάδα υποδιαιρώντας αρκετές φορές τη συχνότητα του master clock.

Αυτό, για την περίπτωση που θέλουμε παλμοσειρά σταθερής συχνότητας, γίνεται όπως είδαμε με έναν χρονιστή (χρονιστή Α) και με χρήση μιας μεταβλητής SPS_timer_register η οποία αντιπροσωπεύει το περιεχόμενο ενός καταχωρητή και παίρνει τιμή ίση με την τιμή master_clock / SPS, όπου SPS είναι ο αριθμός των βημάτων ανά δευτερόλεπτο, η ταχύτητα δηλαδή στην οποία επιθυμούμε να περιστρέφεται ο κινητήρας.
Είναι λογικό να θέλουμε ο κινητήρας να κινείται σε ταχύτητα ίση ή πολύ κοντά στην ταχύτητα pull out. Άρα SPS = pull out rate.
Η τιμή ***c = SPS_timer_register*** αντιστοιχεί σε μια εσωτερική καθυστέρηση (interstep delay) που επιβάλλεται μεταξύ των παλμών που παράγονται από το ρολόι του περιφερειακού. Η χρονική καθυστέρηση μεταξύ των παλμών τότε θα είναι $\delta t = ct_t = c / f_t$ η οποία και θα διατηρείται σταθερή.
Για παλμοσειρά μεταβλητής συχνότητας, και επειδή η συχνότητα f_t παραμένει σταθερή, χρειάζεται να μεταβληθεί η εσωτερική καθυστέρηση μεταξύ των διαδοχικών παλμών δηλαδή η τιμή του ***c***. Η εσωτερική καθυστέρηση μεταξύ του πρώτου και του δεύτερου παλμού τίθεται ίση με $c_0 = c$ και κατόπιν προσδιορίζουμε την τιμή της καθυστέρησης μεταξύ των επόμενων παλμών. Για το λόγο αυτόν χρησιμοποιούμε και έναν δεύτερο χρονιστή, τον χρονιστή επιτάχυνσης σκοπός του οποίου είναι να μεταβάλλει τον χρονιστή Α σε περιοδική βάση.
Στο παρακάτω σχήμα 3.4.2 τη χρονική στιγμή t0, t1, t2, t3, t4, t5,..... δίνονται οι παλμοί του χρονιστή επιτάχυνσης.



Σχήμα 3.4.2 – Παλμοί χρονιστή επιτάχυνσης

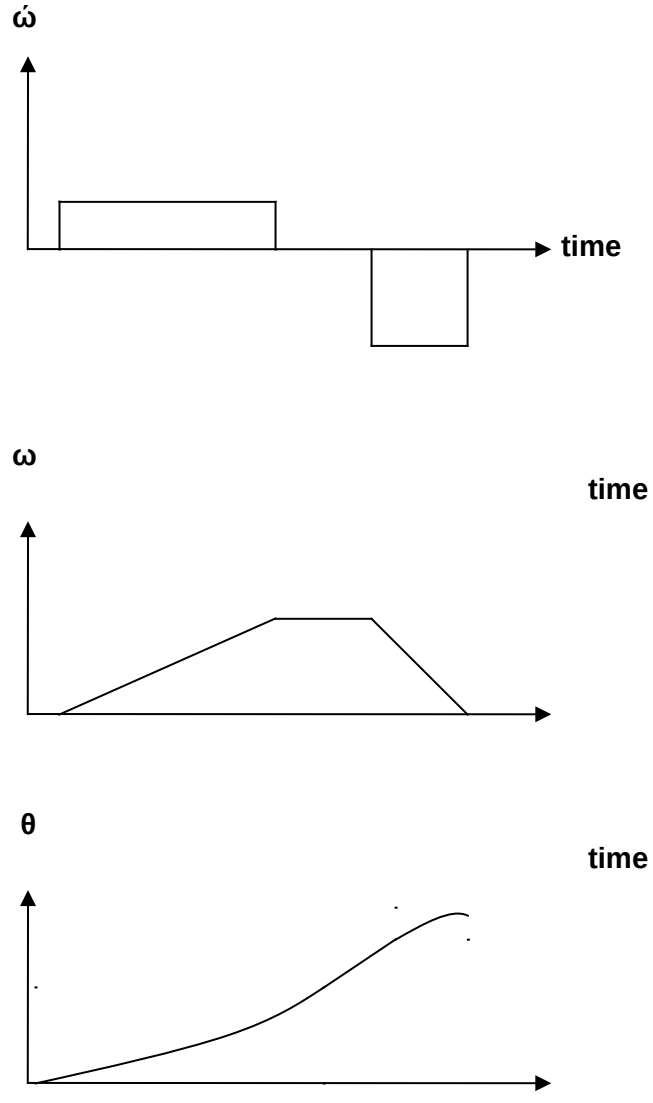
Έστω ότι η ταχύτητα είναι ίση με sps0 και τη χρονική στιγμή t1 δίνεται ο δεύτερος παλμός του χρονιστή επιτάχυνσης. Τότε θα έχουμε την αύξηση της ταχύτητας η οποία για τη χρονική περίοδο από t1 έως t2 θα είναι ίση με sps1. Τη χρονική στιγμή t2 δίνεται ο επόμενος παλμός του χρονιστή επιτάχυνσης και τότε η ταχύτητα αυξάνει εκ νέου και για το χρονικό διάστημα από t2 έως t3 θα είναι ίση με sps2 κοκ.
Αυτό σημαίνει ότι ο χρονιστής Α από t1 έως t2 θα παράγει παλμούς με χρονική καθυστέρηση που δίνεται από την: ***SPS_timer_register = master_clock / sps1 = c1***
Δηλαδή θα παράγει έναν παλμό μετά από ***c1*** παλμούς του master clock.
Η παλμοσειρά αυτή θα αντιστοιχεί σε περιστροφή του κινητήρα με ταχύτητα sps1
Ομοίως από t2 έως t3 θα παράγει έναν παλμό μετά από ***c2*** παλμούς του master clock με την παλμοσειρά αυτή να αντιστοιχεί σε περιστροφή του κινητήρα με ταχύτητα sps2 κλπ.

Αυτό που πρέπει να υπολογίσθεί είναι οι χρονικές στιγμές t1, t2, t3,... ή αλλίως τα χρονικά διαστήματα $\delta t_0 = t1-t0$, $\delta t_1 = t2-t1$, $\delta t_2 = t3-t2$....$\delta t_m = t_{m+1} - t_m$.

Ώμως επειδή $\boldsymbol{\delta t = ct_t = c / f_t}$ για να υπολογίσουμε την τιμή του χρονικού διαστήματος (χρονικής καθυστέρησης) $\boldsymbol{\delta t_i}$ μας αρκεί να υπολογίσουμε την τιμή της αντίστοιχης εσωτερικής καθυστέρησης $\boldsymbol{c_i}$.

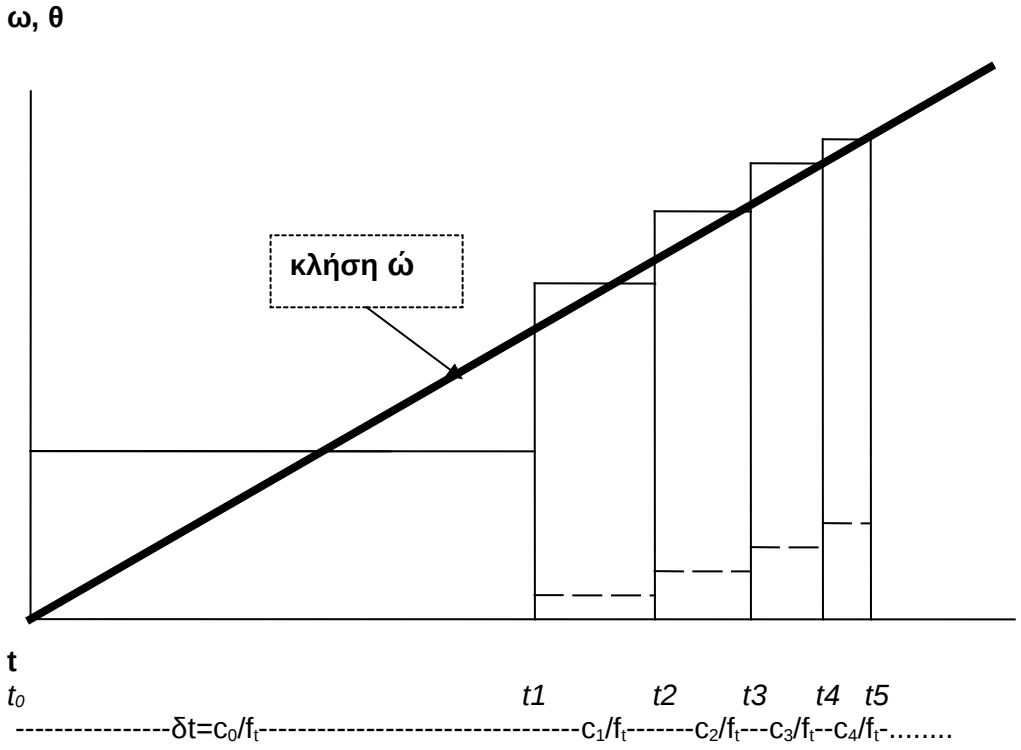
Αυτό γίνεται με μια αναδρομική μέθοδο η οποία υπολογίζει την επόμενη τιμή της εσωτερικής καθυστέρησης **c** με βάση την προηγούμενη τιμή της. Μεταβολή στην τιμή του **c** θα έχει ως αποτέλεσμα την μεταβολή του χρονικού διαστήματος **δt** (μεταβολή η οποία ουσιαστικά ελέγχει την κίνηση του κινητήρα), ενώ τα διαδοχικά χρονικά διαστήματα **δt** υπολογίζονται έτσι ώστε ο κινητήρας να εκτελεί επιταχυνόμενη κίνηση υπό σταθερή επιτάχυνση **ώ**.

Το παρακάτω σχήμα 3.4.3 δείχνει την σχέση μεταξύ επιτάχυνσης (***ώ***), ταχύτητας (***ω***) και θέσης (***θ***).



Σχήμα 3.4.3 – Επιτάχυνση, ταχύτητα και θέση άξονα βηματικού κινητήρα

Παρατηρούμε ότι σταθερή επιτάχυνση / επιβράδυνση έχει ως αποτέλεσμα ένα γραμμικό προφίλ ταχύτητας. Η γραφική παράσταση δηλαδή της επιτάχυνσης έχει μορφή ράμπας. Όταν η επιτάχυνση δεν είναι σταθερή το προφίλ ταχύτητας είναι εκθετικού τύπου.
Η επίδραση που έχει η μεταβολή της τιμής **c** στο χρονισμό των παλμών φαίνεται στο παρακάτω σχήμα 3.4.4:



Σχήμα 3.4.4 – Γραμμικό προφίλ ταχύτητας

Παρατηρούμε ότι οι τιμές $\mathbf{c_1}$, $\mathbf{c_2}$,.... διαμορφώνουν τα χρονικά διαστήματα δt μεταξύ των παλμών με τέτοιο τρόπο ώστε να έχουμε μια γραμμική (σταθερή) επιτάχυνση.

Η τιμή της συχνότητας $\mathbf{f_t}$ θα πρέπει να είναι αρκετά υψηλή αλλά ταυτόχρονα να επιτρέπει και υψηλές τιμές στην interstep delay ιδίως στην περίπτωση που ο κινητήρας εκκινεί από στάση. Μπορούμε να χρησιμοποιήσουμε τιμή f_t = 1MHz. Τότε ταχύτητα περιστροφής 300 rpm αντιστοιχεί σε interstep delay: c = SPS_timer_register = 1000

Η γωνία βήματος ***α*** του κινητήρα δίνεται από τη σχέση ***α = 2π / spr*** [rad] όπου **spr** είναι ο αριθμός των βημάτων για μια πλήρη περιστροφή. Η γωνιακή ταχύτητα ***ω*** δίνεται από τη σχέση ***ω = α / δt = α*** $\boldsymbol{f_t}$ ***/ c*** [rad/sec] για σταθερή (κάθε φορά) τιμή του **c** και η θέση ***θ*** του άξονα από την ***θ = nα*** [rad] όπου ***n*** ο αριθμός των βημάτων που παράγονται από την παλμοσειρά που δίνει ο χρονιστής_A.
Για παράδειγμα σε κινητήρα με ***spr*** = 200 έχουμε ***α*** = 1.8 μοίρες = 0,0314 rad,ενώ για f_t =8MHz και c = 20000 έχουμε:
ω=0,0314*8000000 / 20000 = 12,56 rad/sec = 120rpm=400 steps/sec .

Η (σταθερή) επιτάχυνση **ώ** [rad/sec^2] εκφράζεται και ως συνάρτηση των διαδοχικών τιμών $\mathbf{c_i}$, $\mathbf{c_{i+1}}$ και δίνεται από τη σχέση : ***ώ = [2α*** $\boldsymbol{f_t^2}$ ***(c1-c2)] / [c1 c2 (c1+c2)]***

Για γραμμική επιτάχυνση ισχύει **dώ / dt = 0** και έτσι για την ταχύτητα έχουμε **ω = ω(t) = ώt** όπου **t = δt** το χρονικό διάστημα στο οποίο συμβαίνει η αύξηση της ταχύτητας και από όπου με ολοκλήρωση προκύπτει η σχέση ***θ =ώt² / 2 = nα*** [rad]

Έτσι λοιπόν όταν ο άξονας του κινητήρα βρίσκεται στην θέση **θ = nα** τότε είναι η στιγμή για την εφαρμογή του n-οστού παλμού της παλμοσειράς.

Για την χρονική αυτή στιγμή έχουμε ότι $t_n = ((2\,n\,\alpha) / \omega)^{1/2}$ και αντίστοιχα θα ισχύουν και στον επόμενο παλμό. Επομένως έχουμε:

$$\delta t = c / f_t = t_{n-1} - t_n = (2a/\omega)^{1/2} [(n+1)^{1/2} - n^{1/2}]$$

από όπου για n= 0 παίρνουμε:

$$c = c_0 = (1 / t_t) (2a/\omega)^{1/2}$$

και για n=1 παίρνουμε:

$$c = c_1 = (1 / t_t) (2a/\omega)^{1/2} [(n+1)^{1/2} - n^{1/2}] = c_0 [(n+1)^{1/2} - n^{1/2}] \mid n=1$$

Γενικά από τη σχέση $\delta t = c / f_t = t_{n-1} - t_n$ παίρνουμε $c_n = f_t (t_{n-1} - t_n)$ που αντιπροσωπεύει την ακριβή τιμή του c_n για να επιτευχθεί η σωστή καθυστέρηση μεταξύ των n και n+1 παλμών Οπότε οι τιμές για το c_0 καθώς και οι επόμενες τιμές c_n δίνονται από τις παρακάτω σχέσεις:

$$c_0 = (1 / t_t) (2a/\omega)^{1/2} \quad \text{και} \quad c_n = c_0 [(n+1)^{1/2} - n^{1/2}]$$

Για να αποφύγουμε τους υπολογισμούς των υψώσεων σε δύναμη χρησιμοποιούμε τον παρακάτω προσεγγιστικό τύπο:

$$c_n = c_{n-1} - [2c_{n-1} / (4n+1)]$$

που μας δίνει αναδρομικά τις τιμές των συντελεστών μέσω των οποίων υπολογίζουμε τη χρονική διάρκεια **δt** μεταξύ των παλμών του **n-1** και **n – οστού** βήματος.
Επειδή η συχνότητα παραμένει σταθερή, όσο μεταβάλλονται (μειώνονται) οι τιμές των c_i τόσο μικραίνει το **δt**, άρα τόσο πιο γρήγορα εφαρμόζεται ο επόμενος παλμός.

Έχοντας υπολογίσει τις χρονικές στιγμές που θα πρέπει να εφαρμόζονται οι παλμοί του χρονιστή επιτάχυνσης μπορούμε να υπολογίσουμε τις ταχύτητες *sps1*, *sps2*, *sps3* κλπ απευθείας από τη σχέση: $\omega = \alpha / \delta t$

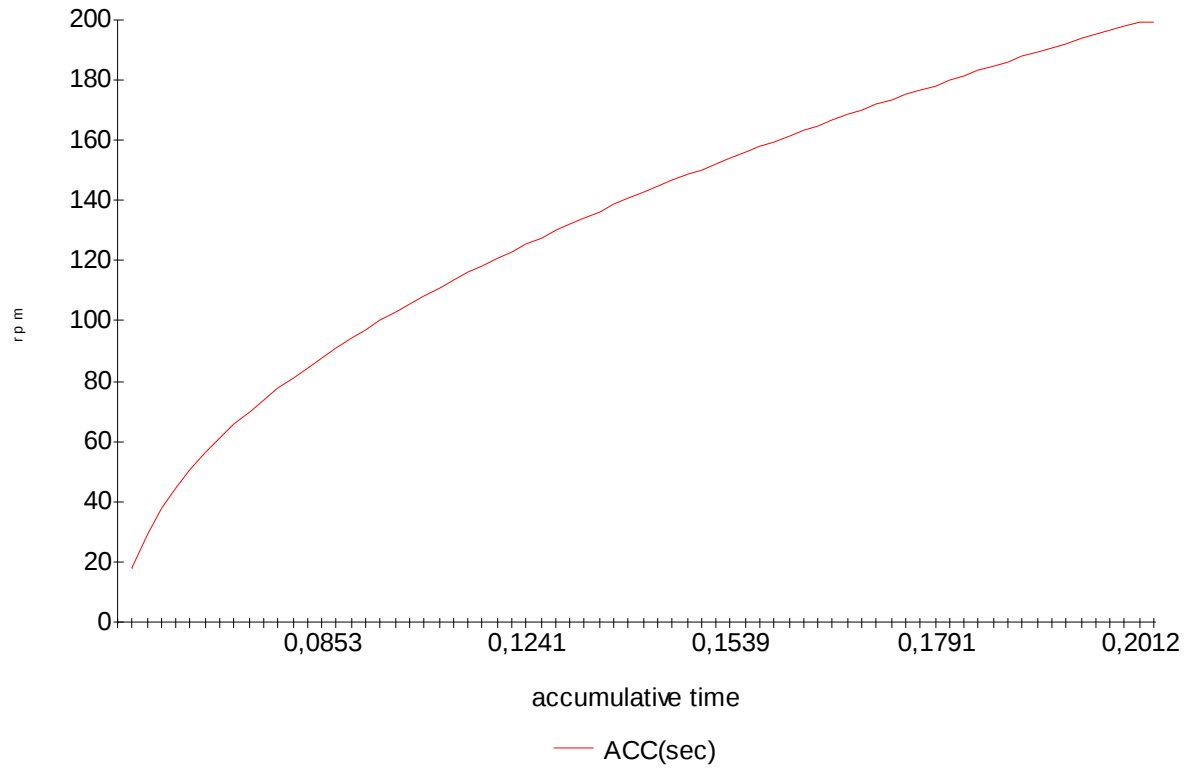
Η παραπάνω αριθμητική – αναδρομική μέθοδος καθορίζει το πότε θα δίνονται οι παλμοί του χρονιστή επιτάχυνσης και το κατά πόσο θα αυξάνεται η τιμή της ταχύτητας (πχ από sps1 σε sps2). Η αύξηση της ταχύτητας υπολογίζεται για κάθε βήμα του κινητήρα κατά τη διάρκεια της ράμπας επιτάχυνσης. Η διαδικασία αυτή επαναλαμβάνεται μέχρι η ταχύτητα του κινητήρα να φτάσει την επιθυμητή τιμή. Μόλις αυτό συμβεί το προφίλ επιτάχυνσης τελειώνει και η κίνηση συνεχίζει με σταθερή ταχύτητα. Η τιμή του **c** στην περίπτωση αυτή είναι η ελάχιστη.
Η όλη επαναληπτική διαδικασία μπορεί να υλοποιηθεί στο πλαίσιο μιας ρουτίνας εξυπηρέτησης διακοπών (ISR) του χρονιστή επιτάχυνσης. Όταν χρειάζεται υπολογισμός της νέας τιμής του **c**, ο έλεγχος του προγράμματος μεταφέρεται στην ρουτίνα εξυπηρέτησης διακοπών η οποία επιστρέφει την νέα τιμή του **c**.

Το προφίλ της ταχύτητας βοηθάει στην σχεδίαση του συστήματος. Ουσιαστικά αποτελεί μια απεικόνιση (ιδανική) του τρόπου με τον οποίο πρόκειται να μετακινηθεί ο κινητήρας μέχρι να

φτάσει στην επιθυμητή θέση. Το αν θα γίνει αυτό και στην πραγματικότητα εξαρτάται από την χρονική καθυστέρηση **δt** μεταξύ των παλμών που ελέγχει την ταχύτητα και η οποία χρονική καθυστέρηση πρέπει να υπολογιστεί με τέτοιον τρόπο ώστε η ταχύτητα να “ακολουθεί” το αντίστοιχο προφίλ όσο το δυνατόν περισσότερο.

Στο παρακάτω σχήμα 3.4.5 φαίνεται το προφίλ ταχύτητας κατά τη φάση της επιτάχυνσης του κινητήρα που χρησιμοποιούμε στο σύστημα μας που είναι ο SM-42BYG011-25 της εταιρίας mercury motors.
Η ταχύτητα start / stop του κινητήρα είναι 170 RPM ενώ η ταχύτητα pull out είναι 200 RPM.
Η προδιαγραφή που θέτουμε για την κίνηση του κινητήρα είναι να φτάσει την ταχύτητα pull out ξεκινώντας από στάση σε χρονικό διάστημα 0,20 sec. Η προδιαγραφή που θέτουμε για το χρόνο μας καθορίζει την επιτάχυνση που θα χρησιμοποιήσουμε.Ταχύτητα 200 RPM ισούται με 20,9 rad/sec. ‘Αρα για να ολοκληρωθεί η κίνηση σε 0,20 sec θα πρέπει η επιτάχυνση να είναι 100 rad/sec/sec. Περιορισμός είναι η εκκίνηση να μην γίνεται σε ταχύτητα μεγαλύτερη από την ταχύτητα start / stop .Έτσι έχουμε:

f_t = 1000000 Hz
ώ =100 rad/sec/sec (θέλουμε η επιτάχυνση να ολοκληρωθεί σε 0,20 sec)



Σχήμα 3.4.4 – Προφίλ ταχύτητας κατά την επιτάχυνση

Ο κινητήρας με βάση την παραπάνω μέθοδο εκκινεί σε ταχύτητα χαμηλότερη από την ταχύτητα start / stop και συνεχίζει να επιταχύνει μέχρι να φτάσει στην ταχύτητα pull out. Αυτό συμβαίνει μετά από 70 βήματα και έπειτα από χρόνο 0,2027 sec. Στην στήλη “perstep” βλέπουμε τις διαδοχικές τιμές του **c**. Η τιμή c0 είναι 16941 που σημαίνει ότι για κάθε 16941 παλμούς του master clock (1MHz) δίνεται και ένας παλμός που αντιστοιχεί σε προώθηση του άξονα του κινητήρα κατά 1 βήμα. Η επόμενη τιμή c1 είναι 10164 και η ελάχιστη είναι 1492.

spr=	200
f:=	1MHz
a=	0,0314
acceleration(rad/s/s):=	100

accumulative time (sec)	***δt(sec)***	***n***	***elapsed***	***perstep Ci***	***rad/ sec***	***RPM***	***steps/ sec***
0,0169	0,01694	0		**16.941**	1,854	**18**	59
0,0271	0,01016	1	16.941	**10.164**	3,091	**30**	98
0,0350	0,00791	2	27.105	**7.906**	3,974	**38**	126
0,0417	0,00669	3	35.010	**6.689**	4,696	**45**	149
0,0476	0,00590	4	41.700	**5.902**	5,322	**51**	169
0,0529	0,00534	5	47.602	**5.340**	5,882	**56**	187
0,0579	0,00491	6	52.942	**4.913**	6,394	**61**	204
0,0624	0,00457	7	57.855	**4.574**	6,868	**66**	219
0,0667	0,00430	8	62.429	**4.297**	7,311	**70**	233
0,0708	0,00406	9	66.726	**4.065**	7,728	**74**	246
0,0747	0,00387	10	70.791	**3.866**	8,125	**78**	259
0,0784	0,00369	11	74.658	**3.695**	8,503	**81**	271
0,0819	0,00354	12	78.352	**3.544**	8,864	**85**	282
0,0853	0,00341	13	81.896	**3.410**	9,212	**88**	293
0,0886	0,00329	14	85.306	**3.290**	9,547	**91**	304
0,0918	0,00318	15	88.596	**3.183**	9,871	**94**	314
0,0949	0,00308	16	91.779	**3.085**	10,184	**97**	324
0,0979	0,00300	17	94.863	**2.995**	10,488	**100**	334
0,1008	0,00291	18	97.859	**2.913**	10,784	**103**	343
0,1036	0,00284	19	100.772	**2.837**	11,071	**106**	352
0,1064	0,00277	20	103.609	**2.767**	11,351	**108**	361
0,1091	0,00270	21	106.377	**2.702**	11,625	**111**	370
0,1117	0,00264	22	109.079	**2.642**	11,892	**114**	379
0,1143	0,00258	23	111.720	**2.585**	12,153	**116**	387
0,1168	0,00253	24	114.305	**2.531**	12,409	**119**	395
0,1193	0,00248	25	116.837	**2.481**	12,660	**121**	403
0,1218	0,00243	26	119.318	**2.434**	12,906	**123**	411
0,1241	0,00239	27	121.752	**2.389**	13,147	**126**	419
0,1265	0,00235	28	124.141	**2.347**	13,384	**128**	426
0,1288	0,00231	29	126.488	**2.307**	13,617	**130**	433
0,1311	0,00227	30	128.795	**2.269**	13,846	**132**	441
0,1333	0,00223	31	131.064	**2.233**	14,071	**134**	448
0,1355	0,00220	32	133.297	**2.198**	14,292	**136**	455
0,1377	0,00216	33	135.495	**2.165**	14,511	**139**	462
0,1398	0,00213	34	137.660	**2.133**	14,725	**141**	469
0,1419	0,00210	35	139.793	**2.103**	14,937	**143**	476
0,1440	0,00207	36	141.896	**2.074**	15,146	**145**	482
0,1460	0,00205	37	143.970	**2.046**	15,352	**147**	489
0,1480	0,00202	38	146.016	**2.019**	15,556	**149**	495
0,1500	0,00199	39	148.036	**1.994**	15,756	**150**	502
0,1520	0,00197	40	150.029	**1.969**	15,955	**152**	508
0,1539	0,00195	41	151.998	**1.945**	16,150	**154**	514
0,1559	0,00192	42	153.943	**1.922**	16,344	**156**	520
0,1578	0,00190	43	155.865	**1.900**	16,535	**158**	526
0,1596	0,00188	44	157.765	**1.878**	16,724	**160**	532
0,1615	0,00186	45	159.644	**1.858**	16,911	**161**	538
0,1633	0,00184	46	161.501	**1.838**	17,096	**163**	544
0,1652	0,00182	47	163.339	**1.818**	17,278	**165**	550
0,1670	0,00180	48	165.157	**1.799**	17,459	**167**	556

0,1687	0,00178	49	166.956	**1.781**	17,638	**168**	561
0,1705	0,00176	50	168.737	**1.763**	17,816	**170**	567
0,1722	0,00175	51	170.500	**1.746**	17,991	**172**	573
0,1740	0,00173	52	172.246	**1.729**	18,165	**173**	578
0,1757	0,00171	53	173.976	**1.713**	18,337	**175**	584
0,1774	0,00170	54	175.689	**1.697**	18,508	**177**	589
0,1791	0,00168	55	177.386	**1.682**	18,677	**178**	595
0,1807	0,00167	56	179.068	**1.667**	18,844	**180**	600
0,1824	0,00165	57	180.735	**1.652**	19,010	**182**	605
0,1840	0,00164	58	182.388	**1.638**	19,175	**183**	610
0,1857	0,00162	59	184.026	**1.624**	19,338	**185**	616
0,1873	0,00161	60	185.650	**1.611**	19,500	**186**	621
0,1889	0,00160	61	187.261	**1.598**	19,660	**188**	626
0,1904	0,00158	62	188.859	**1.585**	19,820	**189**	631
0,1920	0,00157	63	190.444	**1.572**	19,978	**191**	636
0,1936	0,00156	64	192.016	**1.560**	20,134	**192**	641
0,1951	0,00155	65	193.577	**1.548**	20,290	**194**	646
0,1967	0,00154	66	195.125	**1.537**	20,444	**195**	651
0,1982	0,00153	67	196.661	**1.525**	20,597	**197**	656
0,1997	0,00151	68	198.187	**1.514**	20,749	**198**	661
0,2012	0,00150	69	199.701	**1.503**	20,900	**200**	665
0,2027	0,00149	70	201.204	**1.492**	21,050	**201**	670

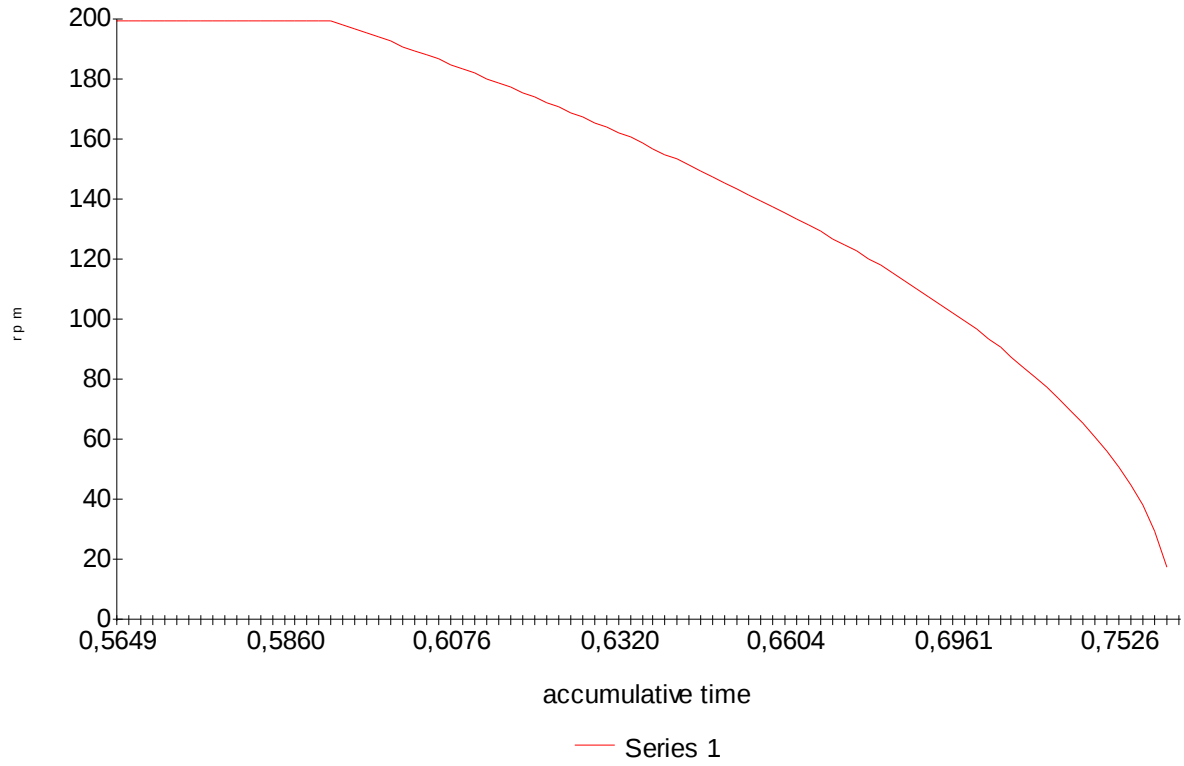
Η σχέση $c_0 = 1 / t_t (2a/ώ)^{1/2}$ συνδέει την επιτάχυνση **ώ** με το c_0 και με το **n** που σημαίνει ότι η ακρίβεια στον υπολογισμό του c_0 παίζει σημαντικό ρόλο για το εάν η παλμοσειρά που θα προκύψει θα είναι τέτοια που όντως θα επιταχύνει τον κινητήρα με επιτάχυνση **ώ.**
Ο παρακάτω πίνακας δείχνει ότι η επιτάχυνση μπορεί να θεωρηθεί ότι διατηρείται σταθερή και ίση με την τιμή που θέλαμε. Αυτό σημαίνει ότι η κίνηση του κινητήρα ακολουθεί με πολύ μεγάλη ακρίβεια το γραμμικό προφίλ ταχύτητας άρα με βάση το προφίλ αυτό μπορούμε να καθορίσουμε ακριβώς τη συμπεριφορά του κινητήρα στο σύστημα.

n	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
ώ	**91,18**	**97,69**	**98,96**	**99,41**	**99,62**	**99,73**	**99,80**	**99,85**	**99,88**	**99,90**	**99,92**	**99,93**
n	*13*	*14*	*15*	*16… 18*	*19… 22*	*23...33*	*34… 58*	*59*	*60*	*61*	*62*	*63*
ώ	**99,94**	**99,95**	**99,95**	**99,96**	**99,97**	**99,98**	**99,99**	**99,99**	**99,99**	**99,99**	**99,99**	**99,99**
n	*64*	*65*	*66*	*67*	*68*	*69*	*70*	***ώ = [2α f_t^2 (c1-c2)] / [c1 c2 (c1+c2)]***				
ώ	**99,99**	**99,99**	**99,99**	**99,99**	**99,99**	**99,99**	**99,99**					

Στο παρακάτω σχήμα 3.4.5 βλέπουμε το προφίλ για τη φάση της επιβράδυνσης. Για να υπολογίσουμε τα χρονικά διαστήματα μεταξύ δυο διαδοχικών παλμών χρησιμοποιούμε την παρακάτω εξίσωση:

$$c_i = c_{i-1} - [2c_{i-1} / (4(i-m)+1)], \quad i<m$$

που είναι μια παραλλαγή της σχέσης που χρησιμοποιούμε για την επιτάχυνση. Η σχέση για την επιβράδυνση έχει το χαρακτηριστικό ότι μπορεί να δέχεται αρνητικές τιμές για τον αριθμό των βημάτων. Εδώ αντί για **n** χρησιμοποιούμε την ποσότητα **i-m** με **i<m** όπου **m** είναι ο συνολικός αριθμός των βημάτων και όπου **i** είναι το βήμα στο οποίο θα ξεκινήσει η διαδικασία της επιβράδυνσης. Επειδή η επιβράδυνση θεωρείται ίση με την επιτάχυνση η ράμπα της επιβράδυνσης είναι ίδια με την ράμπα της επιτάχυνσης που σημαίνει ότι το **i** θα είναι ίδιο με το n που είδαμε προηγουμένως αλλά μετρώντας από το τέλος της διαδρομής. Για παράδειγμα στην φάση της επιτάχυνσης αυτή τελείωσε στο βήμα **n = 70**, τώρα το i θα είναι το βήμα **#330**.



Σχήμα 3.4.5 – Προφίλ ταχύτητας κατά την επιβράδυνση

spr=	200
f:=	1MHz
a=	0,0314
acceleration(rad/s/s):=	100

accumulative time (sec)	***δt(sec)***	***n***	***elapsed***	***perstep***	***rad/ sec***	***RPM***	***steps/ sec***
0,5638	0,00149	312	562.268	**1.492**	21,055	**201**	670
0,5653	0,00149	313	563.760	**1.492**	21,055	**201**	670
0,5667	0,00149	314	565.252	**1.492**	21,055	**201**	670
0,5682	0,00149	315	566.744	**1.492**	21,055	**201**	670
0,5697	0,00149	316	568.236	**1.492**	21,055	**201**	670
0,5712	0,00149	317	569.728	**1.492**	21,055	**201**	670
0,5727	0,00149	318	571.220	**1.492**	21,055	**201**	670
0,5742	0,00149	319	572.712	**1.492**	21,055	**201**	670
0,5757	0,00149	320	574.204	**1.492**	21,055	**201**	670
0,5772	0,00149	321	575.696	**1.492**	21,055	**201**	670
0,5787	0,00149	322	577.188	**1.492**	21,055	**201**	670
0,5802	0,00149	323	578.680	**1.492**	21,055	**201**	670
0,5817	0,00149	324	580.172	**1.492**	21,055	**201**	670
0,5832	0,00149	325	581.664	**1.492**	21,055	**201**	670
0,5846	0,00149	326	583.156	**1.492**	21,055	**201**	670
0,5861	0,00149	327	584.648	**1.492**	21,055	**201**	670
0,5876	0,00149	328	586.140	**1.492**	21,055	**201**	670

0,5891	0,00149	329	587.632	**1.492**	21,055	**201**	670
0,5906	0,00150	330	589.124	1.503	20,905	**200**	665
0,5921	0,00151	331	590.627	1.514	20,754	**198**	661
0,5937	0,00152	332	592.140	1.525	20,602	**197**	656
0,5952	0,00154	333	593.665	1.536	20,449	**195**	651
0,5967	0,00155	334	595.201	1.548	20,294	**194**	646
0,5983	0,00156	335	596.749	1.560	20,139	**192**	641
0,5999	0,00157	336	598.309	1.572	19,982	**191**	636
0,6015	0,00158	337	599.881	1.585	19,824	**189**	631
0,6031	0,00160	338	601.466	1.597	19,665	**188**	626
0,6047	0,00161	339	603.063	1.611	19,504	**186**	621
0,6063	0,00162	340	604.674	1.624	19,343	**185**	616
0,6079	0,00164	341	606.298	1.638	19,179	**183**	611
0,6096	0,00165	342	607.936	1.652	19,015	**182**	605
0,6113	0,00167	343	609.588	1.667	18,849	**180**	600
0,6129	0,00168	344	611.254	1.682	18,681	**178**	595
0,6146	0,00170	345	612.936	1.697	18,512	**177**	589
0,6163	0,00171	346	614.633	1.713	18,341	**175**	584
0,6181	0,00173	347	616.346	1.729	18,169	**174**	578
0,6198	0,00175	348	618.075	1.746	17,995	**172**	573
0,6216	0,00176	349	619.820	1.763	17,820	**170**	567
0,6234	0,00178	350	621.583	1.781	17,642	**168**	562
0,6252	0,00180	351	623.364	1.799	17,463	**167**	556
0,6270	0,00182	352	625.162	1.818	17,282	**165**	550
0,6288	0,00184	353	626.980	1.837	17,099	**163**	544
0,6307	0,00186	354	628.817	1.857	16,915	**162**	538
0,6326	0,00188	355	630.674	1.878	16,728	**160**	532
0,6345	0,00190	356	632.552	1.899	16,539	**158**	526
0,6364	0,00192	357	634.452	1.922	16,348	**156**	520
0,6383	0,00194	358	636.373	1.945	16,154	**154**	514
0,6403	0,00197	359	638.318	1.968	15,958	**152**	508
0,6423	0,00199	360	640.287	1.993	15,760	**151**	502
0,6443	0,00202	361	642.280	2.019	15,559	**149**	495
0,6463	0,00205	362	644.299	2.046	15,356	**147**	489
0,6484	0,00207	363	646.344	2.074	15,150	**145**	482
0,6505	0,00210	364	648.418	2.103	14,941	**143**	476
0,6527	0,00213	365	650.521	2.133	14,729	**141**	469
0,6548	0,00216	366	652.653	2.164	14,514	**139**	462
0,6570	0,00220	367	654.818	2.197	14,296	**137**	455
0,6592	0,00223	368	657.015	2.232	14,074	**134**	448
0,6615	0,00227	369	659.247	2.268	13,849	**132**	441
0,6638	0,00231	370	661.516	2.306	13,620	**130**	434
0,6662	0,00235	371	663.822	2.347	13,387	**128**	426
0,6686	0,00239	372	666.169	2.389	13,150	**126**	419
0,6710	0,00243	373	668.557	2.434	12,909	**123**	411
0,6735	0,00248	374	670.991	2.481	12,663	**121**	403
0,6760	0,00253	375	673.472	2.531	12,412	**119**	395
0,6786	0,00258	376	676.003	2.584	12,156	**116**	387
0,6812	0,00264	377	678.587	2.641	11,895	**114**	379
0,6839	0,00270	378	681.228	2.702	11,628	**111**	370
0,6867	0,00277	379	683.929	2.767	11,354	**108**	361
0,6895	0,00284	380	686.696	2.837	11,074	**106**	353
0,6924	0,00291	381	689.533	2.912	10,786	**103**	343
0,6954	0,00299	382	692.445	2.994	10,490	**100**	334
0,6985	0,00308	383	695.440	3.084	10,186	**97**	324
0,7017	0,00318	384	698.524	3.182	9,873	**94**	314
0,7050	0,00329	385	701.705	3.290	9,549	**91**	304
0,7084	0,00341	386	704.995	3.409	9,214	**88**	293
0,7119	0,00354	387	708.404	3.543	8,866	**85**	282
0,7156	0,00369	388	711.947	3.694	8,505	**81**	271
0,7195	0,00387	389	715.641	3.866	8,127	**78**	259
0,7236	0,00406	390	719.507	4.064	7,730	**74**	246
0,7279	0,00430	391	723.570	4.296	7,312	**70**	233
0,7324	0,00457	392	727.866	4.573	6,869	**66**	219

0,7374	0,00491	393	732.439	4.912	6,395	**61**	204
0,7427	0,00534	394	737.351	5.339	5,884	**56**	187
0,7486	0,00590	395	742.690	5.901	5,323	**51**	169
0,7553	0,00669	396	748.591	6.688	4,697	**45**	150
0,7632	0,00790	397	755.279	7.904	3,975	**38**	127
0,7733	0,01016	398	763.183	10.162	3,091	**30**	98
0,7903	0,01694	399	773.345	16.937	1,855	**18**	59
0,7733	-0,01694	400	790.282	-16.937	-1,855	**-18**	-59

Στο **Παράρτημα 1** φαίνεται ολόκληρο το προφίλ ταχύτητας.

Εάν θέλουμε να αλλάξει το βήμα στο οποίο ο κινητήρας θα φτάσει την επιθυμητή ταχύτητα τότε θα πρέπει να αλλάξουμε την τιμή της επιτάχυνσης. Αυτό συμβαίνει όταν η προδιαγραφή που έχουμε δεν μας δίνει το χρονικό διάστημα για να φτάσει ο κινητήρας την επιθυμητή ταχύτητα αλλά την απόσταση.
Για παράδειγμα θέλουμε ο κινητήρας να φτάσει την ταχύτητα pull out σε 53 βήματα.

Η (νέα) χρονική στιγμή t_n που θα εκτελεσθεί το βήμα #53 είναι συνάρτηση των **ώ**, **ω** και **θ**.

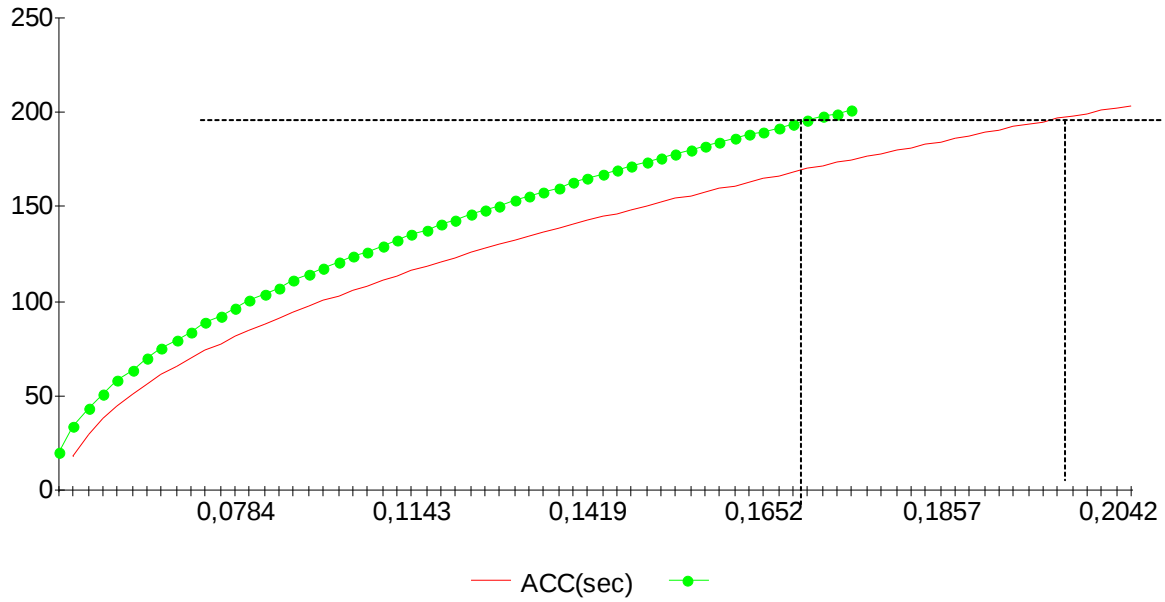
$t_n = \omega_n / \dot{\omega}$ και $n = \dot{\omega} t^2_n / 2\alpha$ από όπου προκύπτει

$n\dot{\omega} = \omega^2/2\alpha$

Η σχέση $n\dot{\omega} = \omega^2/2\alpha$ δείχνει ότι εάν η ταχύτητα **ω** είναι η επιθυμητή ταχύτητα που θέλουμε να πιάσει ο κινητήρας (ιδανικά η ταχύτητα pull out) τότε ο αριθμός των βημάτων που χρειάζονται για να φτάσουμε στην ταχύτητα αυτή είναι αντιστρόφως ανάλογος της νέας επιτάχυνσης.

$n_1 \dot{\omega}_1 = n_2 \dot{\omega}_2$

Είδαμε προηγουμένως ότι ο συγκεκριμένος κινητήρας φτάνει την ταχύτητα pull out σε 70 βήματα όταν η επιτάχυνση είναι 100 rad/sec/sec. Με τη νέα προδιαγραφή που έχουμε θέλουμε να φτάσει την ίδια ταχύτητα σε 53 βήματα. Η παραπάνω σχέση τότε μας δίνει την τιμή $\dot{\omega}_2$ = 130 rad/sec/sec. Και εδώ βασικός περιορισμός είναι ο κινητήρας να μην εκκινεί σε ταχύτητα μεγαλύτερη από την ταχύτητα start / stop.
Στο παρακάτω σχήμα 3.4.6 βλέπουμε το προφίλ επιτάχυνσης για την περίπτωση αυτή. Με το πράσινο χρώμα βλέπουμε την απεικόνιση της κίνησης του κινητήρα για κίνηση με επιτάχυνση ώ2 = 130 rad/sec/sec ενώ με το κόκκινο χρώμα βλέπουμε την κίνηση όταν η επιτάχυνση ήταν ώ1 = 100 rad/sec/sec. Παρατηρούμε ότι ο κινητήρας έφτασε την επιθυμητή ταχύτητα σε χρόνο που αντιστοιχεί στην εκτέλεση του βήματος #53 σύμφωνα με την προδιαγραφή μας.



Σχήμα 3.4.6 – Μεταβολή του προφίλ ταχύτητας κατά την επιτάχυνση

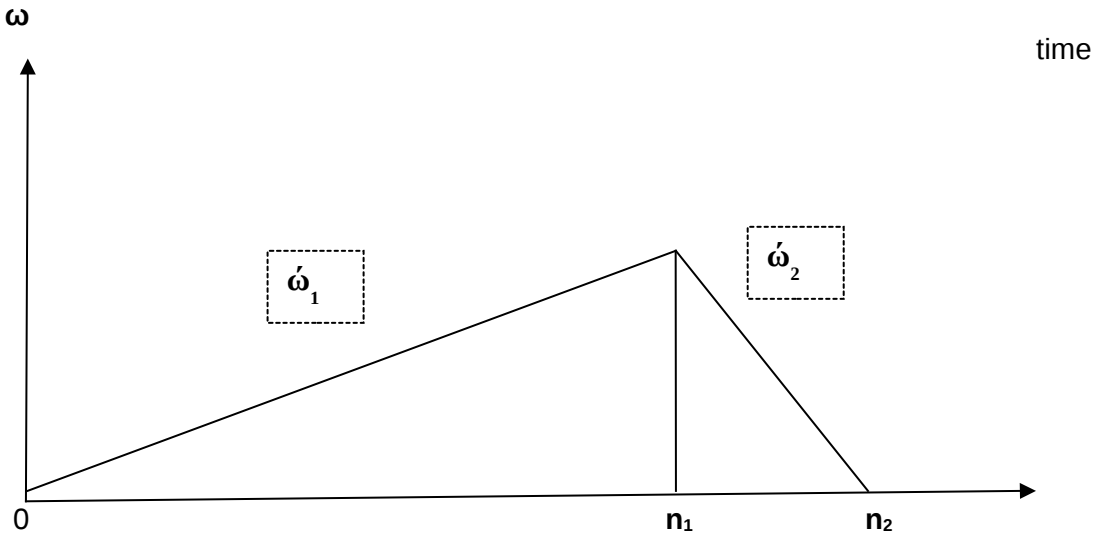
Η ίδια σχέση μπορεί να χρησιμοποιηθεί και για τον υπολογισμό των βημάτων για την ράμπα της επιβράδυνσης στην περίπτωση που θέλουμε η ράμπα της επιβράδυνσης να είναι διαφορετική από αυτήν της επιτάχυνσης. Αν δηλαδή ισχύει ότι η επιτάχυνση είναι $\dot{\omega}_1$ και η επιβράδυνση είναι $\dot{\omega}_2$ με $\dot{\omega}_2 \neq \dot{\omega}_1$ και με δεδομένο ο κινητήρας έφτασε την επιθυμητή ταχύτητα ω υπό επιτάχυνση $\dot{\omega}_1$ στο βήμα # n_1. Συνήθως επιλέγουμε $\dot{\omega}_2 > \dot{\omega}_1$ με την ράμπα της επιβράδυνσης $\dot{\omega}_2$ να έχει μεγαλύτερη κλίση.
Έτσι αν θέλουμε από επιβράδυνση (= επιτάχυνση) ω_1 να πάμε σε επιβράδυνση ω_2 με τιμή τέτοια ώστε η ράμπα επιβράδυνσης να έχει μεγαλύτερη κλίση, τότε το βήμα n_2 στο οποίο θα ξεκινήσει η επιβράδυνση δίνεται από την παρακάτω σχέση:

$$n_1 = (n_1 + n_2)\, \dot{\omega}_2 / (\dot{\omega}_1 + \dot{\omega}_2)$$

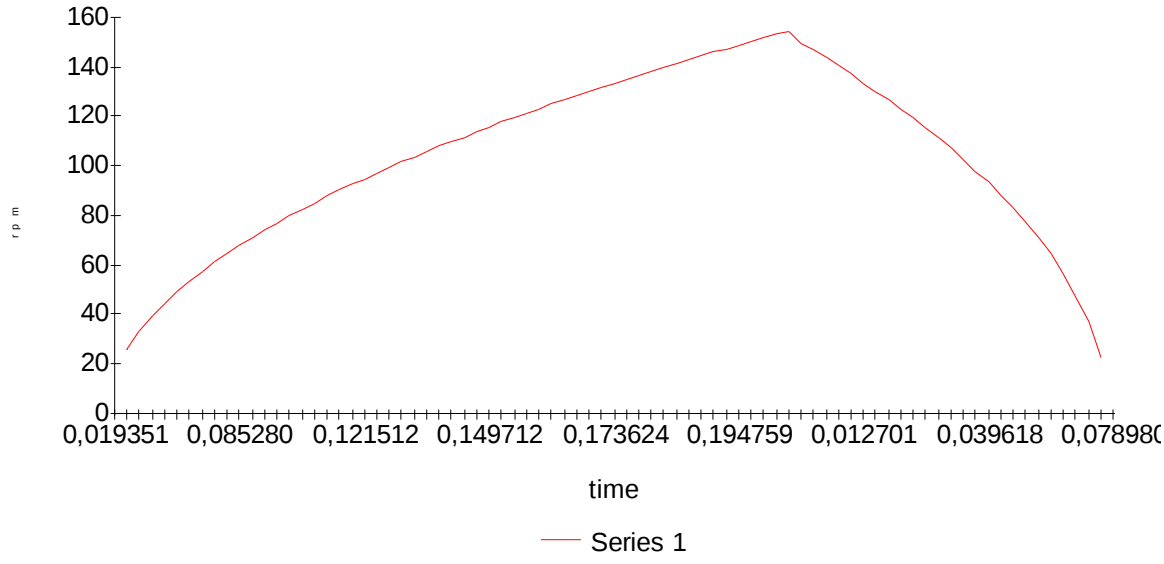
όπου n_1 και n_2 είναι ο αριθμός των βημάτων που χρειάζονται για τις ράμπες επιτάχυνσης και επιβράδυνσης αντίστοιχα.
Έτσι για παράδειγμα αν έχουμε επιτάχυνση = επιβράδυνση = 100 rad /sec/sec η ράμπα της επιτάχυνσης θέλει 70 βήματα όπως είδαμε και από το προφίλ επιτάχυνσης προηγουμένως, ο ίδιος αριθμός βημάτων χρειάζεται και για τη ράμπα της επιβράδυνσης. Εάν θέλουμε η επιβράδυνση να γίνει 200 rad / sec / sec τότε το βήμα από το οποίο θα ξεκινήσει η ράμπα της επιβράδυνσης είναι το βήμα #35 και η τιμή αυτή επαληθεύει τις παραπάνω σχέσεις.

Στην περίπτωση που ο συνολικός αριθμός m των βημάτων μέχρι το στόχο είναι μικρός τότε η φάση της επιτάχυνσης δεν θα προλάβει να ολοκληρωθεί καθώς θα πρέπει να ακολουθήσει η επιβράδυνση. Το βήμα n που αυτό θα συμβεί δίνεται από τη σχέση:

$$n = m\dot{\omega}_2 / \dot{\omega}_1 + \dot{\omega}_2.$$



Στο παρακάτω σχήμα φαίνεται ένα προφίλ ταχύτητας για την περίπτωση αυτή.



Σχήμα 3.4.6 – Προφίλ ταχύτητας όταν η επιτάχυνση είναι διαφορετική από την επιβράδυνση

Εδώ πρέπει να σημειώσουμε ότι η μόνη προϋπόθεση που θέτουμε είναι ο κινητήρας να σταματήσει σε μια ταχύτητα χαμηλότερη από την ταχύτητα start / stop έτσι ώστε όταν θα σταματήσει να μην έχει χάσει βήματα.
Εάν τα αποτελέσματα αυτά δεν καλύπτουν τις προδιαγραφές τις εφαρμογής μας θα πρέπει να αλλάξουμε τις παραμέτρους εισόδου.
Είδαμε προηγουμένως την περίπτωση να έχουμε προδιαγραφή τέτοια που να θέλουμε ο κινητήρας να φτάσει την ταχύτητα pull out αλλά καλύπτοντας λιγότερη απόσταση από αυτή που αντιστοιχεί στα 70 βήματα. Ακόμα είδαμε ότι αυτό μπορούμε να το πετύχουμε αλλάζοντας

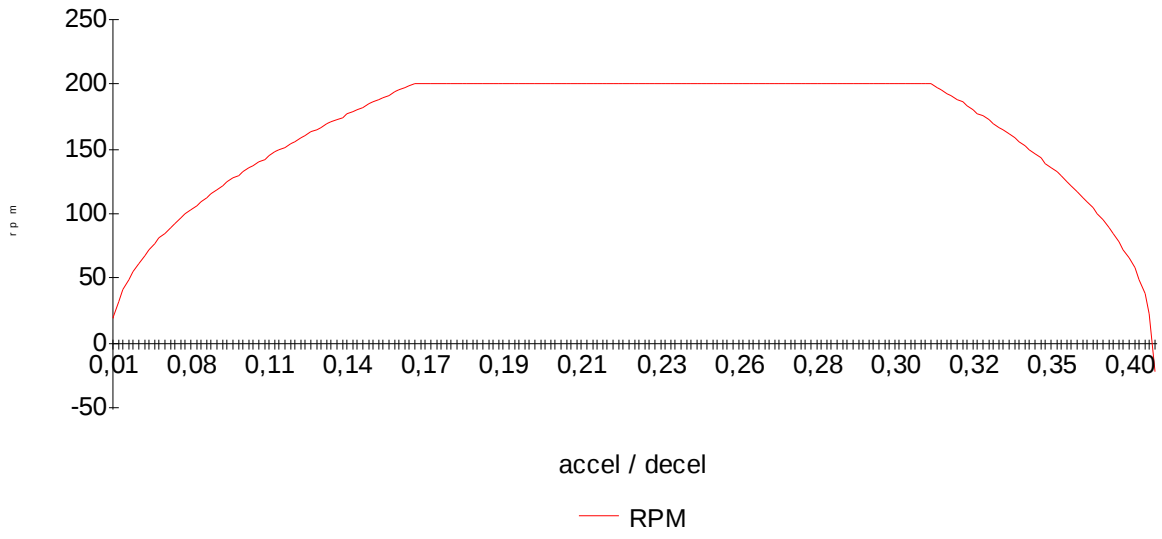
την επιτάχυνση. Τότε όμως ο κινητήρας θα διανύσει μεν απόσταση σύμφωνα με την προδιαγραφή που έχουμε, θα κάνει όμως και λιγότερο χρόνο πράγμα που μπορεί να μην το θέλουμε εάν για παράδειγμα χρειάζεται να περιμένουμε κάποια άλλη κίνηση του συστήματος να ολοκληρωθεί όπως η κίνηση ενός ρομποτικού βραχίονα.

Έτσι λοιπόν ένα πρόβλημα είναι εάν η εκτέλεση ενός προκαθορισμένου αριθμού βημάτων πρέπει να ολοκληρωθεί εντός προκαθορισμένου χρονικού διαστήματος.
Σε μια τέτοια περίπτωση θα πρέπει να εξασφαλίσουμε ότι ο κινητήρας θα σταματήσει αφού έχουν εκτελεσθεί όλα τα βήματα που έχουμε προγραμματίσει να εκτελεσθούν και ότι αυτό θα συμβεί σε καθορισμένη χρονική στιγμή. Για να το πετύχουμε αυτό χρειάζεται να υπολογίσουμε το πότε θα πρέπει να ξεκινήσει η επιβράδυνση.
Έστω ότι έχουμε σαν προδιαγραφή ο κινητήρας να διανύσει ακριβώς 200 βήματα σε χρονικό διάστημα ισο προς 0,42 sec (με μια ανοχή +5%έως -5%)
Έστω επίσης ότι αλλάζουμε την επιτάχυνση από 100 rad/sec/sec σε 130 rad/sec/sec.
Αυτό σημαίνει ότι τώρα η επιτάχυνση ολοκληρώνεται σε 58 βήματα.
Εάν η επιβράδυνση ήταν 130 rad/sec/sec θα έπρεπε να ξεκινήσει στο βήμα #142.
Από το προφίλ ταχύτητας βλέπουμε ότι από την έναρξη της επιτάχυνσης μέχρι το βήμα αυτό έχουν παρέλθει 0,2937 sec. Εμείς θα θέλαμε η κίνηση να ολοκληρωθεί σε 0,42sec ή σε χρονικό διάστημα ΔΤ = 0,41 – 0,2937 = 0,1163 sec
Μέσα σε αυτό το χρονικό διάστημα θέλουμε η ταχύτητα από 21rpm ιδανικά να μηδενιστεί.
Άρα θέλουμε επιβράδυνση ω_2 ίση προς 21 / 0,1163 = 181 rad/sec/sec
Από τη σχέση $n_1 \omega_1 = n_2 \omega_2$ παίρνουμε ότι ώ2 = (58*130) / 181 = 42. Αυτό σημαίνει ότι η ράμπα της επιβράδυνσης θα πρέπει να ξεκινήσει 42 βήματα πρίν ολοκληρωθεί η κίνηση.
Συνολικά θέλουμε 200 βήματα άρα η ράμπα της επιβράδυνσης θα πρέπει να ξεκινήσει στο βήμα # 158.
Από τη σχέση

$$c_i = c_{i-1} - [2c_{i-1} / (4(i-m)+1)],$$

αν θέσουμε όπου m = 42 προκύπτει ο χρονισμός των παλμών κατά την επιβράδυνση.

Το προφίλ της κίνησης φαίνεται στο παρακάτω σχήμα 3.4.7:



Σχήμα 3.4.7 – Προφίλ ταχύτητας εντός προκαθορισμένου χρονικού διαστήματος

Στο βήμα #198 ο κινητήρας έχει ταχύτητα 38rpm, στο βήμα #199 ;έχει ταχύτητα 23 rpm και η ταχύτητα μηδενίζεται μεταξύ του βήματος #199 και του βήματος #200. Ο χρόνος που έχει παρέλθει είναι 0,44 sec, επομένως η κίνηση δεν ανταποκρίνεται στην προδιαγραφή μας. Το πρόβλημα εντοπίζεται στην πολύ μεγάλη διάρκεια των δύο τελευταίων βημάτων.

Είδαμε προηγουμένως μια αριθμητική μέθοδο που μπορούμε να την χρησιμοποιήσουμε για τον έλεγχο της θέσης του κινητήρα στην πορεία του προς την θέση – στόχο. Το ζήτημα είναι στην πράξη ο κινητήρας να μην αποκλίνει από το καθορισμένο προφίλ ταχύτητας ή να αποκλίνει εντός συγκεκριμένων διαστημάτων ανοχής.

Με το παρακάτω πρόγραμμα μπορούμε να παρακολουθούμε την πορεία της δημιουργίας του προφίλ ταχύτητας από τον μικροεπεξεργαστή. Η έξοδος του προγράμματος μας δίνει την αύξηση του χρονικού διαστήματος καθώς ο κινητήρας επιταχύνει βήμα – βήμα προς την καθορισμένη ταχύτητα. Έτσι μπορούμε άμεσα να υπολογίσουμε το χρονικό διάστημα που μεσολαβεί μεταξύ δυο διαδοχικών βημάτων, να κατασκευάσουμε το “πραγματικό” προφίλ ταχύτητας και συγκρίνοντας τα δυο προφίλ να βγάλουμε συμπεράσματα για το πώς θα συμπεριφερθεί στην πραγματικότητα ο κινητήρας.

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 2);

void forwardstep() {
  motor1.onestep(FORWARD, DOUBLE);
}
void backwardstep() {
  motor1.onestep(BACKWARD, DOUBLE);
}
AccelStepper stepper(forwardstep, backwardstep); // use functions to step
float spd = 30.0;
long maxspd =650;

long accel=3000;
//float reqspd=sqrt(2.0 * dist * accel);
//float step_interval = abs(1000.0 / reqspd);
float lastStepTime;
float timer;
void setup()
{
  Serial.begin(9600); // set up Serial library at 9600 bps
  Serial.println("Stepper test!");
  //Serial.println(step_interval);
  long dist=400;
  float reqspd;
  stepper.setMaxSpeed(maxspd);
  stepper.setSpeed(reqspd);
  stepper.setAcceleration(accel);
  stepper.moveTo(dist);

  for(int i=1;i<dist;i++){
    reqspd=sqrt(2.0 * dist * accel);
    //Serial.println(reqspd);
```

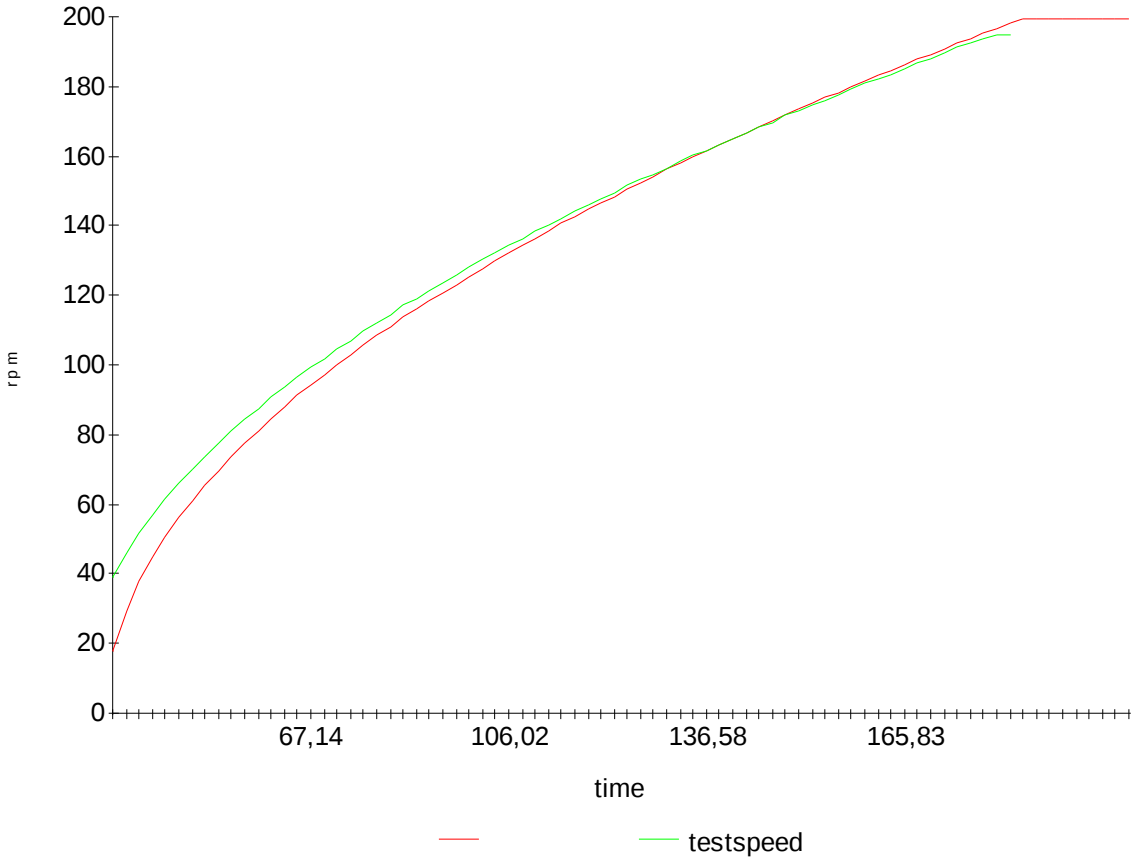
```
    if (reqspd > spd)
    {
      // Need to accelerate in clockwise direction
      if (spd == 0)
        reqspd = sqrt(2.0 * accel);
      else
        reqspd = spd + abs(accel / spd);

      if (reqspd > maxspd)
        reqspd = maxspd;
      spd = reqspd;

      float step_interval = abs(1000.0 / spd);
      timer = lastStepTime + step_interval;
      lastStepTime = timer;
      Serial.println(timer);
      //Serial.println(spd);
    }
    dist=dist-1;
  }
}
void loop(){

  while(stepper.currentPosition() != 400){
    stepper.runToPosition();
  }

}
```

Έτσι εισάγοντας στο πρόγραμμα παραμέτρους ακριβώς ίδιες με αυτές για τον θεωρητικό υπολογισμό του προφίλ ταχύτητας παίρνουμε τα παρακάτω αποτελέσματα.

“Θεωρητικός” συνολικός χρόνος (msec)	“Θεωρητική” ταχύτητα (steps/sec)	# βήματος	“Πραγματικός” συνολικός χρόνος (msec)	“Πραγματική” ταχύτητα (steps/sec)
16,94	59	0		130
27,10	98	1	7,69	153
35,01	126	2	14,22	172
41,70	149	3	20,02	189
47,60	169	4	25,28	205
52,94	187	5	30,14	220
57,86	204	6	34,67	234
62,43	219	7	38,95	246
66,73	233	8	43	258
70,79	246	9	46,86	270
74,66	259	10	50	281
78,35	271	11	54,1	292
81,90	282	12	57,53	302
85,31	293	13	60,83	312
88,60	304	14	64,03	322
91,78	314	15	67,14	331
94,86	324	16	70,15	340
97,86	334	17	73,09	349
100,77	343	18	75,95	357
103,61	352	19	78,75	366

106,38	361	20	81,48	374
109,08	370	21	84,15	382
111,72	379	22	86,76	390
114,31	387	23	89,33	397
116,84	395	24	91,84	405
119,32	403	25	94,3	412
121,75	411	26	96,73	420
124,14	419	27	99,11	427
126,49	426	28	101,45	434
128,80	433	29	103,75	441
131,06	441	30	106,02	448
133,30	448	31	108,25	454
135,49	455	32	110,45	461
137,66	462	33	112,61	467
139,79	469	34	114,75	474
141,90	476	35	116,86	480
143,97	482	36	118,94	486
146,02	489	37	121	492
148,04	495	38	123	499
150,03	502	39	125	505
152,00	508	40	127	511
153,94	514	41	129	516
155,87	520	42	131	522
157,77	526	43	132,81	528
159,64	532	44	134,7	534
161,50	538	45	136,58	539
163,34	544	46	142,08	545
165,16	550	47	143,88	550
166,96	556	48	145,66	556
168,74	561	49	147,42	561
170,50	567	50	149,17	566
172,25	573	51	150,9	572
173,98	578	52	152,62	577
175,69	584	53	154,32	582
177,39	589	54	156	587
179,07	595	55	157,68	592
180,74	600	56	159,33	598
182,39	605	57	160,98	603
184,03	610	58	162,61	608
185,65	616	59	164,23	612
187,26	621	60	165,83	617
188,86	626	61	167,43	622
190,44	631	62	169	627
192,02	636	63	170,58	632
193,58	641	64	172,14	637
195,12	646	65	173,68	641
196,66	651	66	175,22	646
198,19	656	67	176,76	650
199,70	656	68	178,3	650

Έτσι για παράδειγμα βλέπουμε ότι θεωρητικά στο χρονικό διάστημα που μεσολαβεί μεταξύ του βήματος #52 και του #53 είναι 1,73 msec ενώ ο χρονιστής του μικροεπεξεργαστή παράγει τους δυο διαδοχικούς παλμούς με καθυστέρηση 1,70 msec. Επίσης στο #53 βήμα η ταχύτητα είναι 584 και 582 rpm αντίστοιχα.
Μπορούμε να εξάγουμε το συμπέρασμα ότι ο μικροεπεξεργαστής κατασκευάζει ένα προφίλ ταχύτητας που είναι αρκετά κοντά στον θεωρητικό υπολογισμό του.
Στο παρακάτω σχήμα 3.4.8 με κόκκινο χρώμα είναι το προφίλ που κατασκευάζει η αριθμητική μέθοδος ενώ με το πράσινο χρώμα είναι το προφίλ που παράγεται από τον επεξεργαστή.



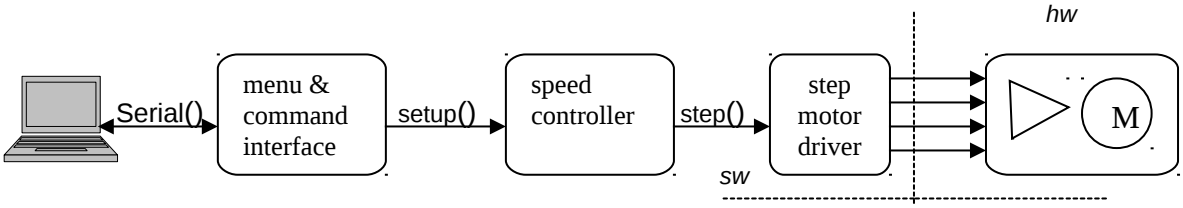
Σχήμα 3.4.8 – Θεωριτικό και "πραγματικό" προφίλ ταχύτητας

3.5 Γραμμικός έλεγχος ταχύτητας

Στην παράγραφο αυτή θα αναφερθούμε στα βασικά τμήματα που αποτελούν έναν γραμμικό ελεγκτή ταχύτητας ανοικτού βρόχου μέσω του οποίου ελέγχουμε τον αριθμό των βημάτων, την μέγιστη ταχύτητα, την επιτάχυνση και την επιβράδυνση. Το προφίλ ταχύτητας είναι στην περίπτωση αυτή γραμμικό. Η υλοποίηση θα γίνει σε software που θα τρέχει σε 8 bit μικροεπεξεργαστή και βασίζεται στην αριθμητική μέθοδο που είδαμε στην προηγούμενη παράγραφο.
Τα βασικά τμήματα που αποτελούν έναν ελεγκτή ταχύτητας είναι τα εξής:

- Ένα πολύ βασικό interface για να μπορεί ο χρήστης να επικοινωνεί με τον μικροεπεξεργαστή μέσω usb θύρας. Μέσω της θύρας αυτής ο χρήστης μπορεί να δίνει τις σχετικές εντολές και ο μικροεπεξεργαστής να οδηγεί τον κινητήρα αναλόγως.
- Συναρτήσεις που υπολογίζουν τα απαραίτητα κάθε φορά δεδομένα και παράγουν την παλμοσειρά ώστε η περιστροφή του κινητήρα να γίνεται ακολουθώντας ένα γραμμικό προφίλ ταχύτητας.
- Μέτρηση των εκτελούμενων βημάτων βάσει της οποίας δίδονται στον κινητήρα κατάλληλα σήματα ελέγχου όπως σήματα για εναλλαγή στη διάγερση των φάσεων, για έναρξη και στάση κλπ.

Παρακάτω (Σχήμ 3.5.1) φαίνεται το βασικό block διάγραμμα του ελεγκτή.
Η επικοινωνία του χρήστη με τον μικροεπεξεργαστή γίνεται μέσω των συναρτήσεων της βιβλιοθήκης Serial() και τα αποτελέσματα της επικοινωνίας φαίνονται στο ειδικό σειριακό παράθυρο. Κατόπιν η συνάρτηση setup() εκτελεί τους αναγκαίους υπολογισμούς και ενεργοποιεί τις συναρτήσεις που υπολογίζουν και παράγουν το προφίλ ταχύτητας. Οι συναρτήσεις αυτές χρησιμοποιούν δεδομένα που έχουν προ – υπολογισθεί από την setup() και καλούν τη συνάρτηση step() η οποία μετακινεί τον άξονα του κινητήρα ένα βήμα κάθε φορά.



Σχήμα 3.5.1 – Γραμμικός ελεκτής ταχύτητας ανοικτού βρόχου

Για τον έλεγχο του κινητήρα τέσσερις παράμετροι που περιγράφουν το προφίλ ταχύτητας χρειάζονται. Οι τιμές των παραμέτρων αυτών ορίζονται από το χρήστη και είναι οι εξής:

— *steps*: Αριθμός των βημάτων που θα μετακινηθεί ο άξονας του κινητήρα
— *accel*: Επιτάχυνση
— *decel*: Επιβράδυνση
— *speed*: Μέγιστη ταχύτητα περιστροφής

Ανάλογα με τις τιμές των παραμέτρων για ταχύτητα, απόσταση, επιτάχυνση, επιβράδυνση μπορούμε να διακρίνουμε τις περιπτώσεις:

1. η επιτάχυνση ολοκληρώνεται και ο κινητήρας περιστρέφεται με την επιλεγμένη σταθερή ταχύτητα
2. η επιτάχυνση δεν ολοκληρώνεται και πρέπει να ξεκινήσει η επιβράδυνση χωρίς ο κινητήρας να φτάσει στην επιλεγμένη ταχύτητα.

Οι παραπάνω περιπτώσεις “κωδικοποιούνται” ως εξής:
Υπολογίζουμε τον αριθμό των βημάτων **n** μέχρι να ολοκληρωθεί η επιτάχυνση.

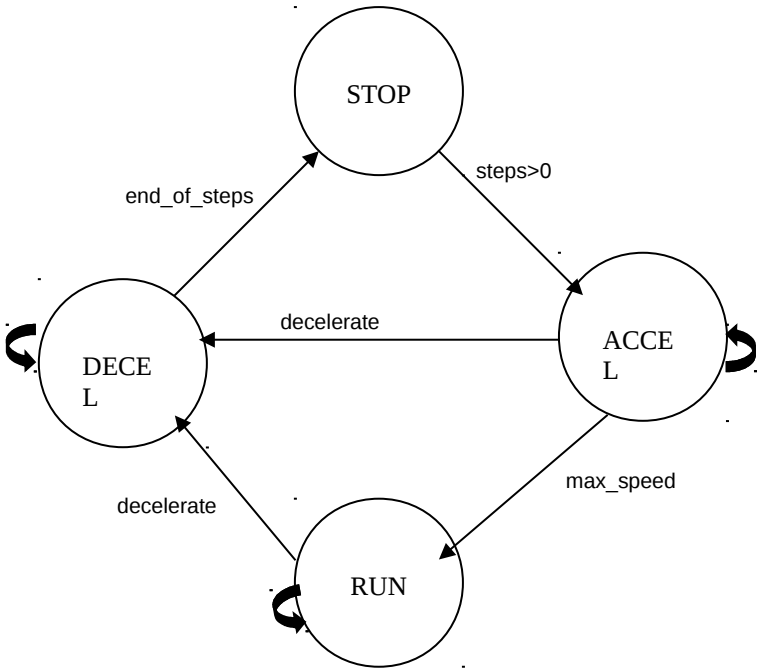
n = (*speed***^2) / (2 x a x** *accel* **x 100)**

Υπολογίζουμε τον αριθμό του βήματος **n1** που, ανεξαρτήτως ταχύτητας, θα πρέπει να ξεκινήσει η ράμπα της επιβράδυνσης, προκειμένου η κίνηση να ολοκληρωθεί στον προκαθορισμάνο αριθμό βημάτων.

n1* = (steps* ***x*** *decel***) / (** *accel* **+** *decel* **)**

Εάν **n < n1** η επιτάχυνση τερματίζει όταν ο κινητήρας φτάσει την επιθυμητή ταχύτητα *speed*. Τότε η ράμπα της επιβράδυνσης ξεκινάει στο βήμα με αριθμό: ***n2 = -n1 x (****accel* ***/*** *decel****).***
Εάν **n>n1** η επιβράδυνση θα πρέπει να ξεκινήσει χωρίς να έχει προλάβει να ολοκληρωθεί η επιτάχυνση. Τότε ***n2 = - (****steps* ***– n1)***

Η συνάρτηση setup() λαμβάνει τις τιμές των παραμέτρων και προ – υπολογίζεί: Την τιμή **c** η οποία αντιστοιχεί στην ελάχιστη καθυστέρηση μεταξύ των παλμών που εφαρμόζεται όταν η περιστροφή του κινητήρα γίνεται με σταθερή ταχύτητα. Την τιμή **c0** που όπως είδαμε αντιστοιχεί στην χρονική καθυστέρηση μεταξύ πρώτου και δεύτερου παλμού κατά τη διάρκεια της οποίας εκτελείται το πρώτο βήμα. Την τιμή **n1** και την τιμή **n2**.
Οι τιμές αυτές χρησιμοποιούνται από τις συναρτήσεις που κατασκευάζουν το προφίλ ταχύτητας και που αποτελούν το module “speed controller”.
Μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης setup(), ο έλεγχος μεταφέρεται στον speed controller ο οποίος υλοποιείται ως μια μηχανή πεπερασμένων καταστάσεων (finite state machine - fsm) όπως φαίνεται στο παρακάτω σχήμα 3.5.2. Το αποτέλεσμα είναι η παραγωγή της παλμοσειράς που θα κινήσει τον κινητήρα σύμφωνα με το επιθυμητό προφίλ ταχύτητας. Σε κάθε παλμό έχουμε την κλίση της συνάρτησης step() που προωθεί τον άξονα του κινητήτα κατά ένα βήμα. Η κλίση της step() γίνεται μέσω μιας ρουτίνας εξυπηρέτησης διακοπών (ISR) και κατόπιν ο έλεγχος μεταφέρεται πάλι στην μηχανή καταστάσεων για την παραγωγή του επόμενου παλμού.



Σχήμα 3.5.2 – Μηχανή πεπερασμένων καταστάσεων για υλοποίηση της βαθμίδας speed controller

Όταν η συνάρηση setup() ολοκληρώσει την εκτέλεσή της ο έλεγχος μεταφέρεται στην κατάσταση STOP. Εάν η παράμετρος *steps* είναι > 0 τότε έχουμε μετάβαση στην κατάσταση ACCEL. Στην κατάσταση αυτή έχουμε την επιτάχυνση του κινητήρα έως ότου είτε η κίνηση γίνει με την επιθυμητή ταχύτητα (παράμετρος *speed*), οπότε ο κινητήρας μεταβαίνει στην κατάσταση RUN, είτε θα πρέπει να ξεκινήσει η επιβράδυνση όποτε η επόμενη κατάσταση είναι η DECEL. Για όσο ο κινητήρας βρίσκεται στην RUN, η κίνηση γίνεται με σταθερή ταχύτητα μέχρι που να ξεκινήσει η επιβράδυνση οπότε και γίνεται η μετάβαση στην κατάσταση DECEL. Ο κινητήρας θα παραμείνει στην κατάσταση αυτή έως ότου εκτελεσθούν όλα τα προγραμματισμένα βήματα. Τη στιγμή εκείνη η κατάσταση επιστρέφει στην STOP.
Η κάθε κατάσταση υλοποιείται από την αντίστοιχη συνάρτηση που τρέχει στο εσωτερικό της. Στα πλαίσια της συνάρτησης αυτής γίνονται οι απαραίτητοι υπολογισμοί και έλεγχοι, οι οποίοι επαναλαμβάνονται για κάθε βήμα κατά τη διάρκεια της επιτάχυνσης / κίνησης με σταθερή ταχύτητα / επιβράδυνσης (real time approach). Το αποτέλεσμα είναι ο υπολογισμός της νέας καθυστέρησης που πρέπει να εφαρμοστεί για την πραγματοποίηση του επόμενου βήματος. Στη συνέχεια γίνεται η κλίση της συνάρτησης step( ), το βήμα εκτελειται και κατόπιν ο έλεγχος επιστρέφει είτε στην ίδια είτε στην επόμενη κατάσταση. Για παράδειγμα η κατάσταση DECEL υλοποιείται από την συνάρτηση statedrag( ). Η συνάρτηση αυτή χρησιμοποιεί την τιμή **c** που υπολογίστηκε από την setup( ), υπολογίζει την επόμενη χρονική καθυστέρηση αναδρομικά με βάση την προηγούμενη καθυστέρηση, σύμφωνα με την αριθμητική μέθοδο της προηγούμενης παραγράφου, μετράει τα βήματα που απομένουν μέχρι το τέλος και σε κάθε βήμα καλεί την συνάρτηση step( ) για να το εκτελέσει.
Κάθε κλίση της step( ) ενεργοποιεί τον step motor driver που είναι το τμήμα εκείνο του προγράμματος που παράγει τα σήματα για την σωστή εναλλαγή της διεγερσης των φάσεων ανάλογα με τον επιλεγμένο τρόπο διέγερσης. Υποστιρίζονται full step με one και two phase on, half step και microstepping.

Το παρακάτω πρόγραμμα αποτελεί μια πολύ βασική προσέγγιση (demo) των παραπάνω διαδικασιών για συγκεκριμένη απόσταση, μέγιστη ταχύτητα, επιτάχυνση και επιβράδυνση.

```
#include <AFMotor.h>
#include <AccelStepper.h>
#include <math.h>

//speed = 190 rpm
//distance = 600 steps
//accel = 30 rad/s/s
//deccel = 30 rad/s/s
AF_Stepper motor1(200,2);
float a=0.0314;
double f1 = 1000000;
float c;// minimum delay
float c0; // first step delay
long y;
double x;
long int n,n1,n2;
long int n3;

typedef enum {
  STOP, RUN, DECEL, ACCEL}
states;

states state = STOP;
unsigned int currentValue;

void s
```
setup( ){

```
  Serial.begin(9600);

  c = (a*f1*100) /  1990; // 190 rpm = 634 steps/sec = 19,9 rad/sec --> 1990 (= rad/sec*100)
  y = 2*a*10000000;
  x=y*1000;
  c0 = ((0.676*1000000)/100)*(sqrt(x/3000)/100);// c0, acceleration = 30 r/s/s --> 3000
  n=(19.9*19.9)/(2*a*30*1);// ypologismos "n"
  n1=(600*30)/(60); // ypologismos "n1"   acceleration =30, deceleration = 30 r/s/s
  Serial.print("c = ");
  Serial.print(c);
  Serial.println( );
  Serial.print("c0= ");
  Serial.print(c0);
  Serial.println( );
  Serial.print("n= ");
  Serial.print(n);
  Serial.println( );
  Serial.print("n1= ");
  Serial.print(n1);
  if(n < n1){
    n2=(-n);         // ypologismos n2
    n2=n2;
    n3=-n2;         //  n3 = -n2
  }
  else{
    n2=-(400-n1);
    n3=-n2;
  }
  Serial.println( );
  Serial.print("n2= ");
  Serial.print(n2);
  Serial.println( );
  Serial.print("n3= ");
  Serial.print(n3);
  Serial.println( );
  state = STOP;     // metavasi stin arxiki katastasi
}

void statestop(const unsigned int value)
{
  //motor1.onestep(FORWARD,DOUBLE);
  motor1.release();
  Serial.println("Running...");
}



void stateaccel(const unsigned int value){
  float c1,c2,del,del2;
  int ac=1;
  int z=0;
  c1 = (c0 - ((2*c0) / (4*ac+1)));
  del=c1 / 1000; //ms
  delay(del);
  motor1.step(1,FORWARD,DOUBLE);
```

```
  // Serial.println(c1);

  for(int i=1;i<n;i++){
    ac=ac+1;
    c2=(c1-((2*c1)/(4*ac+1)));
    del2 = c2 / 1000;
    delay(del2);
    //Serial.println(c2);
    motor1.step(1,FORWARD,DOUBLE);
    c1=c2;
    //  z=z+1;

  }

  //  Serial.println(z);
  motor1.release( );
}

void staterun(const unsigned int value)
{
  float del3;
  int k=0;
  int m = 600-n3;
  //motor1.setSpeed(spd);


  for(int j=n;j<m;j++){

    del3=c/1000;
    delay(2);
    motor1.step(1,FORWARD,DOUBLE);


    //   k=k+1;
    //Serial.println(c);
  }
  //Serial.println(k);
  motor1.release( );

}

void statedrag(const unsigned int value){

  float c5,del2;
  int w=0;
  int x = 600-n3;
  for(int i=x;i<600;i++){

    c5=(c-((2*c)/((4*n2)+1)));

    del2 = c5 / 1000;
    delay(del2);
    //Serial.println(c5);
    motor1.step(1,FORWARD,DOUBLE);
```

```
    c=c5;
    n2=n2+1;
    //w=w+1;
   }
   // Serial.println(w);
   motor1.release( );
   Serial.println("DONE!");

}

void handlePreviousState(){
  switch (state){
  case STOP:
    statestop(currentValue);
    break;
  case ACCEL:
    stateaccel(currentValue);
    break;
  case RUN:
    staterun(currentValue);
    break;
  case DECEL:
    statedrag(currentValue);
    break;
   }

  currentValue = 0;
}


void processIncomingByte (const byte c){

  handlePreviousState( );
  switch(c){
  case 'S':
    state = STOP;
    break;
  case 'A':
    state = ACCEL;
    break;
  case 'R':
    state = RUN;
    break;
  case 'D':
    state = DECEL;
    break;
  default:
    state = STOP;
    break;
  }
}
void loop(){
  if (Serial.available ( ))
    processIncomingByte (Serial.read ( ));
}
```

Στο παρακάτω σχήμα 3.5.3 με κόκκινο χρώμα φαίνεται το προφίλ ταχύτητας που προκύπτει από την οδήγηση του κινητήρα με τον παραπάνω τρόπο ελέγχου για τις κάτωθι τιμές των παραμέτρων:

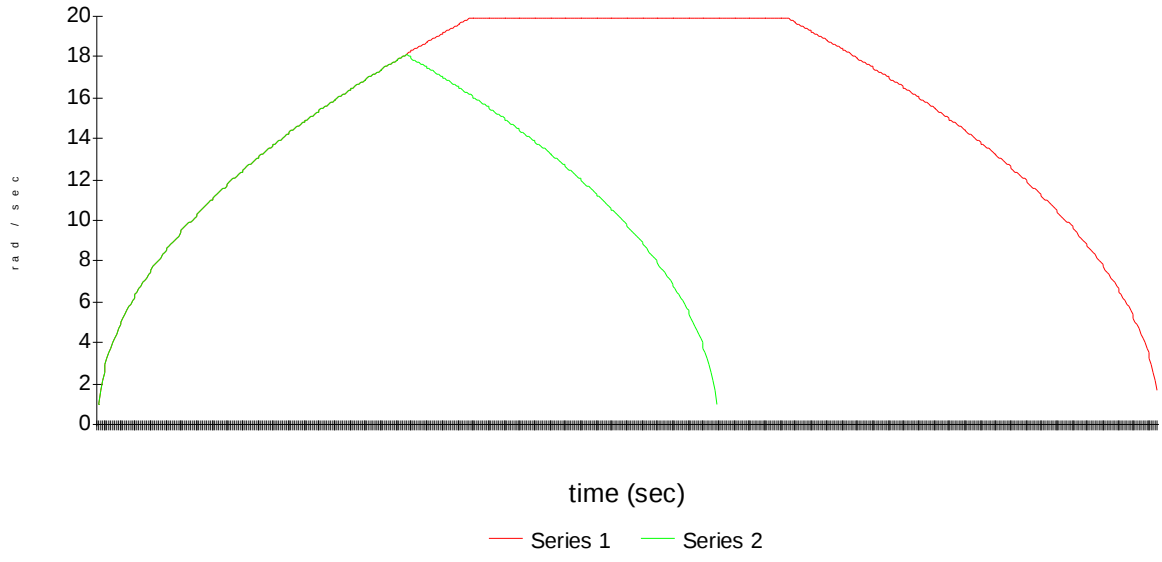
speed = 190 rpm
steps = 600 steps
accel = 30 rad/s/s
decel = 30 rad/s/s

Στην περίπτωση αυτή η ράμπα επιτάχυνσης ολοκληρώνεται και η κίνηση συνεχίζεται με σταθερή ταχύτητα ίση με την επιθυμητή ταχύτητα.

Με πράσινο χρώμα φαίνεται το προφίλ για τις παρακάτω τιμές.

speed = 190 rpm
steps = 350 steps
accel = 30 rad/s/s
decel = 30 rad/s/s

Στην περίπτωση αυτή η ράμπα επιτάχυνσης δεν ολοκληρώνεται και η επιβράδυνση ξεκινά πριν ο κινητήρας φτάσει την επιθυμητή ταχύτητα.

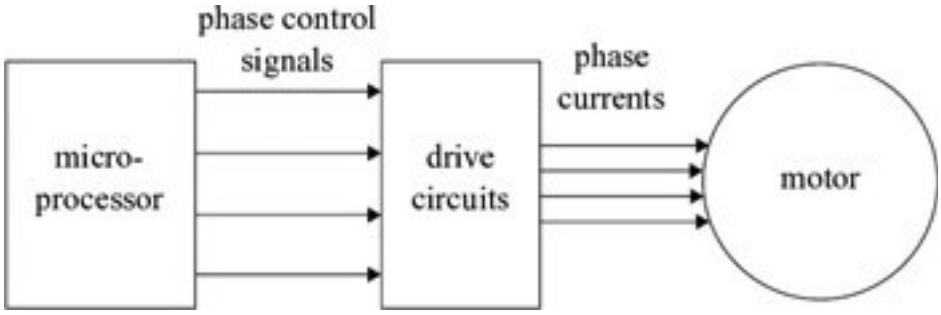


Σχήμα 3.5.3 – Οδήγηση βηματικού κινητήρα με τον γραμμικό ελεκτή ταχύτητας

4.Συστήματα βηματικού κινητήρα με μικροεπεξεργαστή

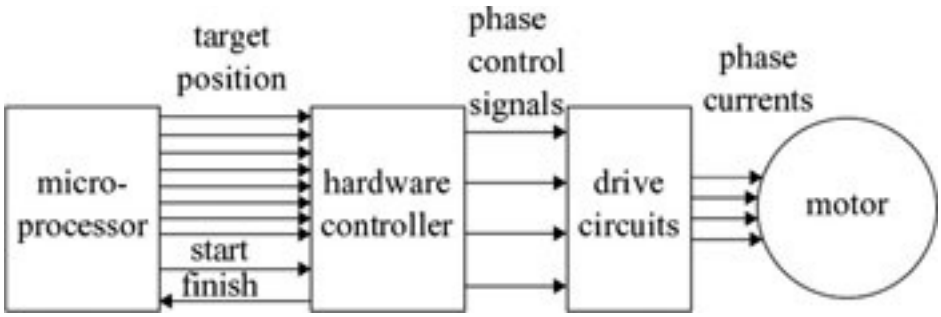
4.1 Εισαγωγή

Οι βηματικοί κινητήρες χρησιμοποιούνται συχνά σαν συσκευές εξόδου συστημάτων τα οποία ελέγχονται από μικροεπεξεργαστές. Για παράδειγμα η γραφίδα ενός εκτυπωτικού μηχανήματος (plotter) καθοδηγείται από βηματικούς κινητήρες, κατά τους άξονες X και Y, οι οποίοι με τη σειρά τους οδηγούνται και ελέγχονται από ένα σύστημα μικροεπεξεργαστή. Το βασικό χαρακτηριστικό αυτών των συστημάτων είναι το ότι ο μικροεπεξεργαστής εκτελεί ένα πρόγραμμα και ο βηματικός κινητήρας πρέπει να οδηγήσει το φορτίο του στη θέση εκείνη η οποία αντιστοιχεί στα αποτελέσματα της εκτέλεσης του προγράμματος του μικροεπεξεργαστή. Αυτό που διαφέρει είναι ο βαθμός «εμπλοκής» του μικροεπεξεργαστή στην όλη διαδικασία ελέγχου. Έτσι διακρίνουμε δυο βασικές κατηγορίες: συστήματα ελέγχου ανοικτού βρόχου τα οποία μπορούμε να τα κατονομάσουμε σαν “software – intensive” , συστήματα δηλαδή στα οποία στην όλη διαδικασία ελέγχου επικρατεί το λογισμικό (σχήμα 4.1.1) και συστήματα τα οποία μπορούμε να κατονομάσουμε σαν “hardware - intensive”, που είναι τα συστήματα εκείνα στα οποία στην διαδικασία ελέγχου επικρατεί το υλικό (σχήμα 4.1.2).
Βέβαια στην πράξη χρησιμοποιούνται και συστήματα ελέγχου που είναι σχεδιασμένα ώστε να αποτελούν συνδυασμό των δυο παραπάνω κατηγοριών, με άλλα συστήματα να υλοποιούν τις περισσότερες λειτουργίες ελέγχου βασιζόμενα στο software και άλλα βασιζόμενα στο hardware.



Σχήμα 4.1.1 – Σύστημα ελέγχου ανοικτού βρόχου software - intensive

Στα software – intensive συστήματα ο μικροεπεξεργαστής εκτελεί το πρόγραμμα το οποίο είναι υπεύθυνο τόσο για το χρονισμό όσο και για την αλληλουχία – διαδοχή των σημάτων, και σαν αποτέλεσμα της εκτέλεσης του προγράμματος παράγονται τα σήματα ελέγχου των φάσεων του κινητήρα (phase control signals). Τα σήματα αυτά τροφοδοτούν το κύκλωμα οδήγησης του κινητήρα και έτσι καθορίζουν το εάν και με ποια φορά θα περάσει ρεύμα από ποια φάση (τύλιγμα) του κινητήρα έτσι ώστε ο άξονάς του να μετακινηθεί στην καθορισμένη θέση.
Στα hardware – intensive συστήματα το πρόγραμμα που εκτελείται από τον μικροεπεξεργαστή μπορεί να χαρακτηριστεί σαν στοιχειώδες αφού η μόνη λειτουργία είναι να τροφοδοτήσει την επόμενη βαθμίδα του συστήματος (hardware controller) με την επιθυμητή θέση του άξονα (target position) και να δώσει την εντολή start. Τώρα ο hardware controller είναι αυτός που παράγει τα σήματα ελέγχου των φάσεων αλλά και το σήμα finish προς τον μικροεπεξεργαστή όταν ο άξονας του κινητήρα φτάσει στην επιθυμητή θέση.



Σχήμα 4.1.2 – Σύστημα ελέγχου ανοικτού βρόχου hardware – intensive

Σε συστήματα που αποτελούν συνδυασμό των παραπάνω μπορεί να έχουμε έλεγχο του χρονισμού ή / και του αν ο άξονας του κινητήρα έφτασε στην καθορισμένη θέση, αν εκτέλεσε δηλαδή των σωστό αριθμό βημάτων από software, ενώ οι λειτουργίες τις εναλλαγής στην διέγερση των φάσεων ή / και του καθορισμού της κατεύθυνσης περιστροφής να ελέγχονται από hardware.
Είναι φανερό ότι για να υλοποιήσουμε ένα σύστημα ελέγχου software – intensive χρειαζόμαστε περισσότερη υπολογιστική ισχύ. Στην περίπτωση που η ισχύς αυτή είναι διαθέσιμη τότε η λύση αυτή είναι η προτεινόμενη αφού έχει και σχετικά χαμηλό οικονομικό κόστος. Επίσης σχετικά εύκολα μπορούμε να αναπτύξουμε και το αναγκαίο software καθώς υπάρχουν διαθέσιμα αρκετά εργαλεία ανάπτυξης. Όμως στην περίπτωση που η υπολογιστική ισχύς που διαθέτουμε χρησιμοποιείται για την επιτήρηση και τον έλεγχο και άλλων συσκευών και ιδίως εάν ο έλεγχος γίνεται σε πραγματικό χρόνο (real time control), όπως συμβαίνει σε πολλά ενσωματωμένα συστήματα, τότε η λύση του hardware – intensive συστήματος ή ενός συστήματος του οποίου οι περισσότερες λειτουργίες να υλοποιούνται από hardware είναι η καλύτερη αφού υλοποιώντας τις συναρτήσεις ελέγχου που μας ενδιαφέρουν σε hardware έχουμε καλύτερο και αμεσότερο έλεγχο χωρίς περιορισμούς από το πρόγραμμα ή την απόδοση του μικροεπεξεργαστή.

4.2 Λειτουργία με σταθερή ταχύτητα

Εδώ ο κινητήρας λειτουργεί σε σταθερή ταχύτητα η οποία είναι χαμηλότερη από την ταχύτητα start / stop. Αυτός ο τρόπος λειτουργίας προτιμάται σε εφαρμογές στις οποίες δεν υπάρχει αυστηρή προδιαγραφή σε ότι αφορά τον χρόνο που θα κάνει ο κινητήρας για να φτάσει στην θέση στόχο.

Έστω το παρακάτω πρόγραμμα:

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 2);

void forwardstep() {
motor1.onestep(FORWARD, DOUBLE);
}
void backwardstep() {
motor1.onestep(BACKWARD, DOUBLE);
}
```

```
AccelStepper stepper(forwardstep, backwardstep);

void setup()
{
Serial.begin(9600);
Serial.println("Stepper test!");
}


void loop()
{
stepper.moveTo(1800);
stepper.setSpeed(150);        //steps per second
stepper.run();
}
```

Το πρόγραμμα αυτό χρησιμοποιεί συναρτήσεις ελέγχου των βιβλιοθήκών AFMotor και AccelSteper και υλοποιεί βασικές λειτουργίες που συναντάμε σε κάθε σύστημα ελέγχου ανοικτού βρόχου, τις ακόλουθες:

- Καθορισμός της ταχύτητας περιστροφής του κινητήρα. Βασική προϋπόθεση να είναι μικρότερη από την μέγιστη ταχύτητα εκκίνησης του κινητήρα (ταχύτητα start / stop).
- Καθορισμός του τρόπου διέγερσης των φάσεων.
- Μέτρηση των βημάτων που εκτελεί ο κινητήρας και παύση της λειτουργίας του όταν ο άξονας του κινητήρα φτάσει στην προκαθορισμένη θέση.

Κάθε μια από τις παραπάνω λειτουργίες ελέγχου μπορούν να υλοποιηθούν από ένα αμιγώς software σύστημα ή από ένα αμιγώς hardware σύστημα ή από σύστημα που αποτελεί συνδυασμό των παραπάνω.
Στο παραπάνω πρόγραμμα ο καθορισμός της σταθερής ταχύτητας περιστροφήςαλλά και της φοράς περιστροφής γίνεται από την συνάρτηση setSpeed(), ο καθορισμός της θέσης–στόχου γίνεται από την συνάρτηση moveTo() ενώ η μέτρηση του αριθμού των βημάτων προς την καθορισμένη θέση γίνεται από την συνάρτηση run() η οποία είναι η συνάρτηση που κινεί τον κινητήρα σύμφωνα και με τη φορά που ορίστηκε από την setSpeed().

Ο καθορισμός του τρόπου διέγερσης των φάσεων γίνεται από hardware και συγκεκριμένα από έναν καταχωρητή serial to parallel. Ο καταχωρητής δέχεται στην κύρια είσοδο του ένα byte που παράγεται από το πρόγραμμα και ανάλογα διεγείρει τις φάσεις του κινητήρα.
Επομένως από τις τρεις παραπάνω λειτουργίες ελέγχου οι δύο υλοποιούνται από λογισμικό και η μια από υλικό. Αυτό μπορούμε να το πούμε παρόλο που το byte που δέχεται στην είσοδό του ο καταχωρητής παράγεται από το λογισμικό εφαρμογής που τρέχει ο μικροεπεξεργαστής, εντούτοις οι είσοδοι των κυκλωμάτων οδήγησης δεν τροφοδοτούνται απευθείας από τις εξόδους του μικροεπεξεργαστή αλλά μεσολαβεί ο serial to parallel καταχωρητής. Αυτός ο σχεδιασμός γίνεται για να μην απασχολούνται 8 έξοδοι του μικροεπεξεργαστή ταυτόχρονα αλλά μόνο τέσσερις. Έτσι πετυχαίνουμε οικονομία αφού οι άλλες τέσσερις έξοδοι μπορεί να χρησιμοποιηθούν για άλλους σκοπούς.
Βέβαια εδώ το “λογισμικό” είναι ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου με ότι αυτό συνεπάγεται για τον ακριβή έλεγχο του συστήματος. Αν έχουμε υψηλές απαιτήσεις από την εφαρμογή μας σε ακρίβεια, ασφάλεια, αν ο έλεγχος αφορά σε εξοπλισμό που υποστηρίζει κρίσιμες λειτουργίες (π.χ ιατρικά μηχανήματα, αντλίες δοσομετρητών κλπ) τότε είναι σαφώς προτιμότερο να προγραμματίσουμε το σύστημα σε μια γλώσσα χαμηλότερου επιπέδου όπως η γλώσσα μηχανής του μικροεπεξεργαστή. Αυτό μας εξασφαλίζει αμεσότερο

έλεγχο και εποπτεία του συστήματος αλλά και μας παρέχει μεγαλύτερη ευελιξία σε πιθανές τροποποιήσεις.

4.2.1 Καθορισμός της ταχύτητας περιστροφής - Η συνάρτηση setSpeed(*float speed*)

Με τη συνάρτηση αυτή ορίζουμε την επιθυμητή σταθερή ταχύτητα [steps / sec] με την οποία θέλουμε να περιστραφεί ο κινητήρας.

```
void AccelStepper::setSpeed(float speed)
{
   _speed = speed;
   _stepInterval = abs(1000.0 / _speed);      //ms

}
```

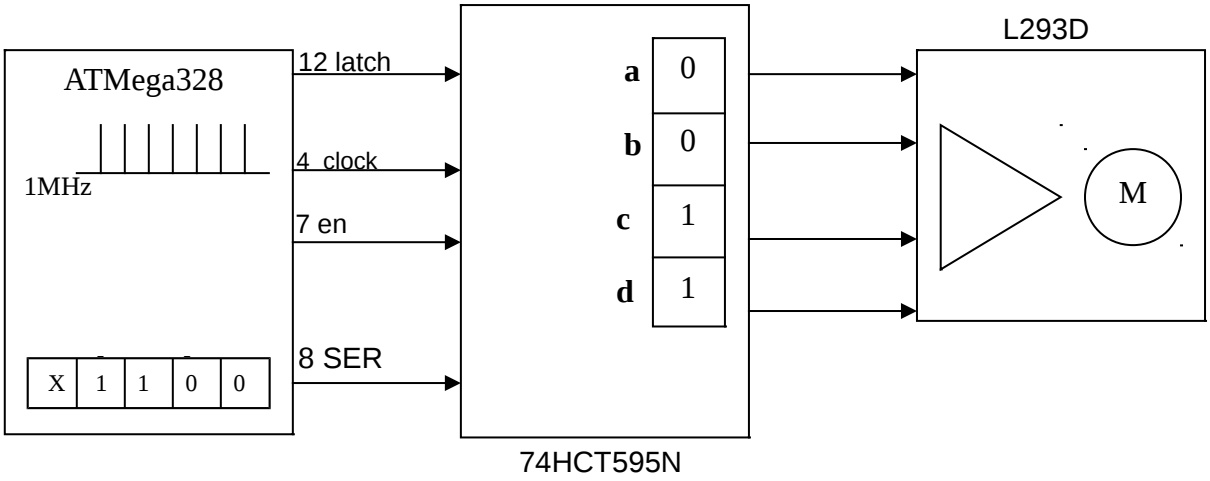
Εάν δεν ορίζεται αλλιώς στο πρόγραμμα θετική τιμή ορίσματος σημαίνει περιστροφή cw και αρνητική σημαίνει περιστροφή ccw.
Κάθε φορά που καλούμε την setSpeed(), θέτωντας το όρισμά της, γίνεται κλίση στην συνάρτηση speed()

```
float AccelStepper::speed()
{
   return _speed;
}
```

Η περιστροφή γίνεται στην επιθυμητή ταχύτητα μέσω του interstep delay που καθορίζεται από την setSpeed(). Η συχνότητα του master_clock είναι 1MHz και μέσω του interstep delay καθορίζεται η (σταθερή) χρονική καθυστέρηση μεταξύ δυο διαδοχικών παλμών.
Για παράδειγμα για ταχύτητα περιστροφής 400 steps/sec,για κάθε 2500 παλμούς του master clock παράγεται ένας παλμός που αντιστοιχεί σε βήμα. Αυτό σημαίνει ότι θα έχουμε interstep delay ίση με 2,5ms.

4.2.2 Καθορισμός του τρόπου διέγερσης των φάσεων

Όπως αναφέραμε προηγουμένως για την διέγερση των φάσεων χρησιμοποιούμε τον shift register 74HTC595N οι έξοδοι του οποίου είναι τα σήματα ελέγχου φάσεων προς το κύκλωμα οδήγησης. Ο τρόπος με τον οποίον γίνεται η διέγερση φαίνεται στο παρακάτω σχηματικό διάγραμμα. Θεωρούμε ότι έχουμε κινητήρα με full step two phase on διέγερση όπως για παράδειγμα συμβαίνει με έναν υβριδικό κινητήρα με δύο φάσεις.
Εδώ μέσω του προγράμματος για λειτουργία του κινητήρα με σταθερή ταχύτητα ορίζεται μια μεταβλητή η οποία προσωμοιώνει έναν “εικονικό” καταχωρητή 8 bit. Στη συγκεκριμένη περίπτωση μας ενδιαφέρουν τα τέσσερα από τα οκτώ bits (ξεκινώντας από το msb) τα οποία παίρνουν τιμές που αντιστοιχούν σε λογικό “1” ή λογικό “0” από τις μεταβλητές a, b, c και d. Παλμοί από ένα ρολόι σταθερής συχνότητας κατευθύνονται από μια συγκεκριμένη ψηφιακή έξοδο (εδώ από την έξοδο #4) του μικροεπεξεργαστή στην κατάλληλη είσοδο του shift register. Κάθε παλμός του ρολογιού έχει σαν αποτέλεσμα να μεταφέρεται στο εσωτερικό του shift register και από ένα bit του “εικονικού” καταχωρητή. Αφού ολοκληρωθεί η μεταφορά με ένα ακόμη παλμό που δίνεται στην ψηφιακή έξοδο 12, ο “εικονικός” καταχωρητής εμφανίζεται στην παράλληλη έξοδο.



Ο shift register διαθέτει 8 παράλληλες εξόδους από τις οποίες μας ενδιαφέρουν μόνο οι τέσσερις μέσω των οποίων εφαρμόζονται τα σήματα οδήγησης στη γέφυρα.
Η εναλλαγή στη διέγερση γίνεται σύμφωνα με τον παρακάτω πίνακα:

a	**b**	**c**	**d**	
1	1	0	0	→ A+B+
0	1	1	0	→ A-B+
0	0	1	1	→ A-B-
1	0	0	1	→ A+B-

Στο συγκεκριμένο στιγμιότυπο δίνεται εντολή για διέγερση της φάσης Α με θετική πολικότητα και για διέγερση της φάσης Β με αρνητική πολικότητα. Η γέφυρα λαμβάνοντας αυτή την είσοδο θα προκαλέσει την μετατόπιση του άξονα του κινητήρα για ένα βήμα.
Με την επόμενη εντολή οι τιμές των μεταβλητών θα είναι τέτοιες ώστε η διέγερση να είναι A+B+. Τότε η γέφυρα θα οδηγήσει τον κινητήρα στο επόμενο βήμα.
Το κυριότερο πλεονέκτημα αυτού του τρόπου διέγερσης είναι η ευκολία στον προγραμματισμό τόσο της κατεύθυνσης περιστροφής, όσο και του επιθυμητού τρόπου διέγερσης αφου και οι δυο αυτές παράμετροι καθορίζονται μέσω μιας και μόνο εντολής και συγκεκριμένα της εντολής

```
motor1.onestep(FORWARD, DOUBLE);
```

όπως φαίνεται και στο παραπάνω πρόγραμμα. Η εντολή αυτή είναι εντολή της βιβλιοθήκης AFMotor.h και μέσω της εντολής αυτής καλείται “κρυφός” κώδικας που διασυνδέει τον μικροεπεξεργαστή με τον shift register, δημιουργεί και αναθέτει τιμές στον “εικονικό” καταχωρητή και στέλνει τα κατάλληλα σήματα για να εφαρμοστεί η έξοδος του shift register στις εισόδους του κυκλώματος οδήγησης.
Μειονέκτημα της μεθόδου είναι η ανάγκη για σωστό προγραμματισμό του χρονικού διαστήματος που μεσολαβεί μεταξύ δυο διαδοχικών διέγερσεων (interstep delay).

4.2.3 Καθορισμός της θέσης στόχου

Ο καθορισμός της απόστασης που θα διανύσει ο άξονας του κινητήρα γίνεται με τη βοήθεια της συνάρτησης moveTo(*long absolute*). Η συνάρτηση αυτή είναι μέλος της βιβλιοθήκης AccelStepper και σαν όρισμα δέχεται τον αριθμό των βημάτων που θέλουμε να εκτελέσει ο

κινητήρας. Η θέση στόχος είναι η απόλυτη σε σχέση με την αρχική θέση 0. Αυτό σημαίνει ότι εάν σε ένα πρόγραμμα έχουμε μια κλίση ως εξής: moveTo(500) και στη συνέχεια μια άλλη κλίση moveTo(200) ο κινητήρας θα μετακινηθεί 300 βήματα προς τα πίσω και όχι άλλα 200 βήματα προς τα εμπρός. Αντίθετα εάν η κλίση γίνει ως εξής: moveTo(550) ο κινητήρας θα εκτελέσει 50 επιπλέον βήματα προς τα εμπρός (και όχι 550). Επειδή η συνάρτηση έχει την δυνατότητα να υπολογίζει και την ταχύτητα με την οποία θα εκτελεσθεί το επόμενο βήμα, σε εφαρμογές που θέλουμε μόνο κίνηση με σταθερή ταχύτητα θα πρέπει μετά από την moveTo() να καλούμε την setSpeed().

```
void AccelStepper::moveTo(long absolute)
{
   _targetPos = absolute;
   computeNewSpeed();
}
```

Για την συνάρτηση computeNewSpeed() η οποία χρησιμοποιείται από την moveTo() θα μιλήσουμε στην παράγραφο 4.3.1

4.2.4 Η συνάρτηση run()

Είδαμε μόλις πρίν ότι η θέση στόχος σε ένα πρόγραμμα καθορίζεται από την πιο πρόσφατη κλίση της συνάρτησης moveTo(). Η συνάρτηση run() είναι η συνάρτηση που με τη σειρά της αναλαμβάνει να κινήσει τον κινητήρα προς τη θέση αυτή. Κάθε κλίση της run() μετακινεί τον άξονα του κινητήρα για ένα βήμα το πολύ (δηλ. ένα ή κανένα). Άρα έχουμε τόσες κλίσεις της συνάρτησης όσα και τα βήματα μέχρι το στόχο. Η run() μπορεί να υλοποιήσει επιτάχυνση και επιδράδυνση αλλά στην συγκεκριμένη περίπτωση η κίνηση γίνεται με σταθερή ταχύτητα όπως αυτή έχει καθοριστεί από την πιο πρόσφατη κλίση της συνάρτησης setSpeed().
Ο ορισμός της συνάρτησης είναι ο εξής:

```
boolean AccelStepper::run()
{

  if (_targetPos == _currentPos)
    return false;

   if (runSpeed())
   computeNewSpeed();
    return true;
}
```

Παρατηρούμε ότι η συνάρτηση είναι τύπου boolean, επιστρέφει δηλαδή τιμή true ή false. Από την πρώτη συνθήκη ελέγχου (*if (_targetPos == _currentPos) then return false;*) και από το ότι η συνάρτηση καλείται από το πρόγραμμα κάθε φορά που υπάρχει βήμα προς εκτέλεση οδηγούμαστε στο συμπέρασμα ότι η run() εκτός από το να μετακινεί τον άξονα του κινητήρα ελέγχει εάν υπάρχει άλλο βήμα για εκτέλεση. Εάν δεν υπάρχει τερματίζει την λειτουργία της, επιστρέφει τιμή false και ο κινητήρας σταματά την περιστροφή του.

4.3 Επιτάχυνση / Επιβράδυνση

Έστω το παρακάτω πρόγραμμα:

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 2);

void forwardstep1( ) {
  motor1.onestep(FORWARD, DOUBLE);
}

void backwardstep1( ) {
  motor1.onestep(BACKWARD, DOUBLE);
}

AccelStepper stepper1(forwardstep1, backwardstep1);

void setup( )
{
   stepper1.setMaxSpeed(600);              // steps per second
   stepper1.setAcceleration(1500.0);       // steps per second per second
   stepper1.moveTo(2000);                  // steps

}
void loop( )
{

   stepper1.run( );
}
```

Το πρόγραμμα αυτό υλοποιεί τις παρακάτω διαδικασίες ελέγχου ανοικτού βρόχου:

- Καθορισμός της (μέγιστης) ταχύτητας περιστροφής του κινητήρα
- Εκκίνηση του κινητήρα με εφαρμογή κατάλληλης επιτάχυνσης
- Έλεγχος ότι ο κινητήρας έφτασε την μέγιστη ταχύτητα περιστροφής
- Έλεγχος ότι ο κινητήρας έφτασε στο σημείο όπου πρέπει να ξεκινήσει η επιβράδυνση ανεξάρτητα από το εάν η μέγιστη ταχύτητα περιστροφής επιτεύχθηκε η όχι.

Οι παραπάνω λειτουργίες μαζί με αυτές που υλοποιούνται από το πρόγραμμα της παραγράφου 4.2 καλύπτουν το σύνολο των λειτουργιών ελέγχου ενός συστήματος ελέγχου ανοικτού βρόχου.

4.3.1 Η συνάρτηση setAcceleration(float *acceleration*)

Μια από τις σημαντικότερες συναρτήσεις της βιβλιοθήκης AccelStepper είναι η συνάρτηση setAcceleration(float *acceleration*) μέσω της οποίας καθορίζουμε το ρυθμό επιτάχυνσης και επιβράδυνσης της κίνησης του άξονα του βηματικού κινητήρα. Αυτό γίνεται μέσω της

παραμέτρου τύπου float η οποία δίνει την επιθυμητή επιτάχυνση / επιβράδυνση με μονάδα μέτρησης [steps/sec/sec]. Η τιμή πρέπει να είναι > 0.
Ο ορισμός τηςσυνάρτησης είναι ως εξής:

```
void AccelStepper::setAcceleration(float acceleration)
{
   _acceleration = acceleration;
   computeNewSpeed();
}
```

Παρατηρούμε ότι και εδώ γίνεται κλίση της συνάρτησης computeNewSpeed();

```
void AccelStepper::computeNewSpeed()
{
   setSpeed(desiredSpeed());
}
```

Η συνάρτηση αυτή υπολογίζει μια νέα στιγμιαία ταχύτητα την οποία και θέτει σαν την νέα ταχύτητα περιστροφής του άξονα του κινητήρα και ο υπολογισμός αυτός γίνεται έπειτα από την επιτυχή ολοκλήρωση της περιστροφής του άξονα του κινητήρα κατά ένα βήμα.
Επίσης η νέα στιγμιαία ταχύτητα υπολογίζεται από την συνάρτηση μετά από αλλαγές, που μπορούν να γίνουν από τον χρήστη, στις παραμέτρους των συναρτήσεων setMaxSpeed(), moveTo() και setAcceleration().
Αυτό γίνεται με κλίση της συνάρτησης setSpeed(), της οποίας το όρισμα καθορίζεται από το αποτέλεσμα που επιστρέφει η συνάρτηση desiredSpeed().
Η συνάρτηση desiredSpeed() ορίζεται ως εξής:

```
float AccelStepper::desiredSpeed()
{
   long distanceTo = target_position – current_position;
//   Εστω ότι έχουμε καθορίσει την επιθυμητή επιτάχυνση μέσω της setAcceleration().
//   Υπολογίζουμε την μέγιστη ταχύτητα που μπορεί να πάρει ο άξονας χωρίς όμως, όταν
//   επιβραδύνει, να ξεπεράσει την προκαθορισμένη θέση στόχο.
   float requiredSpeed;
   if (distanceTo == 0)
         return 0.0; // Ο άξονας του κινητήρα βρίσκεται στην προκαθορισμένη θέση στόχο.
   else if (distanceTo > 0) //  για κίνηση clockwise
         requiredSpeed = sqrt(2.0 * distanceTo * _acceleration);
   else  // για κίνηση counterclockwise
         requiredSpeed = -sqrt(2.0 * -distanceTo * _acceleration);

   if (requiredSpeed > _speed)
// εάν η ταχύτητα που υπολογίσαμε πιο πάνω είναι μικρότερη από την τρέχουσα ταχύτητα τότε
   {
         // μπορούμε να επιταχύνουμε και άλλο (για κίνηση clockwise)
         if (_speed == 0)
            requiredSpeed = sqrt(2.0 * _acceleration);
         else
            requiredSpeed = _speed + abs(_acceleration / _speed);
         if (requiredSpeed > _maxSpeed) // εάν η ταχύτητα που υπολογίσαμε είναι > maxSpeed
            requiredSpeed = _maxSpeed; // τότε γίνεται ίση με αυτή
   }
   else if (requiredSpeed < _speed)
   {
         // μπορούμε να επιταχύνουμε και άλλο (για κίνηση counterclockwise)
```

```
        if (_speed == 0)
            requiredSpeed = -sqrt(2.0 * _acceleration);
        else
            requiredSpeed = _speed - abs(_acceleration / _speed);
        if (requiredSpeed < -_maxSpeed) // εάν η ταχύτητα που υπολογίσαμε είναι > maxSpeed
            requiredSpeed = -_maxSpeed; // τότε γίνεται ίση με αυτή
    }
//  Serial.println(requiredSpeed);
    return requiredSpeed;
}
```

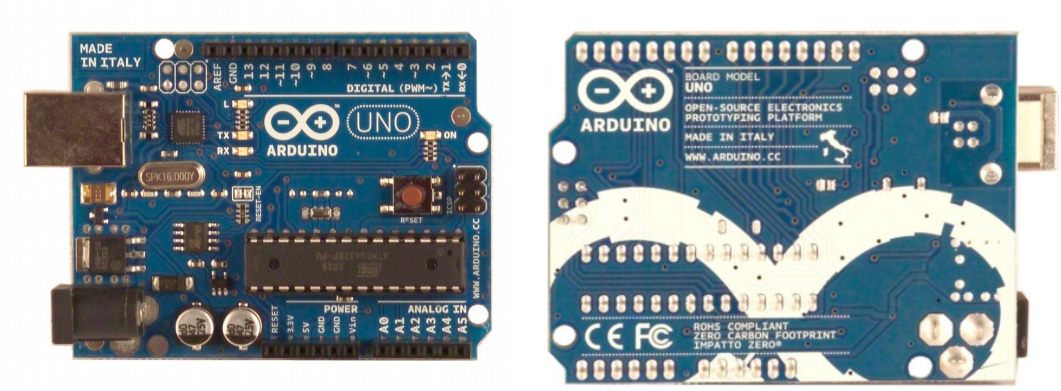
Αρα η συνάρτηση desiredSpeed() υπολογίζει κάθε φορά (μετά από κάθε βήμα) την μέγιστη ταχύτητα που μπορεί να πάρει ο άξονας του κινητήρα έτσι ώστε να η κίνηση να γίνει με την επιθυμητή επιτάχυνση υπο τους περιορισμούς να μην χαθεί ο συγχρονισμός με το μαγνητικό πεδίο, να μην ξεπεραστεί η θέση στόχος και να μην ξεπεράσει την μέγιστη ταχύτητα περιστροφής που έχουμε θέσει ως παράμετρο στο πρόγραμμα μέσω της συνάρτησης setMaxSpeed().
Η ταχύτητα αυτή (requiredSpeed) κατόπιν τίθεται ως όρισμα στην συνάρτηση setSpeed() η οποία επιπλέον επιστρέφει και το χρονικό διάστημα μέχρι τον επόμενο παλμό κατά την φάση της επιτάχυνσης (επιβράδυνσης) του άξονα του κινητήρα. Τα δεδομένα αυτά στη συνέχεια χρησιμοποιούνται από την computeNewSpeed() ώστε να υπολογιστεί η καινούργια στιγμιαία ταχύτητα του άξονα του κινητήρα.
Ετσι μπορούμε με μια απλή κλήση της setAcceleration(float acceleration) και αφού καθορίσουμε την τιμή της παραμέτρου να επιτύχουμε την επιταχυνόμενη κίνηση του άξονα του κινητήρα.

5. Φυσική Υλοποίηση του Συστήματος

5.1 Ο μικροελεγκτής Arduino Uno

5.1.1 Γενικά

Το Arduino είναι μια open source hardware – software πλατφόρμα ιδανική για τον σχεδιασμό και την ανάπτυξη interactive εφαρμογών. Ένα σύνολο από μικροελεγκτές, PCB κυκλώματα, περιφερειακά κλπ μπορούν εύκολα να συνδεθούν μεταξύ τους οδηγώντας έτσι στην ανάπτυξη μεγαλύτερης κλίμακας συστημάτων.
Το Arduino Uno (Σχήμα 5.1.1.1)είναι ένας μικροελεγκτής που βασίζεται στον επεξεργαστή ATMega328. Διαθέτει 14 Digital Input / Output ακροδέκτες (6 από τους οποίους χρησιμοποιούνται σαν PWM έξοδοι), 6 ακροδέκτες Analog Inputs που μπορούν επίσης να χρησιμοποιηθούν σαν Digital Inputs, ταλαντωτή 16MHz και USB θύρα μέσω της οποίας συνδέεται με το PC. Ο προγραμματισμός του μικροεπεξεργαστή γίνεται μέσω του ειδικού κατάλληλου περιβάλλοντος ανάπτυξης. Το λογισμικό αυτό είναι open – source και προσφέρει ένα περιβάλλον μέσω του οποίου μπορούμε να γράψουμε τον πηγαίο κώδικα, να τον φορτώσουμε στον μικροεπεξεργαστή και κατοπιν να τρέξουμε την εφαρμογή μας. Είναι συμβατό και με άλλες open source εφαρμογές και τρέχει σε Windows, Linux και MacOS. Η τρέχουσα έκδοση είναι το ARDUINO 1.6.1.



Σχήμα 5.1.1.1 – Το Arduino Uno. Εμπορόσθια και οπίσθια όψη

Στον παρακάτω πίνακα εμφανίζονται συνοπτικά τα βασικά τεχνικά χαρακτηριστικά.

Μικροεπεξεργαστής	ATMega328
Τάση Λειτουργίας	5V
Τάση εισόδου (εξωτερική)	7-12V
Τάση Εισόδου (οριακή)	6-20V
Digital I/O	14 (εκ των οποίων 6 PWM έξοδοι)
Analog Input	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

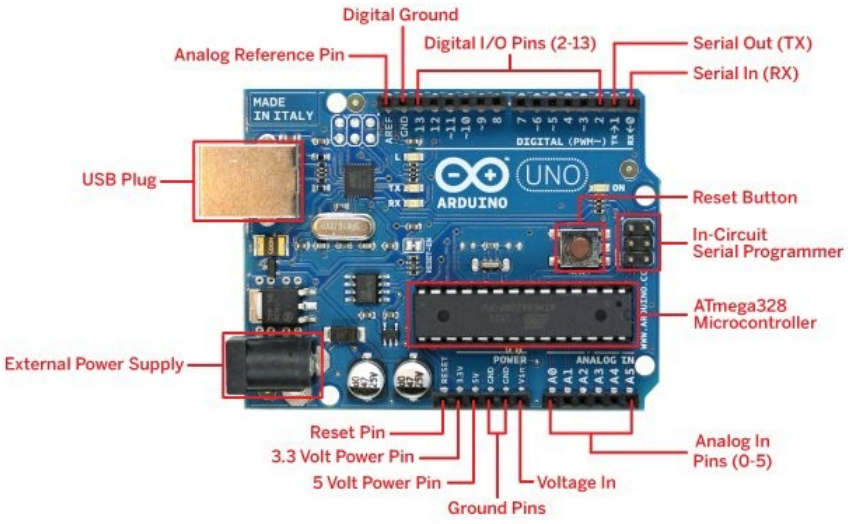
5.1.2 Τροφοδοσία

Η τροφοδοσία του μικροελεγκτή μπορεί να γίνει είτε μέσω της θύρας USB από όπου τροφοδοτείται με τάση 5V, είτε μέσω εξωτερικού τροφοδοτικού το οποίο μπορεί να συνδεθεί στην ειδική υποδοχή της πλακέτας. Η επιλογή της πηγής τροφοδοσίας γίνεται αυτόματα. Ακόμη μπορούμε να χρησιμοποιήσουμε και μπαταρία που συνδέεται στους ακροδέκτες Gnd και Vin της ειδικής υποδοχής POWER. Στην περίπτωση που χρησιμοποιούμε εξωτερικό τροφοδοτικό καλό είναι να περιορίζουμε την τάση στα 12V.

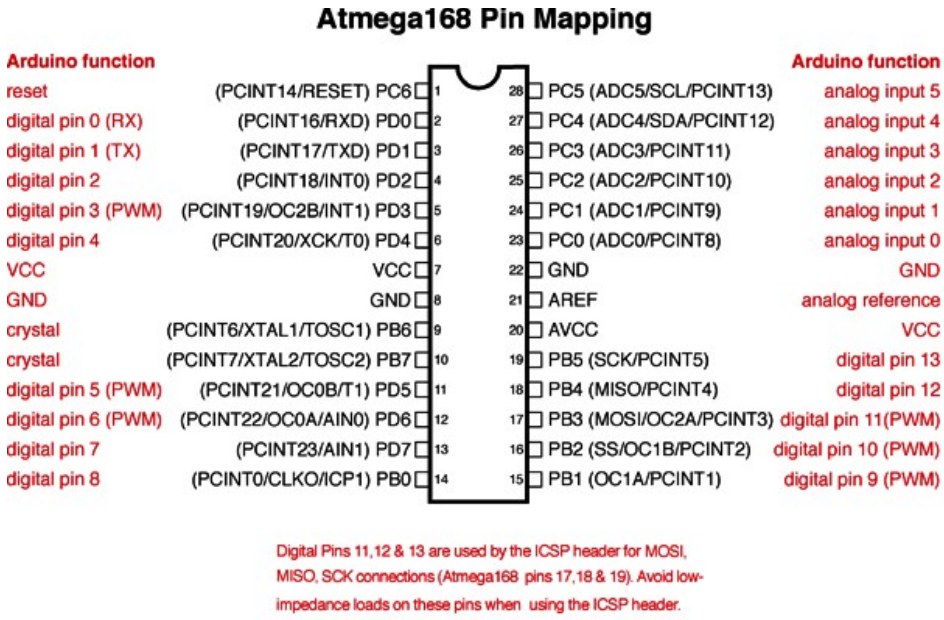
5.1.3 Είσοδοι / Έξοδοι

Οι ψηφιακοί ακροδέκτες παρέχουν λειτουργίες εισόδου / εξόδου (Σχήμα 5.1.3.1) και είναι (κυρίως για σκοπούς προγραμματισμού) οργανωμένες – ομαδοποιημένες σε δυο “θύρες” που λέγονται PORTS. Αυτές είναι η PORTD και η PORTB. Η PORTD αποτελείται από τους ακροδέκτες 0 έως 7 και η PORTB από τους ακροδέκτες 8 έως 13. Με κατάλληλες εντολές μπορούμε να αναθέσουμε τιμές και να ορίσουμε ποιος ακροδέκτης χρησιμοποιείται σαν είσοδος (με ανάθεση της τιμής λογικό 1) και ποιος σαν έξοδος, να διαβάσουμε και να γράψουμε στους ακροδέκτες ή να τους χρησιμοποιήσουμε σαν read only.
Από τα παραπάνω εξαιρούνται οι ακροδέκτες 0, 1 (TX / RX) και 12, 13που χρησιμοποιούνται αποκλειστικά για σειριακή επικοινωνία με άλλες συσκευές (πχ αισθητήρες).
Οι σημαντικότεροι ίσως ακροδέκτες του μικροελεγκτή είναι οι 6 από τους συνολικά 14 ψηφιακούς ακροδέκτες που επιπλέον προσφέρουν έξοδο Pulse Width Modulation (PWM). Αυτοί είναι οι ακροδέκτες 3, 5, 6, 9,10 και 11.

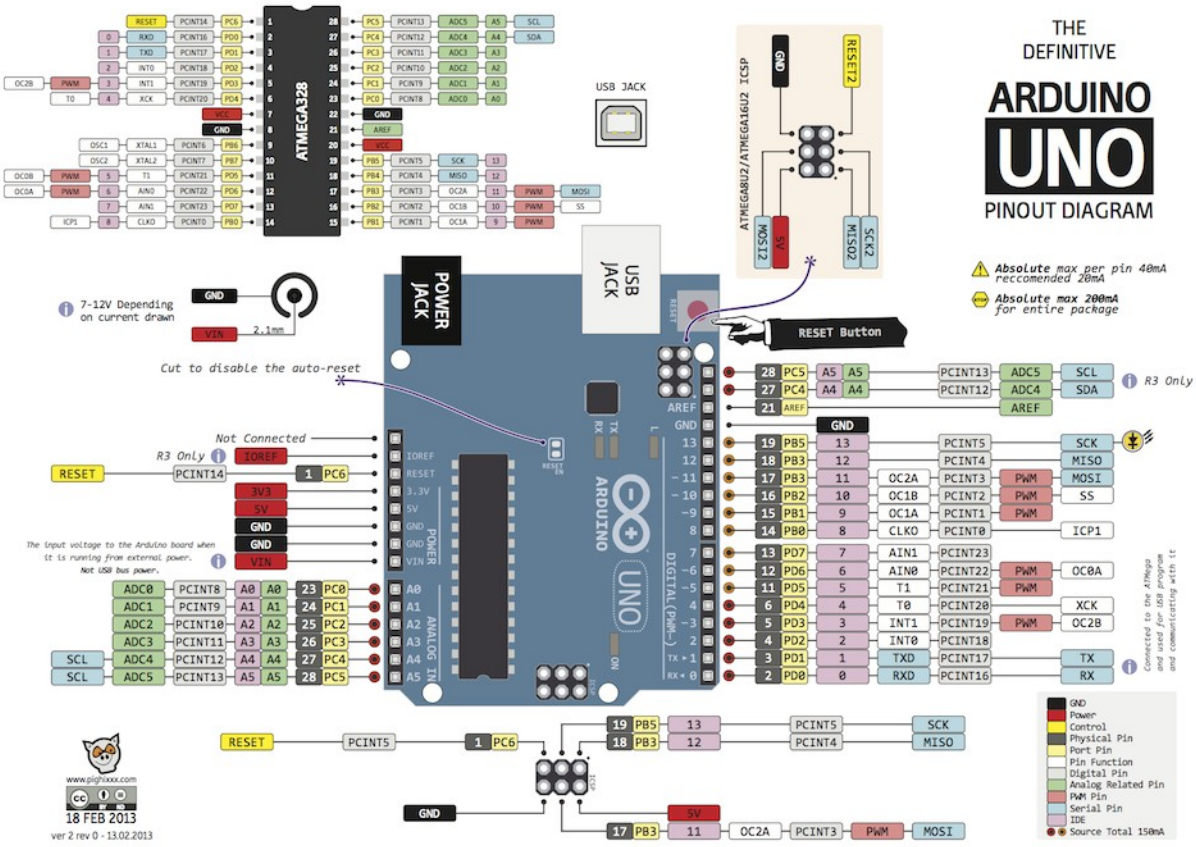
Όπως θα δούμε παρακάτω και σε συνδυασμό με το περιφερειακό motor shield μπορούμε να οδηγήσουμε μέχρι και δυο κινητήρες. Τότε από τους έξι αυτούς ακροδέκτες χρησιμοποιούνται οι τέσσερις και συγκεκριμένα οι 3, 5, 6 και 11. Από αυτούς οι 3 και 11 συνδέονται απευθείας με το κύκλωμα οδήγησης του ενός κινητήρα και οι 5 και 6 συνδέονται απευθείας με το κύκλωμα οδήγησης του δεύτερου κινητήρα.
Όπως είπαμε προηγουμένως “καρδιά” του Arduino Uno είναι ο επεξεργαστής Atmega328. Χρήσιμο είναι επομένως να γνωρίζουμε την αντιστοιχία στους ακροδέκτες επεξεργαστή και τελικής διαμόρφωσης του μικροελεγκτή. Αυτό φαίνεται στην παρακάτω εικόνα (Σχήμα 5.1.3.2). Το ολοκληρωμένο λεπτομερές διάγραμμα (pinout diagram)φαίνεται στο σχήμα 5.1.3.3



Σχήμα 5.1.3.1 – Είσοδοι και έξοδοι του Arduino Uno.



Σχήμα 5.1.3.2 – Αντισοιχία θυρών μικροεπεξεργαστή - μικροελεγκτή.



Σχήμα 5.13.3 – Λεπτομερές διάγραμμα ακροδεκτών Arduino Uno

5.1.4 Το Περιβάλλον Ανάπτυξης

Μέσω του ολοκληρωμένου περιβάλλοντος ανάπτυξης (Arduino IDE) γίνεται δυνατός ο προγραμματισμός του μικροελεκτή Arduino Uno. Το IDE (Σχήμα 5.4.1.1) μας παρέχει έναν text editor για να γράφουμε τον πηγαίο κώδικα, μια περιοχή μηνυμάτων προς το χρήστη όπου απεικονίζονται πληροφορίες όπως ότι ένα πρόγραμμα σώθηκε επιτυχώς ή ότι υπάρχουν λάθη κατά την μετάφραση κλπ, μια γραμμή εργαλείων μέσω της οποίας εκτελούμε τις σημαντικότερες λειτουργίες και ένα serial monitor μέσω του οποίου παρακολουθούμε την επικοινωνία του μικροελεκτή μέσω της θύρας USB. Συνήθως τα προγράμματα που γράφουμε στον text editor και που τρέχει ο μικροελεκτής τα αποκαλούμε sketches.
Μέσω της γραμμής εργαλείων μπορούμε να δημιουργήσουμε ένα νέο πρόγραμμα, να ανοίξουμε ένα υπάρχον πρόγραμμα, να σώσουμε και να φορτώσουμε στον μικροελεκτή ένα πρόγραμμα και να ανοίξουμε το monitor της σειριακής θύρας.
Παρακάτω απεικονίζονται τα σημαντικότερα εικονίδια της γραμμής εργαλείων και οι αντιστοιχες λειτουργίες.


Ελέγχει αν υπάρχουν λάθη στο πρόγραμμα

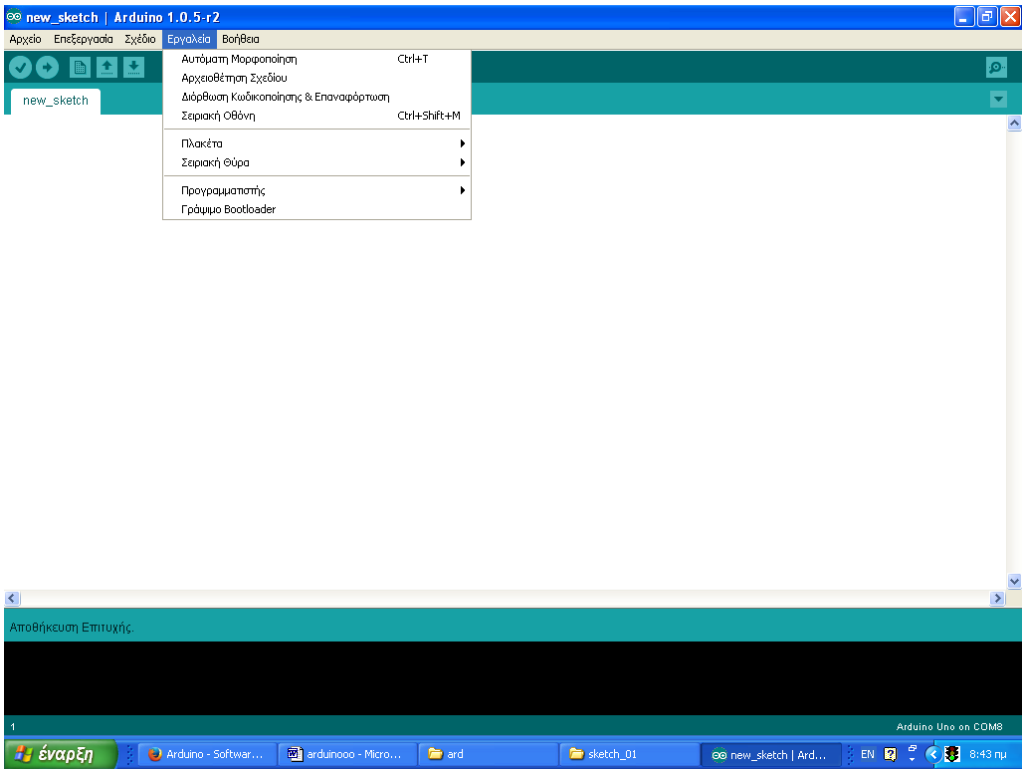

Κάνει την μεταγλώττιση και φορτώνει το πρόγραμμα στον μικροελεκτή


Δημιουργεί ένα νέο πρόγραμμα.


Παρουσιάζει όλα να υπάρχοντα προγράμματα και ανοίγει το επιλεχθέν από αυτά


Σώζει το τρέχον πρόγραμμα


Ανοίγει την σειριακή θύρα

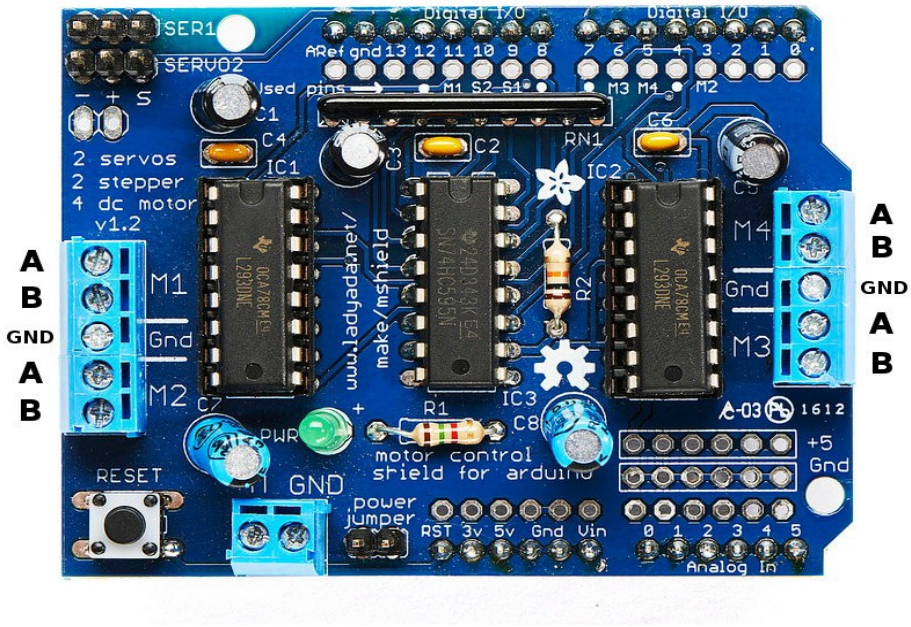


Σχήμα 5.4.1.1 – Το Arduino IDE

5.2 Η πλακέτα ελέγχου – οδήγησης Adafruit Servo / Stepper / DC Motor shield

5.2.1 Γενική Επισκόπηση

Το τυπωμένο αυτό κύκλωμα (Σχήμα 5.2.1.1) είναι συμβατό και συνεργάζεται ως περιφερειακή μονάδα με το Arduino Uno. Πρόκειται για μια πλακέτα PCB της εταιρίας Adafruit και έχει το χαρακτηριστικό ότι τοποθετείται ΠΑΝΩ από το Arduino Uno.

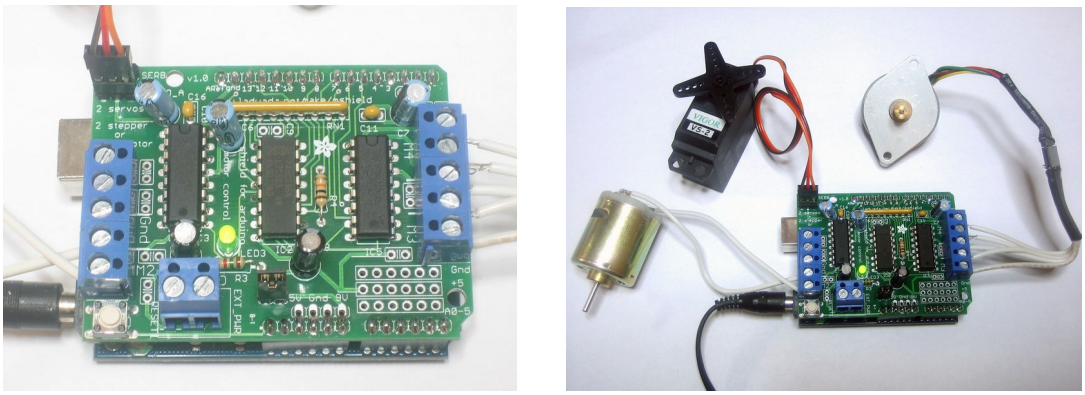


Σχήμα 5.2.1.1 – Το motor shield

Για το λόγο αυτό άλλωστε φέρει και τον χαρακτηρισμό “shield” (βλ. Σχήμα 5.5.1.2).
Με την πλακέτα αυτή και σε συνδυασμό με το Arduino Uno μπορούμε να ελέγξουμε βηματικούς κινητήρες αλλά και DC κινητήρες και σερβοκινητήρες. Σε ότι αφορά τους βηματικούς κινητήρες είναι δυνατή μονοπολική και διπολική οδήγηση με οποιονδήποτε από όλους τους δυνατούς τρόπους διέγερσης των φάσεων του κινητήρα. Έχουμε δηλαδή διέγερση full step (one phase on και two phase on), διέγερση half step και microstepping.
Το shield περιέχει δύο κυκλώματα οδήγησης L293D και έναν καταχωρητή σε ρόλο shift register το 74HCT595N. Ο καταχωρητής αυτός δέχεται εντολές από τον μικροελεγκτή και επικοινωνεί με τα κυκλώματα οδήγησης ώστε να ελέγχει την κατεύθυνση περιστροφής του κινητήρα.
Το motor shield τροφοδοτείται μέσω του Arduino από την ειδική υποδοχή για εξωτερικό τροφοδοτικό. Επίσης μπορεί να τροφοδοτηθεί και από τους ειδικούς ακροδέκτες PWR , GND που είναι πάνω στο shield. Όταν τροφοδοτείται μέσω του Arduino τότε πρέπει να έχουμε τοποθετήσει στη θέση του το power jumper. Εάν έχουμε ξεχωριστές τροφοδοσίες για τις δυο πλακέτες τότε πρέπει να αφαιρέσουμε το power jumper. Καλό θα είναι η τάση να περιορίζεται στα 12VDC.
Απαραίτητη προυπόθεση για τον προγραμματισμό και την επικοινωνία του motor shield με τον μικροελεκτή Arduino Uno και άρα για την λειτουργία του module συνολικά, είναι η εγκατάσταση της βιβλιοθήκης AF_Motor. Η βιβλιοθήκη αυτή περιέχει τις απαραίτητες συναρτήσεις για να προγραμματίσουμε τη λειτουργία του βηματικού κινητήρα. Οι σημαντικότερες από τις συναρτήσεις που περιέχονται στο μενού της βιβλιοθήκης είναι οι παρακάτω.

- **AF_Stepper(*steps*, *stepper#*)**

Η συνάρτηση αυτή δημιουργεί ένα αντικείμενο βηματικού κινητήρα και ενεργοποιεί τις γέφυρες και τον 74HCT595N του motor shield. Η μεταβλητή ***steps*** δηλώνει τον αριθμό βημάτων ανα πλήρη περιστροφή (χαρακτηριστικό του κινητήρα) ενώ η μεταβλητή ***stepper#*** παίρνει την τιμή 1 ή 2 και δηλώνει σε ποιούς ακροδέκτες του motor shield έχουμε συνδέσει τον κινητήρα (βλ. παράγραφο 5.2.3).



Σχήμα 5.2.1.2- Η πλακέτα ελέγχου σε συνδυασμό με το Arduino Uno και συνδεσμολογία DC motor , servo motor , stepper motor

- **setSpeed(*rpm*)**

Με τη συνάρτηση αυτή καθορίζπυμε την επιθυμητή ταχύτητα περιστροφής του κινητήρα σε στροφές ανα λεπτό.

- **step(*#steps, direction, steptype*)**

Με τη συνάρτηση αυτή ουσιαστικά θέτουμε σε κίνηση τον κινητήρα. Την καλούμε όποτε θέλουμε ο κινητήρας να περιστρέψει τον άξονά του σύμφωνα με τις τιμές που έχουμε δώσει στις παραμέτρους της συνάρτησης. Η τιμή της παραμέτρου ***#steps*** καθορίζει τον αριθμό των βημάτων που θα περιστραφεί ο άξονας του κινητήρα. Η τιμή της παραμέτρου ***direction*** μπορεί να είναι FORWARD ή BACKWARD και καθορίζει την κατεύθυνση περιστριφής και η τιμή της παραμέτρου ***steptype*** καθορίζει αν θα έχουμε διέγερση one phase on - full step, διέγερση two phase on – full step, διέγερση half step ή microstepping (βλ. παράγραφο 1.3). Η τιμές που μπορεί να πάρει η παράμετρος είναι SINGLE, DOUBLE, INTERLEAVE και MICROSTEP αντίστοιχα.

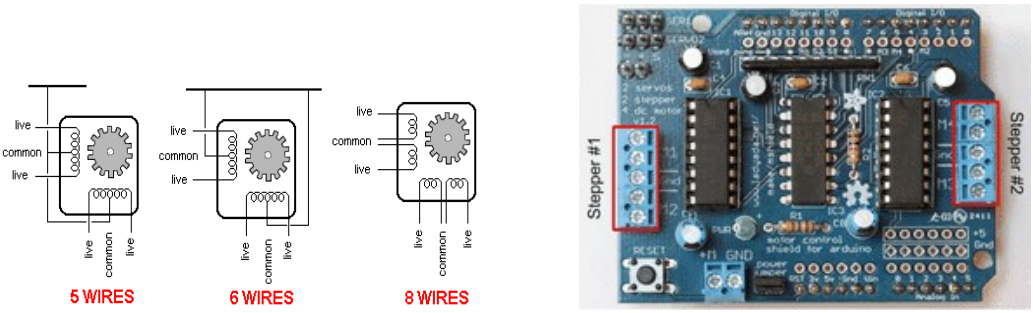
5.2.2 Σύνδεση και επικοινωνία με το Arduino Uno

Όπως αναφέραμε προηγουμένως το motor shield τοποθετείται πάνω από το Arduino Uno. Η φυσική σύνδεση των δυο πλακετών επιτυγχάνεται με τη συγκόλληση των ειδικών συνδετήρων (headers) ενώνοντας έτσι τις δυο πλακέτες. Υπάρχουν 36 τέτοιοι συνδετήρες.
Η ηλεκτρική – ηλεκτρονική επικοινωνία μεταξύ των δυο πλακετών επιτυγχάνεται πάλι μέσω των headers οι οποίοι τοποθετούνται πάνω στο board του Arduino Uno με τέτοιον τρόπο ώστε να “μεταφέρουν” στο motor shield τα σήματα εκείνα που παράγονται από τον μικροελεγκτή και που είναι απαραίτητα στο motor shield για την λειτουργία των κινητήρων. Έτσι και κατά τη

φάση της συγκόλλησης ομοδοποιούμε τους 36 συνδετήρες ως εξής: Δυο ομάδες των 8 συνδετήρων, δυο ομάδες των 6 συνδετήρων, 2 ομάδες των 3 και τέλος μια ομάδα των 2 συνδετήρων. Από αυτές η ομάδα των 2 συνδετήρων χρησιμοποιείται για την τροφοδοσία του module των δυο πλακετών και συγκεκριμένα αποτελεί τους ακροδέκτες στους οποίους συνδέεται το power jumper όταν το module τροφοδοτείται με εξωτερικό τροφοδοτικό το οποίο συνδέεται στην πλακέτα του Arduino Uno. Η ομάδα αυτή δηλαδή δεν μεταφέρει κάποιο σήμα από τον μικροελεγκτή και τοποθετείται απευθείας πάνω στο motor shield. Το ίδιο συμβαίνει και με τις δυο ομάδες των 3 συνδετήρων οι οποίες χρησιμεύουν αποκλειστικά για τη συνδεσμολογία σερβοκινητήρων και δεν παίζουν κάποιο ρόλο κατά την χρήση του motor shield για την οδήγηση βηματικών κινητήρων. Οι δυο ομάδες των 8 συνδετήρων “μεταφέρουν” τους 14 ψηφιακους Digital Input / Output ακροδέκτες από το board του Arduino Uno στο board του motor shield. Από τους ακροδέκτες αυτούς, οι ακροδέκτες 4,7,8 και 12 του Arduino board χρησιμεύουν για την οδήγηση των βηματικών κινητήρων (που έχουμε συνδέσει στο motor shield) μέσω του serial to parallel καταχωρητή 74HCT595. Στο σημείο αυτό πρέπει να τονίσουμε το γεγονός ότι οι βηματικοί κινητήρες επι της ουσίας δεν συνδέονται απευθείας σε κάποιον από τους ακροδέκτες του Arduino Uno αλλά στην πραγματικότητα συνδέονται στο 74HCT595 το οποίο δέχεται εντολές (σήματα) από το board του Arduino Uno. Επίσης μεταφέρονται από το Arduino Uno στο motor shield τα 4 από τα 6 συνολικά PWM σήματα τα οποία και τροφοδοτούν τις δυο γέφυρες στο board του motor shield. Τα σήματα αυτά ξεκινούν από τους ακροδέκτες 3, 5, 6 και 11 του μικροελεκτή (board) Arduino Uno. Από τις ομάδες των 6 συνδετήρων η μια μεταφέρει τους 6 αναλογικούς ακροδέκτες και η άλλη μεταφέρει ακροδέκτες που μπορούν να χρησιμεύσουν στην τροφοδοσία του module. Οι αναλογικοί ακροδέκτες μπορούν να χρησιμοποιηθούν κατά τη λειτουργία του motor shield ώστε να συνδέσουμε για παράδειγμα κάποιον αναλογικό αισθητήρα που θα διακόπτει π.χ τη λειτουργία του module. Επίσης με κατάλληλο προγραμματισμό μπορούν να χρησιμοποιηθούν και ως επιπλέον ψηφιακές είσοδοι / έξοδοι.
Με τον τρόπο αυτό επιτυγχάνουμε την φυσική και την ηλεκτρική / ηλεκτρονική σύνδεση των δυο ξεχωριστών πλακετών σε ένα ενιαίο πλέον module.

5.2.3 Σύνδεση βηματικών κινητήρων

Η τοποθέτηση των βηματικών κινητήρων στο motor shield γίνεται μέσω των ειδικών υποδοχών. Υπάρχουν 2 τέτοιες υποδοχές με πέντε (5) ακροδέκτες ώστε να μπορούν να συνδεθούν όλοι οι τύποι κινητήρων. Στην περίπτωση κινητήρων με 5, 6 ή 8 καλώδια αυτά που αποτελούν τις μεσαίες λήψεις πρέπει να συνδεθούν στον μεσαίο ακροδέκτη Gnd και κατόπιν τα τυλίγματα συνδέονται στα “ζεύγη” ακροδεκτών M1, M2 ή M3, M4 αντίστοιχα . Στην περίπτωση κινητήρων με 4 καλώδια δεν χρησιμοποιούμε τον ακροδέκτη Gnd τα τυλίγματα συνδέονται κανονικά στα ζεύγη M1, M2 ο ένας κινητήρας και M3, M4 ο άλλος. Έτσι στην περίπτωση υβριδικού διπολικού βηματικού κινητήρα συνδέουμε την μια φάση στους ακροδέκτες M1 και την άλλη φάση στους ακροδέκτες M2. Για να συνδέσουμε τον δεύτερο κινητήρα χρησιμοποιούμε τους ακροδέκτες M3 και M4 αντίστοιχα (βλ. σχήμα 5.2.3.1).

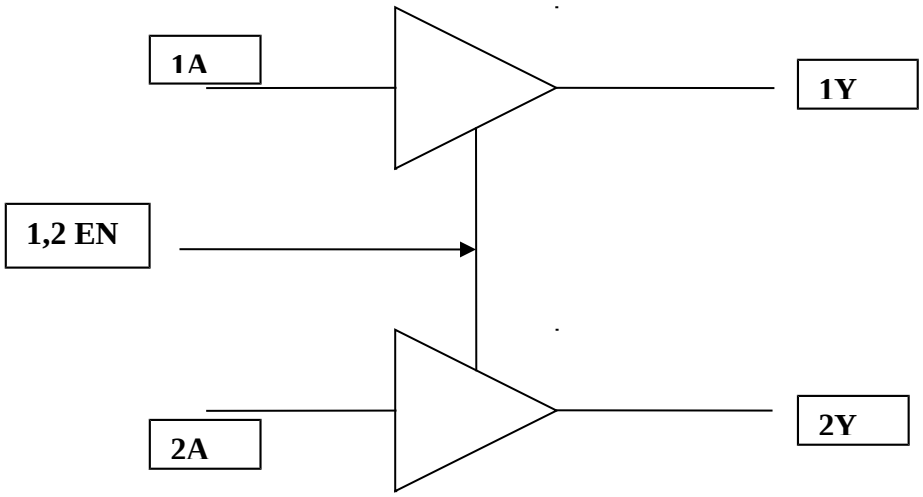


Σχήμα 5.2.3. 1 – Σύνδεση μέχρι 2 βηματικών κινητήρων στο motor shield

Οι ακροδέκτες αυτοί τροφοδοτούνται απευθείας με τα ρεύματα φάσεων από τις εξόδους του αντίστοιχου κυκλώματος οδήγησης. Το motor shield χρησιμοποιεί δυο ολοκληρωμένα κυκλώματα τύπου L293D με το καθένα από αυτά να ελέγχει τον αντίστοιχο κινητήρα.

5.3 Οδήγηση βηματικού κινητήρα

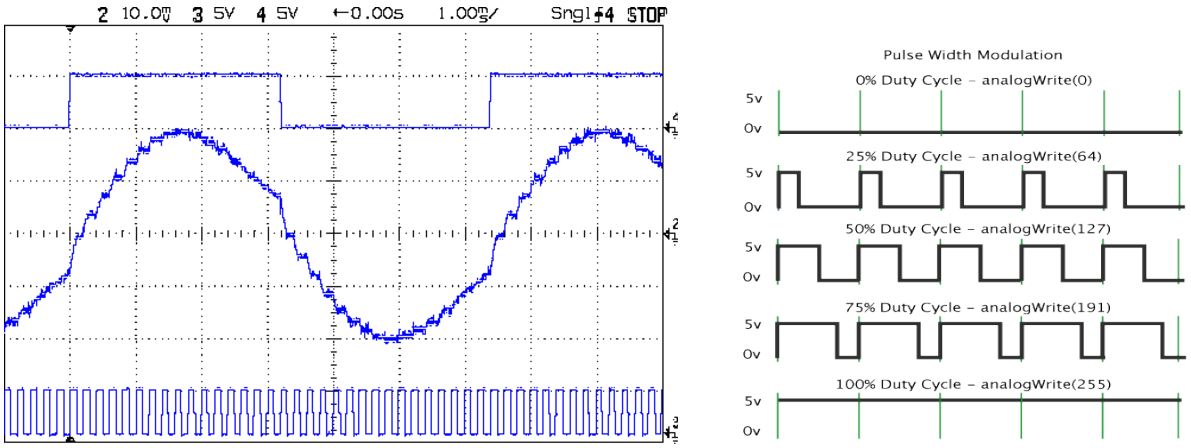
Το motor shield διαθέτει δύο κυκλώματα οδήγησης και έτσι μπορεί να οδηγήσει μέχρι δύο βηματικούς κινητήρες. Το κύκλωμα οδήγησης είναι το L293D που είναι τύπου γέφυρας (full H bridge driver), άρα είναι κατάλληλο και για διπολική οδήγηση. Η κάθε γέφυρα διαθέτει τέσσερις ψηφιακές εισόδους (στους ακροδέκτες 2,7,10 και 15 του ολοκληρωμένου κυκλώματος) και τέσσερις εξόδους (στους ακροδέκτες 3, 6, 11 και 14 του L293D).
Στις εξόδους συνδέονται οι φάσεις του κινητήρα και συγκεκριμένα η μια φάση στις εξόδους 3 και 6 και η άλλη φάση στις 11 και 14.
Τις εισόδους μπορούμε να τις ονομάσουμε και 1A , 2A ,3A και 4A και τις εξόδους 1Y, 2Y, 3Y και 4Y και αυτά ισχύουν για κάθε γέφυρα.
Επίσης στον ακροδέκτη 1 του chip εφαρμόζεται το σήμα 1,2EN που αφορά στις 1 A και 2 A και στον ακροδέκτη 9 εφαρμόζεται το σήμα 3,4 EN που αφορά στις 3 A και 4 A.
Τα σήματα 1,2EN και 3,4 EN παράγονται απευθείας από τον μικροεπεξεργαστή και είναι σήματα PWM. Το ίδιο συμβαίνει και για τα υπόλοιπα 2 ζεύγη εισόδων στην δεύτερη γέφυρα. Μέσω των σημάτων αυτών επιτυγχάνεται η λειτουργία microstepping.Το πώς επιτυγχάνεται ο έλεγχος φαίνεται στο παρακάτω λογικό διάγραμμα και στον αντίστοιχο πίνακα αληθείας.



ΕΙΣΟΔΟΙ		*ΕΞΟΔΟΣ*
A	*EN*	*Y*
H	**H**	**H**
L	**H**	**L**
X	**L**	**Z**

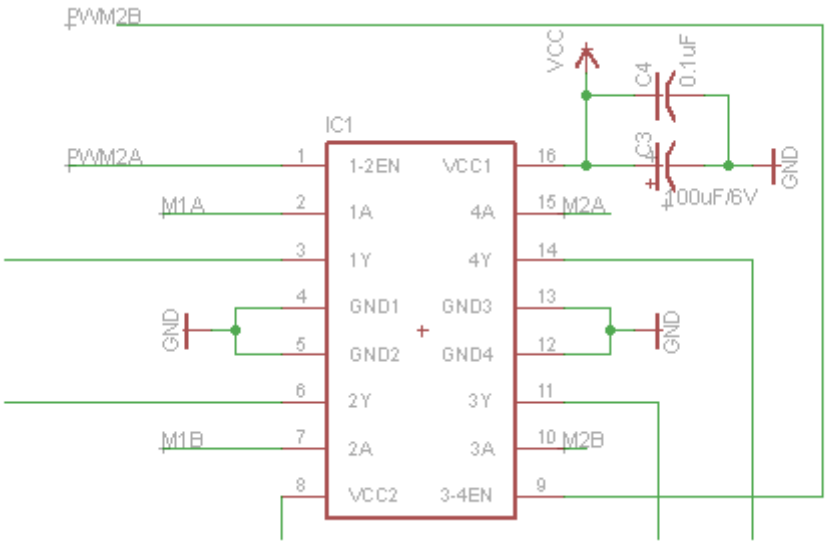
Όταν το σήμα στις 1A και 2 A είναι λογικό HIGH και ταυτόχρονα το σήμα 1,2 EN είναι και αυτό λογικό HIGH η έξοδος Y, (δηλαδή οι ακροδέκτες 1Y και 2 Y) είναι ενεργοποιημένη που σημαίνει ότι τότε διέρχεται ηλεκτρικό ρεύμα από τα τυλίγματα του κινητήρα. Όταν το σήμα στις εισόδους 1 A και 2 A είναι λογικό LOW τότε η έξοδος απενεργοποιείται που σημαίνει ότι σταματάει η ροή του ρεύματος στον κινητήρα. Όταν η είσοδος 1,2 EN είναι LOW η έξοδος είναι απενεργοποιημένη (κατάσταση υψηλής εμπέδησης) ανεξάρτητα από την κατάσταση στις 1 A και 2 A. Το σήμα PWM είναι ένα σήμα τετραγωνικού παλμού υψηλής συχνότητας και ουσιαστικά είναι το σήμα που διαμορφώνει το ρεύμα που διαρρέει τις φάσεις σε μορφή step wave έτσι ώστε το κάθε βήμα να υποδιαιρείται σε μικρότερα. Αυτό γίνεται μέσω του χρονικού διαστήματος που παραμένει σε κατάσταση HIGH (duty cycle). Στο παρακάτω σχήμα 5.3.1 βλέπουμε την δημιουργία του step wave σύμφωνα και με τον παραπάνω πίνακα αλήθειας. Η πάνω κυματομορφή αντιστοιχεί στο σήμα μιας εκ των 1 A, 2 A (τα σήματα μπορεί να είναι συμφασικά ή με διαφορά φάσης 180^{o}) και η κάτω αντιστοιχεί σε PWM σήμα με duty cycle 50% Η μεσαία κυματομορφή είναι το step wave.
Στην περίπτωση του βηματικού κινητήρα που χρησιμοποιούμε το κάθε βήμα υποδιαιρείται σε 16 microsteps και το PWM σήμα είναι συχνότητας 64KHz. Αυτά ορίζονται και στη βιβλιοθήκη AFMotor.h με τις παρακάτω εντολές.

```
#define MICROSTEPS 16
#define STEPPER1_PWM_RATE   MOTOR12_64KHZ      // PWM rate for stepper 1
#define STEPPER2_PWM_RATE   MOTOR34_64KHZ      // PWM rate for stepper 2
```



Σχήμα 5. 3. 1 – Pule Width Modulation (PMW)

Η συνδεσμολογία όλων των σημάτων φαίνεται στο παρακάτω διάγραμμα 5.3.2 που αφορά στην μια από τις δύο γέφυρές άρα σε έναν από τους δυο κινητήρες.



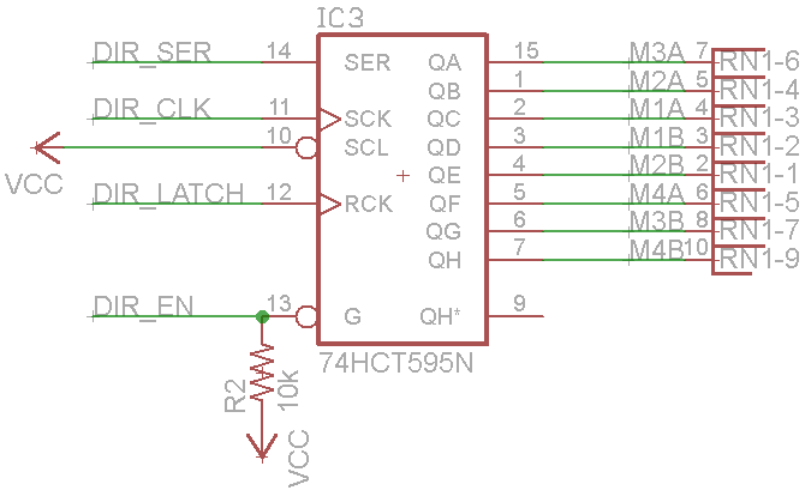
Σχήμα 5. 3. 2 – Συνδεσμολογία ολοκληρωμένου κυκλώματος L293D

Στην περίπτωση που θέλουμε ο κινητήρας να λειτουργεί χωρίς microstepping τότε ο μικροεπεξεργαστής παράγει PWM σήμα με duty cicle 100%.

Είπαμε προηγουμένως ότι το motor shield διαθέτει και έναν shift register τον 74HCT595N. Ο καταχωρητής αυτός έχει σχεδίαση serial to parallel, είναι 8 bits και κάθε έξοδός του είναι συνδεδεμένη με μια από τις 4 εισόδους της κάθε γέφυρας. Λειτουργεί δεχόμενος μια παλμοσειρά στην είσοδό του και την μετατρέπει σε παράλληλη έξοδο. Έτσι οι είσοδοι 1 A, 2 A, 3 A και 4 A της γέφυρας είναι συνδεδεμένες στις τέσσερις εξόδους του καταχωρητή και το ίδιο συμβαίνει και με την άλλη γέφυρα. Στην βιβλιοθήκη AFMotor.h υπάρχουν οι παρακάτω εντολές – ορισμοί: Εδώ οι εντολές MOTOR1_A, MOTOR1_B, MOTOR2_A και MOTOR2_B αφορούν ENA βηματικό κινητήρα και συγκεκριμένα αυτόν που συνδέεται στους ακροδέκτες M1 και M2 του motor shield, ενώ οι εντολές MOTOR4_A, MOTOR4_B, MOTOR3_A και MOTOR3_B αφορούν στον δεύτερο βηματικό κινητήρα που συνδέεται στους ακροδέκτες M3 και M4 του motor shield. Οι αριθμοί στην στήλη αντιστοιχούν στον αριθμό του bit του καταχωρητή με το MSB να είναι το 0. Άρα με βάση την αρίθμηση των bits βλέπουμε ότι ο ένας κινητήρας (έστω αυτός που συνδέεται στους ακροδέκτες M1 και M2 του motor shield) ελέγχεται από τα bits 2, 3, 1, 4 ενώ ο άλλος κινητήρας (αυτός που συνδέεται στους ακροδέκτες M3 και M4 toumotor shield) από τα bits 0, 6, 5 και 7.

```
// Bit positions in the 74HCT595 shift register output

#define MOTOR1_A  2
#define MOTOR1_B  3
#define MOTOR2_A  1
#define MOTOR2_B  4
#define MOTOR4_A  0
#define MOTOR4_B  6
#define MOTOR3_A  5
#define MOTOR3_B  7
```

Στο παρακάτω σχηματικό διάγραμμα 5.3.3 βλέπουμε τις εξόδους και τις εισόδους του καταχωρητή. Εδώ να σημειώσουμε ότι οι αριθμοί αμέσως μετά τον καταχωρητή αφορούν σε αρίθμηση των ακροδεκτών και είναι διαφορετικοί από τους αριθμούς των bits που αναφέραμε πιο πάνω.



Σχήμα 5. 3. 3 – Συνδεσμολογία ολοκληρωμένου κυκλώματος 74HCT595

Στα αριστερά βλέπουμε τους ακροδέκτες για την κύρια είσοδο, τις εισόδους ελέγχου και για την τροφοδοσία. Σε αυτές τις εισόδους εφαρμόζονται απευθείας οι έξοδοι του μικροεπεξεργαστή.

- DIR_SER: Αυτή είναι η βασική είσοδος των δεδομένων που προωθούνται στις εξόδους
- DIR_EN: Η είσοδος του ενεργοποιεί τον καταχωρητή. Είναι low enable.
- DIR_CLK: Είναι high enable και σε κάθε ενεργοποίηση της το bit που βρίσκεται στην είσοδο οδηγείται στο εσωτερικό του καταχωρητή.
- DIR_LATCH: Είναι high enable και προκαλεί την μετατόπιση ολόκληρου του byte στις εξόδους.

Αναφερθήκαμε προηγουμένως στην “ομαδοποίηση” των ακροδεκτών του μικροεπεξεργαστή σε θύρες (PORTB και PORTD). Αυτές διαχειρίζονται από μακροεντολές της γλώσσας C ως εξής:

- DDRB/DDRD: Καθορίζουν το ποιος από τους ακροδέκτες της θύρας θα λειτουργήσει ως είσοδος και ποιος ως έξοδος. Ανάθεση της τιμής λογικό 1 σημαίνει έξοδος.
- PORTB/PORTD: Διαβάζουν ολόκληρη τη θύρα ή/και γράφουν σε ολόκληρη τη θύρα. Έχουμε δηλαδή λειτουργία read / write.
- PINB/PIND: Λειτουργία read only.

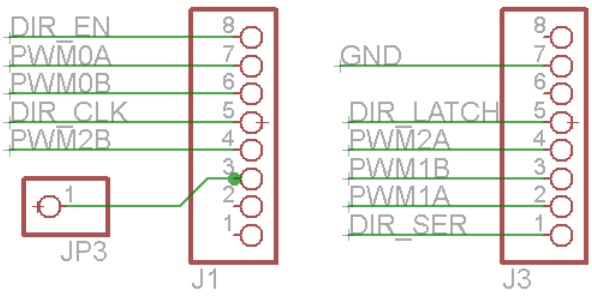
Με τις παρακάτω εντολές γίνεται το mapping ανάμεσα στις εισόδους του καταχωρητή, στις θύρες που ανήκουν, στο bit της θύρας που αντιστοιχεί στην κάθε είσοδο και στον αριθμό του ακροδέκτη που αντιστοιχεί στην πλακέτα του Arduino Uno (και μέσω αυτής στον μικροεπεξεργαστή)

```
#define LATCH 4
#define LATCH_DDR DDRB
#define LATCH_PORT PORTB

#define CLK_PORT PORTD
#define CLK_DDR DDRD
#define CLK 4

#define ENABLE_PORT PORTD
#define ENABLE_DDR DDRD
#define ENABLE 7

#define SER 0
#define SER_DDR DDRB
#define SER_PORT PORTB
```

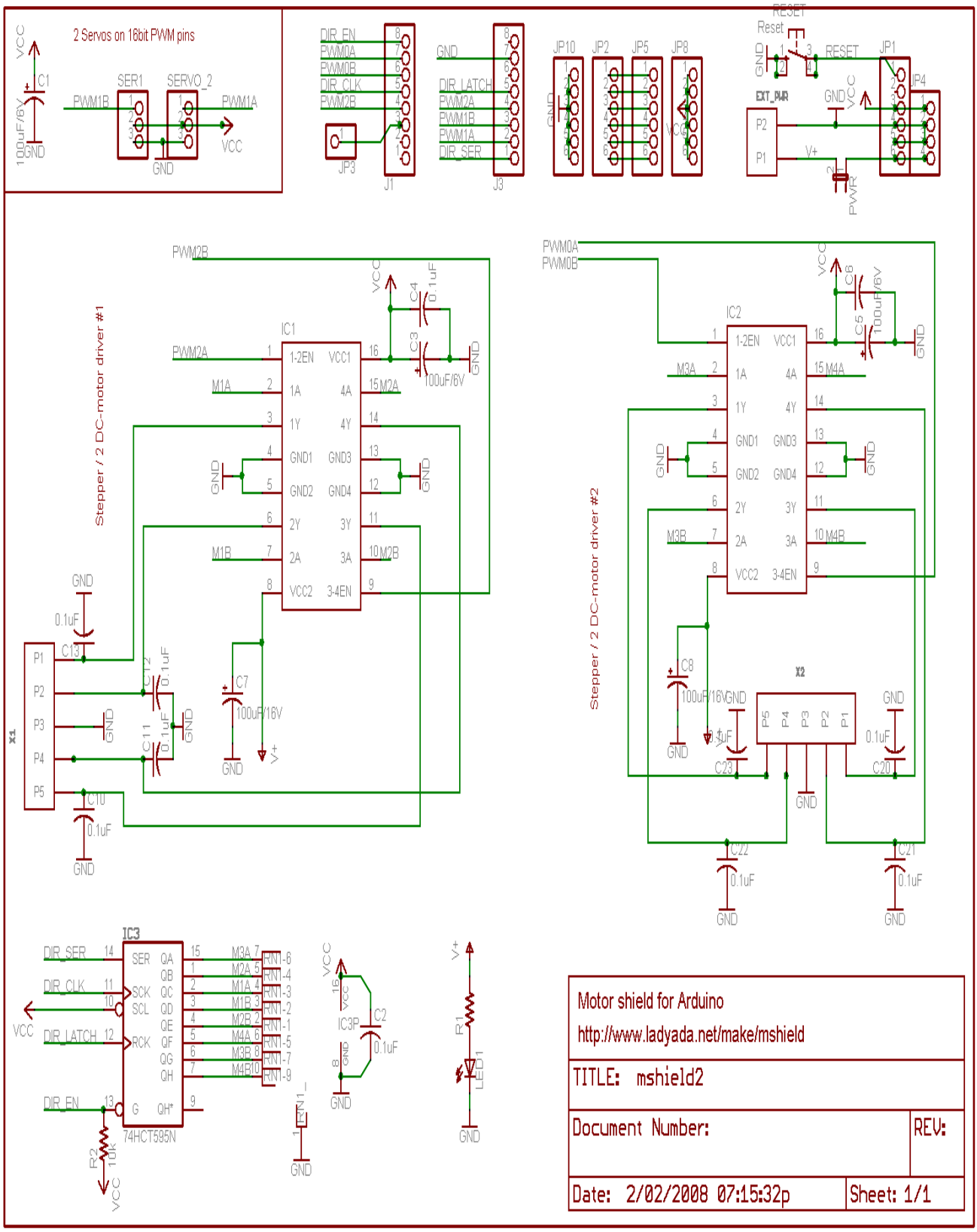
Το mapping αυτό χρησιμοποιείται από τις μακροεντολές της C για τον προγραμματισμό των θυρών.

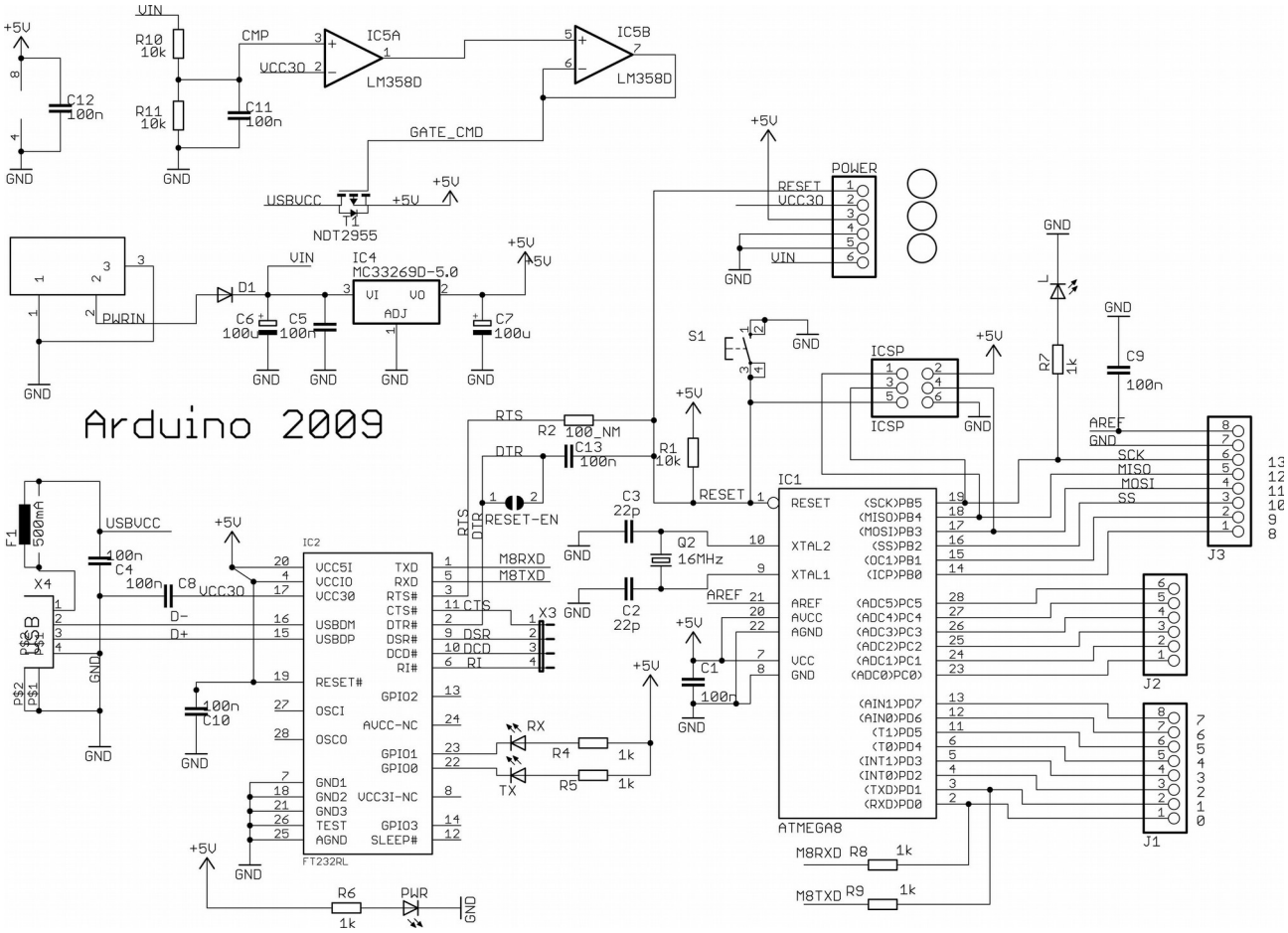
Είσοδος καταχωρητή	*θύρα*	*# bit της θύρας*	*Ψηφιακή έξοδος Arduino Uno*
DIR_LATCH	B	4	12
DIR_CLK	D	4	4
DIR_EN	D	7	7
DIR_SER	B	0	8

Στο παρακάτω σχήμα το module J1 αντιστοιχεί στους ακροδέκτες 0 – 7 (PORTD) και το module J3 στους ακροδέκτες 8 - 14 (PORTB)



Τα αναλυτικά διαγράμματα που παρουσιάζουν τη συνδεσμολογία μεταξύ Arduino Uno και motor shield είναι τα ακόλουθα.


2 Servos on 16bit PWM pins
SER1
SERVO_2
PWM1B
PWM1A
VCC
GND
C1
100uF/6V
DIR_EN
PWM0A
PWM0B
DIR_CLK
PWM2B
JP3
J1
DIR_LATCH
PWM2A
PWM1B
PWM1A
DIR_SER
J3
JP10
JP2
JP5
JP8
RESET
Reset
JP1
JP4
EXT_PWR
P2
P1
V+
PWR
Stepper / 2 DC-motor driver #1
IC1
1-2EN
VCC1
1A
4A
1Y
4Y
GND1
GND3
GND2
GND4
2Y
3Y
2A
3A
VCC2
3-4EN
PWM2A
M1A
M1B
M2A
M2B
C4
0.1uF
C3
100uF/6V
C13
C12
C11
C7
100uF/16V
C10
X1
P1
P2
P3
P4
P5
Stepper / 2 DC-motor driver #2
IC2
PWM0A
PWM0B
M3A
M4A
M3B
M4B
C6
C5
100uF/6V
C8
100uF/16V
X2
C23
C20
C22
C21
IC3
74HCT595N
DIR_SER
DIR_CLK
DIR_LATCH
DIR_EN
SER
QA
QB
QC
QD
QE
QF
QG
QH
SCK
SCL
RCK
G
QH*
RN1-6
RN1-4
RN1-3
RN1-2
RN1-1
RN1-5
RN1-7
RN1-8
RN1_
R2
10k
IC3P
C2
R1
LED1
Motor shield for Arduino
http://www.ladyada.net/make/mshield
TITLE: mshield2
Document Number:
REV:
Date: 2/02/2008 07:15:32p
Sheet: 1/1



Τα διαγράμματα αυτά παρουσιάζουν την ηλεκτρική σύνδεση μεταξύ μικροελεγκτή και motor shield. Έτσι βλέπουμε ότι το PWM σήμα PWM2A ξεκινάει από τον ακροδέκτη 17 του μικροεπεξεργαστή, καταλήγει στον ακροδέκτη 4 του module J3, από εκεί “ανεβαίνει” στο motor shield και καταλήγει στην είσοδο 1,2 EN της γέφυρας. Παρομοίως το PWM2B ξεκινάει από τον ακροδέκτη 5 του μικροεπεξεργαστή και καταλήγει στον ακροδέκτη 4 του module J1. Από εκεί “ανεβαίνει” στο motor shield και καταλήγει στην είσοδο 3,4 EN της ίδιας γέφυρας. Βλέπουμε ακόμα ότι το σήμα DIR_SER ξεκινάει από τον ακροδέκτη 14 του μικροεπεξεργαστή (= ακροδέκτης #8 του Arduino board), καταλήγει στον ακροδέκτη 1 του J3, από εκεί “ανεβαίνει” στο motor shield και από εκεί καταλήγει στο pin 14 του καταχωρητή. Είπαμε προηγουμένως ότι στην είσοδο αυτή (DIR_SER) εφαρμόζονται τα δεδομένα σε μορφή bits τα οποία στη συνέχεια προωθούνται στην παράλληλη έξοδο και από εκεί στις αντίστοιχες εισόδους της κάθε γέφυρας. Το ποια τιμή θα πάρουν τα bits αυτά καθορίζεται από την τιμή που παίρνει η μεταβλητή **latch_state**.

static uint8_t **latch_state**;

Η μεταβλητή αυτή έχει το χαρακτηριστικό ότι έχει μέγεθος (πλάτος) ακριβώς 8 bit. Έτσι προσομοιώνουμε έναν καταχωριτή 8 bit με την τιμή του να μεταβάλλεται ανάλογα με το ποια bit γίνονται set και ποια όχι.

Στη συνέχεια η μεταβλητή αυτή εφαρμόζεται στην είσοδο του shift register 74HCT595N από εκεί στην έξοδό του, και κατόπιν στις εισόδους της γέφυρας. Δηλαδή στις εισόδους 1A,2 A,3 A και 4 A εκείνης της γέφυρας που χρησιμοποιούμε κάθε φορά αναλόγως την εφαρμογή μας. Μπορούμε να χρησιμοποιούμε μόνο την μία από τις δύο γέφυρες ή και τις δύο εναλλάξ.

Με τις παρακάτω εντολές ορίζουμε ποιές εξόδοι (από τις 8 εξόδους) του shift register θα χρησιμοποιηθούν. Από τις οκτώ εξόδους συνολικά, επιλέγουμε κάθε φορά τις τέσσερις ανάλογα με το ποιόν κινητήρα θέλουμε να λειτουργήσουμε. Αυτό γίνεται μέσω της μεταβλητής steppernum.

```
uint8_t AF_Stepper::onestep(uint8_t dir, uint8_t style) {
uint8_t a, b, c, d;
uint8_t ocrb, ocra;

 ocra = ocrb = 255;

 if (steppernum == 1) {
 a = _BV(MOTOR1_A);
 b = _BV(MOTOR2_A);
 c = _BV(MOTOR1_B);
 d = _BV(MOTOR2_B);
 } else if (steppernum == 2) {
  a = _BV(MOTOR3_A);
  b = _BV(MOTOR4_A);
  c = _BV(MOTOR3_B);
  d = _BV(MOTOR4_B);
 } else {
 return 0;
 }
```

Στις παραπάνω εντολές όταν η μεταβλητή a πάρει τιμή λογικό HIGH, αυτό αντιστοιχεί σε διέγερση της φάσης A με θετική πολικότητα, ομοίως η μεταβλητή b σε διέγερση της φάσης B με θετική πολικότητα, η c σε διέγερση της A με αρνητική πολικότητα και η d σε διέγερση της B με αρνητική πολικότητα για κάθε έναν από τους δύο κινητήρες που μπορούμε να συνδέσουμε.

Έστω ότι θέλουμε ο κινητήρας να κινηθεί σε κατεύθυνση clockwise με διέγερση full step two phase on.
Αφού καθορίσαμε ποιές είναι οι έξοδοι του καταχωρητή που θα χρησιμοποιηθούν για την κίνηση του κινητήρα που θέλουμε, στη συνέχεια θα πρέπει να τους στείλουμε τις ανάλογες τιμές. Αυτό γίνεται με το να δώσουμε τις κατάλληλες τιμές **στα αντίστοιχα bits** της μεταβλητής latch_state και κατόπιν να καλέσουμε τη συνάρτηση latch_tx() η οποία στέλνει την μεταβλητή latch_state στον shift register.

Το παρακάτω τμήμα κώδικα δείχνει ότι για να βρούμε το πώς θα εκτελεσθεί το επόμενο βήμα πρέπει να εξετάσουμε την τιμή της παράστασης (currentstep/(MICROSTEPS/2)). Η τιμή της παράστασης αυτής προκύπτει από το ακέραιο μέρος του πηλίκου της διαίρεσης και βρίσκεται πάντα στο διάστημα [0….7]. Ανάλογα με την τιμή που δίνει η παράσταση αλλάζει και η τιμή της μεταβλητής **latch_state** και το αποτέλεσμα αντιστοιχεί στον τρόπο με τον οποίο θα γίνει η επόμενη διέγερση των τυλιγμάτων του κινητήρα. Αυτό γίνεται μέσω της προώθησης της τιμής της μεταβλητής στην παράλληλη έξοδο του shift register και στην

εφαρμογή αυτής της εξόδου στην είσοδο της γέφυρας που οδηγεί τον καθένα από τους δυο βηματικούς κινητήρες. Υποστηρίζονται όλοι οι τρόποι διέγερσης. Με τον τρόπο αυτό πετυχαίνουμε την καθορισμένη ακολουθία διέγερσης των τυλιγμάτων. Οι τέσσερις έξοδοι του καταχωρητή τροφοδοτούν την μια από τις δυο γέφυρες και οι υπόλοιπες τέσσερις τροφοδοτούν την άλλη. Ετσι μπορούμε να οδηγήσουμε μέχρι δυο βηματικούς κινητήρες.

```
latch_state &= ~a & ~b & ~c & ~d;                    // all motor pins to 0


switch (currentstep/(MICROSTEPS/2)) {
 case 0:
 latch_state |= a; // energize coil 1 only                    //A+
 break;
  case 1:
 latch_state |= a | b; // energize coil 1+2                    //A+,B+
 break;
   case 2:
 latch_state |= b; // energize coil 2 only                    //B+
break;
  case 3:
    latch_state |= b | c; // energize coil 2+3                 // B+,A-
  break;
 case 4:
 latch_state |= c; // energize coil 3 only                    //A-
 break;
  case 5:
    latch_state |= c | d; // energize coil 3+4                  //A-,B-
 break;
 case 6:
latch_state |= d; // energize coil 4 only                     //B-
 break;
 case 7:
  latch_state |= d | a; // energize coil 1+4                   //B-,A+
  break;
 }
 MC.latch_tx();                                              // αποστολή των δεδομένων

 return currentstep;
```

	# step	ΦΑΣΗ ΥΠΟ ΔΙΕΓΕΡΣΗ			
M1A		M1A	M2A	M1B	M2B
M2A		A+	B+	A-	B-
M1B	1	1	1	0	0
M2B	2	0	1	1	0
	3	0	0	1	1
	4	1	0	0	1

→ ακροδέκτης εξόδου #2 του shift register
→ ακροδέκτης εξόδου #1 του shift register
→ ακροδέκτης εξόδου #3 του shift register
→ ακροδέκτης εξόδου #4 του shift register

Άρα για οδήγηση full step, two phase πρώτα θα γίνουν set τα bits που αντιστοιχούν στις εξόδους 2 και 1 του shift register και θα εκτελεστεί το αντίστοιχο βήμα, στη συνέχεια θα γίνουν set τα bits που αντιστοιχούν στις εξόδους 1 και 3, θα εκτελεστεί το επόμενο βήμα κοκ.

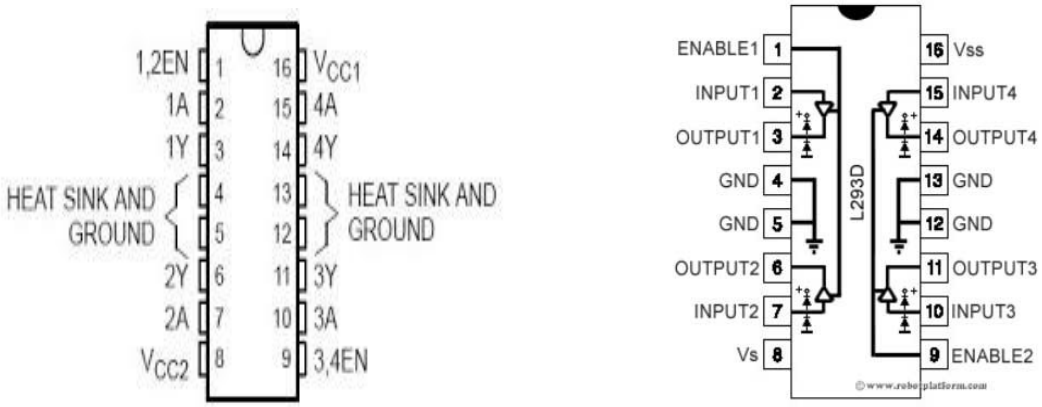
Αφού πάρει τιμή η **latch_state** εκτελούνται οι ακόλουθες εντολές που εφαρμόζουν την τιμή αυτή στον shift register. Κατόπιν η έξοδος του shift register εφαρμόζεται στην είσοδο του κυκλώματος οδήγησης και εκτελείται η επόμένη διέγερση άρα και το επόμενο βήμα.

```
void AFMotorController::latch_tx(void) {
uint8_t i;

 //LATCH_PORT &= ~_BV(LATCH);            //  η είσοδος DIR_LATCH → LOW
 digitalWrite(MOTORLATCH, LOW);

//SER_PORT &= ~_BV(SER);                 // η είσοδος DIR_SER → LOW
 digitalWrite(MOTORDATA, LOW);

 for (i=0; i<8; i++) {                  // για κάθε bit της μεταβλητής latch_state {

 //CLK_PORT &= ~_BV(CLK);                // η είσοδος DIR_CLOCK → LOW
  digitalWrite(MOTORCLK, LOW);

 if (latch_state & _BV(7-i)) {          // εάν το τρέχον bit της latch_state είναι “1” {
//SER_PORT |= _BV(SER);                  // η είσοδος DIR_SER → “HIGH” }
digitalWrite(MOTORDATA, HIGH);
 } else {                               // εάν όχι {
 //SER_PORT &= ~_BV(SER);               // η είσοδος DIR_SER → ‘’LOW ‘’}
  digitalWrite(MOTORDATA, LOW);
 }
 //CLK_PORT |= _BV(CLK);                // η είσοδος DIR_CLOCK → “HIGH”.
digitalWrite(MOTORCLK, HIGH);          //τα bits ένα ένα στο εσωτερικό του καταχωρητή
}
}
 //LATCH_PORT |= _BV(LATCH);            // η είσοδος DIR_LATCH →”HIGH”
 digitalWrite(MOTORLATCH, HIGH);       // όλη η μεταβλητή latch_state στην έξοδο του
// καταχωρητή (και τα 8 bits)  }
}
```

Ανακεφαλαιώνοντας η διαδικασία που ακολουθείται για την οδήγηση του κινητήρα είναι η παρακάτω:
Έστω ότι θέλουμε λειτουργία χωρίς microstepping. Τότε ο μικοεπεξεργαστής παράγει PWM σήμα με duty cicle 100%. Ταυτόχρονα στον μικροεπεξεργαστή εκτελείται ένα πρόγραμμα που αναθέτει τιμή στην μεταβλητή **latch_state**.
Η μεταβλητή αυτή εξέρχεται από την ψηφιακή έξοδο #8 του Arduino board και εφαρμόζεται στην είσοδο DIR_SER του serial to parallel 74HCT595N shift register και κατόπιν εμφανίζεται στην παράλληλη έξοδο του 74HCT595N. Η έξοδος αυτή μαζί με το PWM σήμα εφαρμόζεται στις εισόδους του L293D και ο κινητήρας περιστρέφεται αναλόγως.

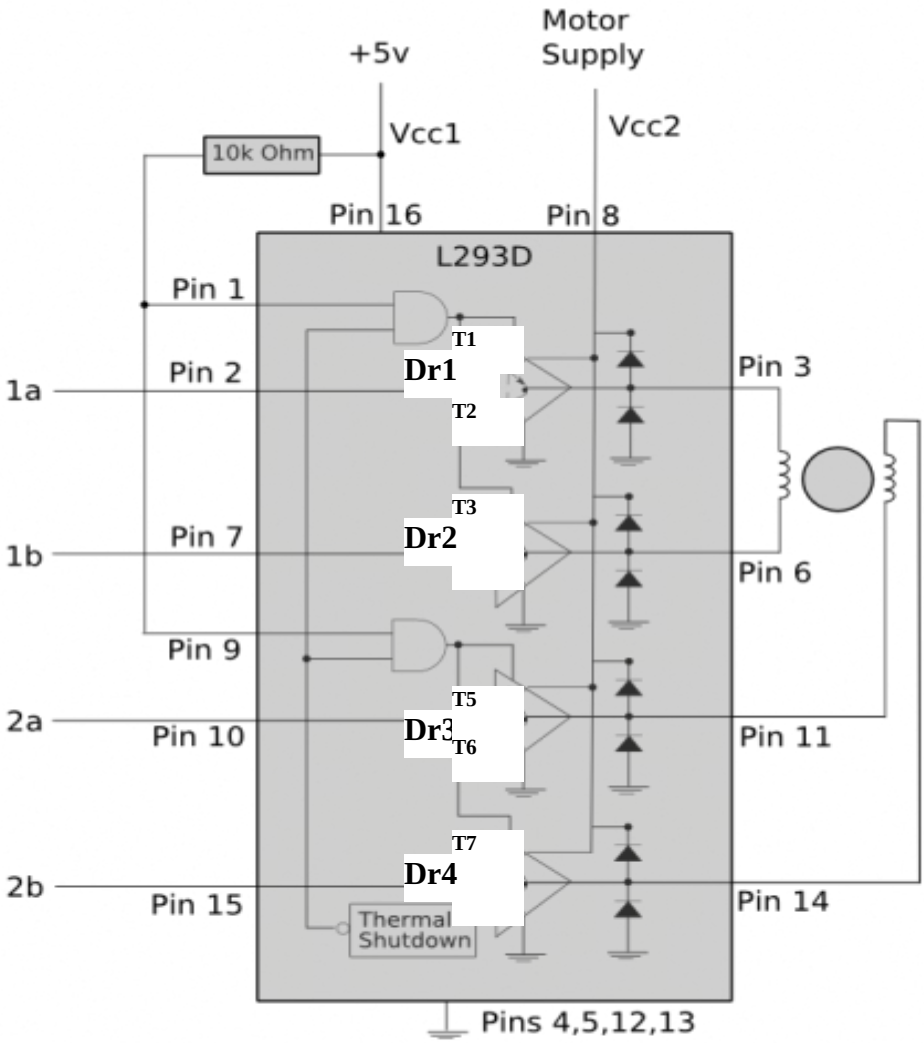
5.4 Το κύκλωμα οδήγησης L293D

Το ολοκληρωμένο κύκλωμα L293D (Σχήμα 5.4.1) είναι σχεδιασμένο για διπολική οδήγηση επαγωγικών φορτίων όπως είναι ο βηματικός κινητήρας και στην περίπτωση αυτή μπορεί να δώσει ρεύμα έως 600mA υπό τάση τροφοδοσίας από 4.5V έως 36V. Ο κινητήρας που χρησιμοποιούμε είναι στα 12V με ονομαστικό ρεύμα 330mA. Οπότε το συγκεκριμένο κύκλωμα είναι κατάλληλο για την οδήγησή του.
Είναι κατάλληλο για λειτουργία σε θερμοκρασίες από 0 ^{o}C έως 70 ^{o}C.



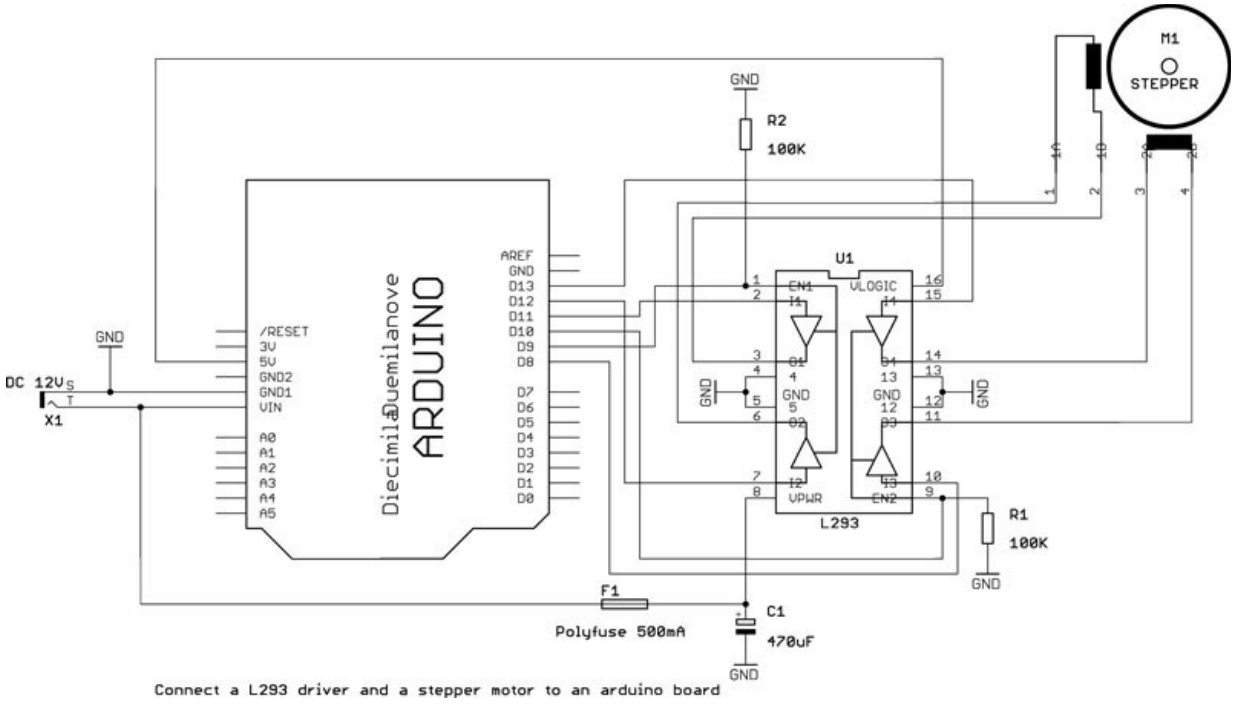
Σχήμα 5.4.1 Το ολοκληρωμένο κύκλωμα L293D

Διαθέτει τέσσερις εισόδους στα pins 2, 7, 10 και 15 (1A, 2A,3A, 4A) στις οποίες εφαρμόζονται τα ψηφιακά σήματα ελέγχου των φάσεων του κινητήρα
Επίσης διαθέτει τέσσερις εξόδους στα pins 3, 6, 11 και 14 (1Y, 2Y, 3Y, 4Y) σε κάθε μια από τις οποίες αντιστοιχεί από ένα πλήρες κύκλωμα διπολικής οδήγησης τύπου ημίσειας γέφυρας (half H driver) με το κάθε τέτοιο κύκλωμα να αποτελείται από δυο τρανζίστορ ισχύος που έχουν ρόλο ηλεκτρονικού διακόπτη. Το σημείο λειτουργίας των τρανζίστορ μετατοπίζεται από την αποκοπή στον κόρο ανάλογα με το ψηφιακό σήμα ελέγχου των φάσεων του κινητήρα το οποίο εφαρμόζεται στη βάση του κάθε τρανζίστορ. Συνολικά έχουμε τέσσερα τέτοια κυκλώματα (quadruble half H driver). Τα κυκλώματα αυτά ενεργοποιούνται σε ζεύγη μέσω των σημάτων ενεργοποίησης, δηλαδή μέσω των σημάτων enable και η ενεργοποίηση αυτή γίνεται όταν το σήμα enable πάρει τιμή “λογικό 1”. Συνολικά έχουμε δυο τέτοια σήματα. Το ENABLE1 ή 1,2 EN και το ENABLE 2 ή 3,4 EN.
Στο παρακάτω σχήμα (Σχήμα 5.4.2) φαίνεται με μεγαλύτερη λεπτομέρια η λειτουργία του κυκλώματος L293D σαν κύκλωμα διπολικής οδήγησης τύπου γέφυρας κατάλληλο για υβριδικό βηματικό κινητήρα 2 φάσεων. Ο half H driver Dr1 αποτελείται από τα τρανζίστορ T1 και T2, ο Dr2 από τα T3 και T4 κ.ο.κ
Στους ακροδέκτες 2, 7, 10 και 15 εφαρμόζονται τα ψηφιακά σήματα ελέγχου των φάσεων του κινητήρα με τον τρόπο που περιγράψαμε στη προηγούμενη παράγραφο.
Στους ακροδέκτες 1 και 9 εμφανίζονται τα 1,2 EN και 3,4 EN αντίστοιχα. Στην περίπτωση που το κύκλωμα της γέφυρας είναι συνδεδεμένο με μικροεπεξεργαστή (Σχήμα 5.4.3) τότε στους ακροδέκτες 1 και 9 εφαρμόζεται το PWM σήμα από τον μικροεπεξεργαστή. Το σήμα αυτό παίζει το ρόλο του σήματος enable. Για την περίπτωση που έχουμε PWM σήμα με duty cycle 100% μπορούμε να θεωρήσουμε ότι το σήμα αυτό ουσιαστικά ταυτίζεται με την τάση τροφοδοσίας 5 V, την τάση δηλαδή που χρησιμοποιούμε για την τροφοδοσία των λογικών κυκλωμάτων.
Αλλιώς (για την περίπτωση microstepping) το σήμα προέρχεται απευθείας από τις αντίστοιχες εξόδους του μικροεπεξεργαστή.



Σχήμα 5.4.2 Λειτουργία γέφυρας για κινητήρα 2 φάσεων

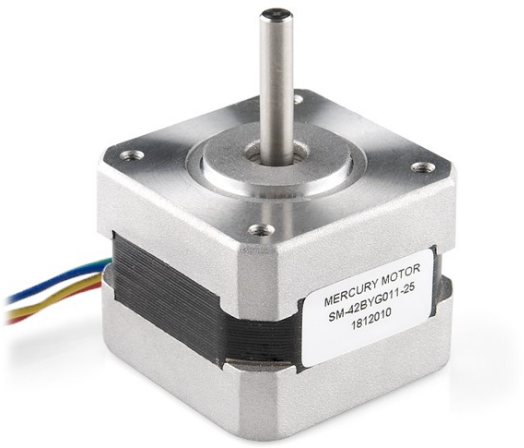
Στους ακροδέκτες 3, 6, 11 και 14 συνδέονται τα τυλίγματα (φάσεις) του κινητήρα. Το σήμα στον ακροδέκτη 1 ενεργοποιεί τους Dr1και Dr2 και η ροή του ρεύματος στα τυλίγματα του κινητήρα γίνεται από την πηγή τροφοδοσίας των τυλιγμάτων (τάση Vcc2 = 12VDC) κατόπιν μέσω του τρανζίστορ Τ1, μέσα από το τύλιγμα με φορά από τον ακροδέκτη 3 προς τον ακροδέκτη 6 και κατόπιν μέσω του τρανζίστορ Τ4 επιστρέφει στην πηγή. Η ροή του ρεύματος στο τύλιγμα κατά την αντίθετη έννοια - και σύμφωνα πάντα με τα ψηφιακά σήματα ελέγχου των φάσεων - γίνεται μέσω της πηγής τροφοδοσίας, κατόπιν μέσω του τρανζίστορ Τ3, στο τύλιγμα με φορά από τον ακροδέκτη 6 προς τον ακροδέκτη 3 και επιστροφή στην πηγή μέσω του τρανζίστορ Τ2. Τα παραπάνω ισχύουν για την μια φάση του βηματικού κινητήρα. Για την άλλη φάση ισχύουν ακριβώς τα αντίστοιχα. Το σήμα enable στον ακροδέκτη 9 ενεργοποιεί τους Dr3 και Dr4 με την κυκλοφορία του ρεύματος να γίνεται μέσω των τρανζίστορ Τ5, Τ8 και Τ7 , Τ6 τα οποία ενεργοποιούνται σε ζεύγη όπως και προηγουμένως.



Σχήμα 5.4. Σύνδεση μικροεπεξεργαστή - γέφυρας για διπολική οδήγηση

5.5 Ο βηματικός κινητήρας SM-42BYG011-25

Ο βηματικός κινητήρας που χρησιμοποιούμε στο σύστημα ελέγχου ανοικτού βρόχου και στις εφαρμογές είναι ο υβριδικός κινητήρας με κωδικό SM-42BYG011-25 της εταιρίας MERCURY MOTORS (βλ. Σχήμα 5.5.1).

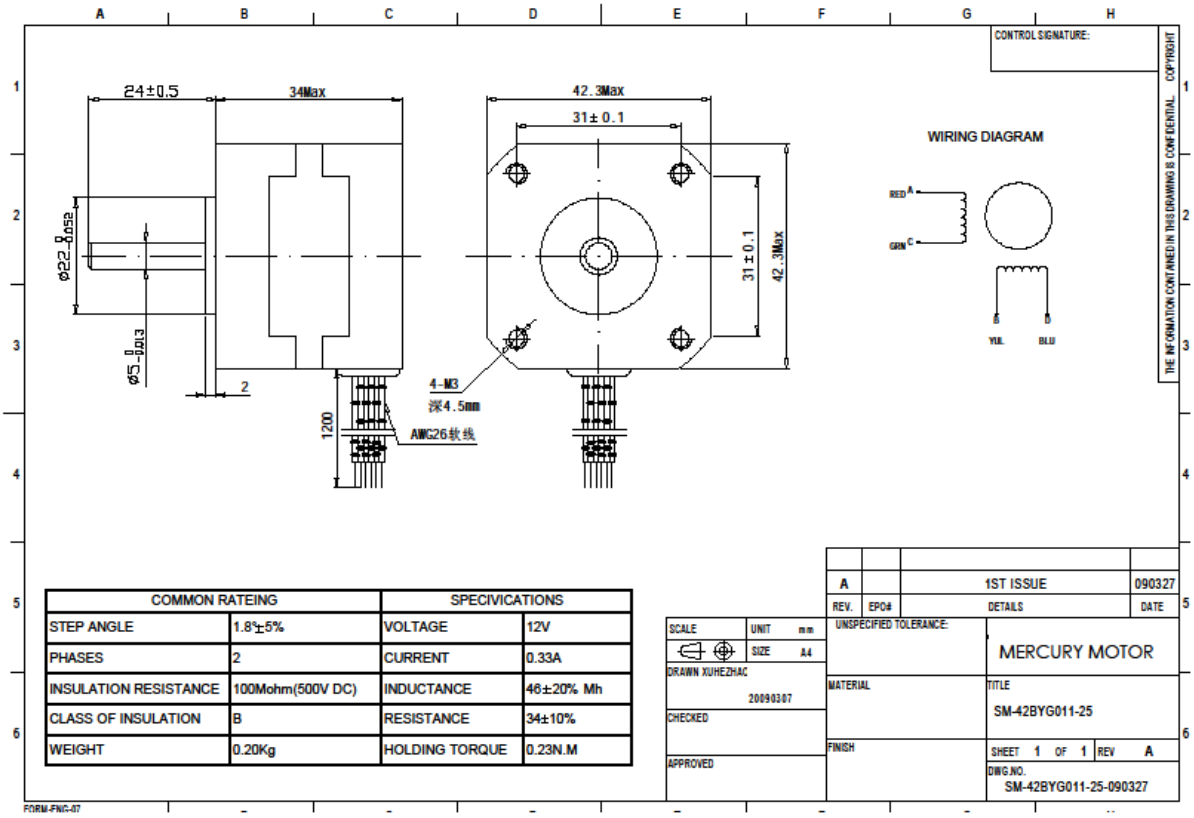


Σχήμα 5.5.1 Ο κινητήρας SM-42BYG011-25

ΒΑΣΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

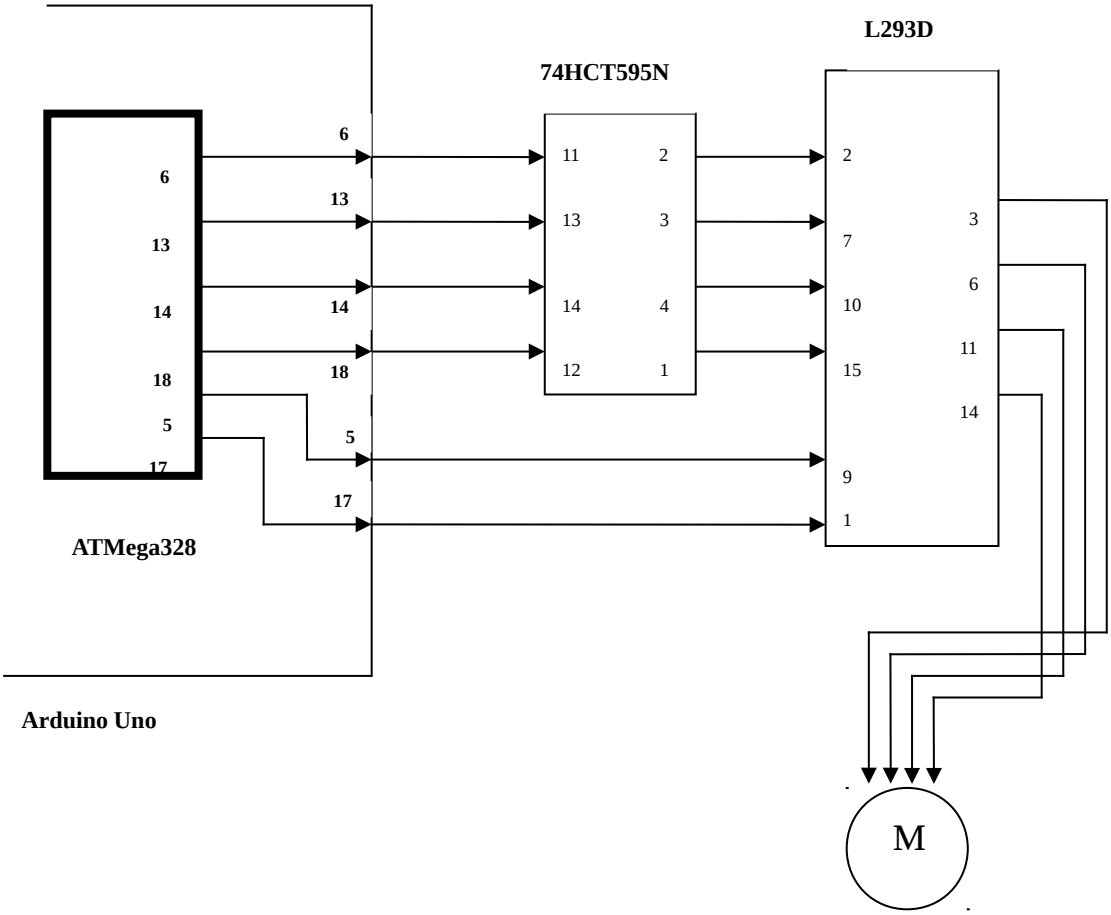
- Υβριδικός
- Κατάλληλος για διπολική οδήγηση
- Διαθέτει δυο τυλίγματα (φάσεις) με ακροδέκτες κόκκινο – πράσινο και κίτρινο – μπλέ
- 200 βήματα ανά περιστροφή (1.8 μοίρες)
- Τάση λειτουργίας 12V
- Ονομαστικό ρεύμα 330mA
- Ροπή στρέψης 0,23 Nm
- Βάρος 200g

Παρακάτω απεικονίζεται το datasheet του κινητήρα.



COMMON RATEING		SPECIVICATIONS	
STEP ANGLE	1.8%±5%	VOLTAGE	12V
PHASES	2	CURRENT	0.33A
INSULATION RESISTANCE	100Mohm(500V DC)	INDUCTANCE	46±20% Mh
CLASS OF INSULATION	B	RESISTANCE	34±10%
WEIGHT	0.20Kg	HOLDING TORQUE	0.23N.M

Παρακάτω φαίνεται ολοκληρωμένο το block διάγραμμα του συστήματος. Οι ακροδέκτες έχουν την φυσική αρίθμηση.

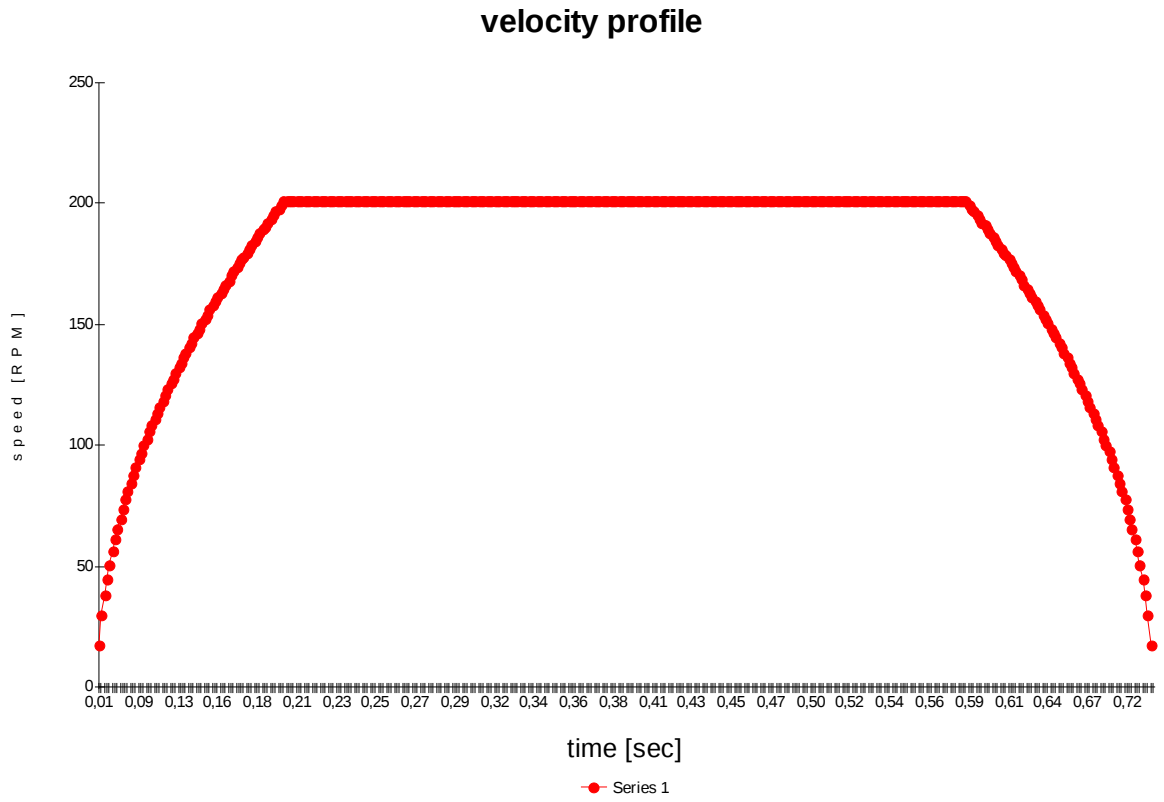

L293D
74HCT595N
6
13
14
18
5
17
ATMega328
Arduino Uno
11
13
14
12
2
3
4
1
2
7
10
15
9
1
3
6
11
14
M

6. Συμπεράσματα

- Ένα software – intensive σύστημα μπορεί να ελέγχει ταυτόχρονα την επιτάχυνση, την επιβράδυνση την ταχύτητα και τη θέση του άξονα του κινητήρα.
- Σε ένα software – intensive σύστημα οι απαιτήσεις μνήμης είναι ελάχιστες.
- Η σωστή υλοποίηση της επιτάχυνσης και της επιβράδυνσης είναι πολύ κρίσιμη για την επιτυχή σχεδίαση κάθε συστήματος που χρησιμοποιεί βηματικούς κινητήρες και έχει ως αποτέλεσμα σωστή, αποδοτική και σύμφωνα με τις προδιαγραφές λειτουργία της εφαρμογής.
- Μειονέκτημα μπορεί να αποτελέσει ο ακριβής υπολογισμός του χρόνου μεταξύ δυο διαδοχικών σημάτων ελέγχου φάσεων αφού γίνεται σε γλώσσα προγραμματισμού υψηλού επιπέδου (συνάρτηση setSpeed()). Το πρόβλημα μπορεί να λυθεί γράφοντας αυτό αλλά και άλλα κρίσιμα τμήματα του προγράμματος σε γλώσσα μηχανής.
- Μπορούν να αναπτυχθούν ακριβής προσεγγιστικές μέθοδοι σε γλώσσα υψηλού επιπέδου (όπως η C), που να υπολογίζουν τον χρονισμό των παλμών σε πραγματικό χρόνο ελαχιστοποιώντας την απαιτούμενη υπολογιστική ισχύ και τις απαιτήσεις σε μνήμη και που ταυτόχρονα να εξασφαλίζουν ένα αρκετά ομαλό προφίλ ταχύτητας.
- Ένα software – intensive σύστημα μας επιτρέπει λεπτομερή έλεγχο του προφίλ ταχύτητας σε ταχύτητες μέχρι 1000 βήματα / sec οι οποίες όμως με βάση την πρόοδο και τις απαιτήσεις της σύγχρονης τεχνολογίας χαρακτηρίζονται ως “μεσαίες”. Ετσι ένα τέτοιο σύστημα ταιριάζει περισσότερο σε εφαρμογές που είναι απαραίτητη η γρήγορη επιτάχυνση και επιβράδυνση ώστε ο άξονας του κινητήρα να φτάσει τη θέση – στόχο σε σύντομο χρονικό διάστημα παρά η επίτευξη υψηλής τελικής ταχύτητας.
- Πλεονέκτημα της υλοποίησης ενός συστήματος ελέγχου ανοικτού βρόχου με χρήση μικροεπεξεργαστή είναι η απλή φυσική και ηλεκτρονική διασύνδεση μικροεπεξεργαστή – κινητήρα η οποία γίνεται με απευθείας χρήση των εξόδων του καταχωρητή 74HCT595 μέσω των οποίων μεταφέρονται τα σήματα ελέγχου των φάσεων του κινητήρα στο κύκλωμα οδήγησης.

ΠΑΡΑΡΤΗΜΑ 1 - Προφίλ ταχύτητας βηματικού κινητήρα

Έστω ότι χρησιμοποιούμε κινητήρα με start / stop rate 50 rpm και pull out rate 200 rpm.
Ο κινητήρας εκτελεί μια πλήρη περιστροφή σε 200 βήματα.
Έχουμε ως προδιαγραφή για τη χρήση του κινητήρα σε κάποια εφαρμογή την εξής:
Θέλουμε ο άξονας του κινητήρα να διανύσει απόσταση που αντιστοιχεί σε 400 βήματα σε χρόνο 0,7 sec με ανοχή από -5% έως +5% στην χρονική διάρκεια. Αυτό γιατί τότε θα εκτελεσθεί κάποια άλλη κίνηση.
Θέλουμε δηλαδή ο κινητήρας να εκτελέσει προκαθορισμένο αριθμό βημάτων σε προκαθορισμένο χρονικό διάστημα.
Για να πετύχουμε την προδιαγραφή εκκινούμε τον κινητήρα με ταχύτητα μικρότερη από την ταχύτητα start / stop και επιταχύνουμε μέχρι την ταχύτητα pull out.
Η επιτάχυνση είναι 100 rad / sec/ sec.
Παρατηρούμε ότι η προδιαγραφή που έχουμε ως προς τον χρόνο δεν ικανοποιείται καθώς η κίνηση ολοκληρώνεται σε χρόνο 0,79 sec. Για να ικανοποιήσουμε την προδιαγραφή θα πρέπει να επιλέξουμε μεγαλύτερη επιτάχυνση υπό την προϋπόθεση ότι ο κινητήρας δεν θα εκκινεί και δεν θα σταματά σε ταχύτητα μεγαλύτερη από την ταχύτητα start / stop. Εάν αυτό δεν μπορεί να επιτευχθεί θα πρέπει να επιλέξουμε κινητήρα με μεγαλύτερη ταχύτητα pull out.



spr=	200
frequency:=	1MHz
a=	0,0314
acceleration(rad/s/s):=	100

accumulative time (sec)	***δt(sec)***		***elapsed***	***perstep***			
		n		***Ci***	***rad/sec***	**RPM**	**steps/sec**
0,01694	0,01694	0		**16.941**	1,854	**18**	59
0,02710	0,01016	1	16.941	**10.164**	3,091	**30**	98
0,03501	0,00791	2	27.105	**7.906**	3,974	**38**	126
0,04170	0,00669	3	35.010	**6.689**	4,696	**45**	149
0,04760	0,00590	4	41.700	**5.902**	5,322	**51**	169
0,05294	0,00534	5	47.602	**5.340**	5,882	**56**	187
0,05786	0,00491	6	52.942	**4.913**	6,394	**61**	204
0,06243	0,00457	7	57.855	**4.574**	6,868	**66**	219
0,06673	0,00430	8	62.429	**4.297**	7,311	**70**	233
0,07079	0,00406	9	66.726	**4.065**	7,728	**74**	246
0,07466	0,00387	10	70.791	**3.866**	8,125	**78**	259
0,07835	0,00369	11	74.658	**3.695**	8,503	**81**	271
0,08190	0,00354	12	78.352	**3.544**	8,864	**85**	282
0,08531	0,00341	13	81.896	**3.410**	9,212	**88**	293
0,08860	0,00329	14	85.306	**3.290**	9,547	**91**	304
0,09178	0,00318	15	88.596	**3.183**	9,871	**94**	314
0,09486	0,00308	16	91.779	**3.085**	10,184	**97**	324
0,09786	0,00300	17	94.863	**2.995**	10,488	**100**	334
0,10077	0,00291	18	97.859	**2.913**	10,784	**103**	343
0,10361	0,00284	19	100.772	**2.837**	11,071	**106**	352
0,10638	0,00277	20	103.609	**2.767**	11,351	**108**	361
0,10908	0,00270	21	106.377	**2.702**	11,625	**111**	370
0,11172	0,00264	22	109.079	**2.642**	11,892	**114**	379
0,11431	0,00258	23	111.720	**2.585**	12,153	**116**	387
0,11684	0,00253	24	114.305	**2.531**	12,409	**119**	395
0,11932	0,00248	25	116.837	**2.481**	12,660	**121**	403
0,12175	0,00243	26	119.318	**2.434**	12,906	**123**	411
0,12414	0,00239	27	121.752	**2.389**	13,147	**126**	419
0,12649	0,00235	28	124.141	**2.347**	13,384	**128**	426
0,12880	0,00231	29	126.488	**2.307**	13,617	**130**	433
0,13106	0,00227	30	128.795	**2.269**	13,846	**132**	441
0,13330	0,00223	31	131.064	**2.233**	14,071	**134**	448
0,13549	0,00220	32	133.297	**2.198**	14,292	**136**	455
0,13766	0,00216	33	135.495	**2.165**	14,511	**139**	462
0,13979	0,00213	34	137.660	**2.133**	14,725	**141**	469
0,14190	0,00210	35	139.793	**2.103**	14,937	**143**	476
0,14397	0,00207	36	141.896	**2.074**	15,146	**145**	482
0,14602	0,00205	37	143.970	**2.046**	15,352	**147**	489
0,14804	0,00202	38	146.016	**2.019**	15,556	**149**	495
0,15003	0,00199	39	148.036	**1.994**	15,756	**150**	502
0,15200	0,00197	40	150.029	**1.969**	15,955	**152**	508
0,15394	0,00195	41	151.998	**1.945**	16,150	**154**	514
0,15587	0,00192	42	153.943	**1.922**	16,344	**156**	520
0,15777	0,00190	43	155.865	**1.900**	16,535	**158**	526
0,15964	0,00188	44	157.765	**1.878**	16,724	**160**	532
0,16150	0,00186	45	159.644	**1.858**	16,911	**161**	538
0,16334	0,00184	46	161.501	**1.838**	17,096	**163**	544
0,16516	0,00182	47	163.339	**1.818**	17,278	**165**	550
0,16696	0,00180	48	165.157	**1.799**	17,459	**167**	556
0,16874	0,00178	49	166.956	**1.781**	17,638	**168**	561
0,17050	0,00176	50	168.737	**1.763**	17,816	**170**	567
0,17225	0,00175	51	170.500	**1.746**	17,991	**172**	573
0,17398	0,00173	52	172.246	**1.729**	18,165	**173**	578
0,17569	0,00171	53	173.976	**1.713**	18,337	**175**	584
0,17739	0,00170	54	175.689	**1.697**	18,508	**177**	589

0,17907	0,00168	55	177.386	**1.682**	18,677	**178**	595
0,18074	0,00167	56	179.068	**1.667**	18,844	**180**	600
0,18239	0,00165	57	180.735	**1.652**	19,010	**182**	605
0,18403	0,00164	58	182.388	**1.638**	19,175	**183**	610
0,18565	0,00162	59	184.026	**1.624**	19,338	**185**	616
0,18726	0,00161	60	185.650	**1.611**	19,500	**186**	621
0,18886	0,00160	61	187.261	**1.598**	19,660	**188**	626
0,19044	0,00158	62	188.859	**1.585**	19,820	**189**	631
0,19202	0,00157	63	190.444	**1.572**	19,978	**191**	636
0,19358	0,00156	64	192.016	**1.560**	20,134	**192**	641
0,19512	0,00155	65	193.577	**1.548**	20,290	**194**	646
0,19666	0,00154	66	195.125	**1.537**	20,444	**195**	651
0,19819	0,00153	67	196.661	**1.525**	20,597	**197**	656
0,19970	0,00151	68	198.187	**1.514**	20,749	**198**	661
0,20120	0,00150	69	199.701	**1.503**	20,900	**200**	665
0,20270	0,00149	70	201.204	**1.492**	21,050	**201**	670
0,20419	0,00149	71	202.696	**1.492**	21,055	**201**	670
0,20568	0,00149	72	204.188	**1.492**	21,055	**201**	670
0,20717	0,00149	73	205.680	**1.492**	21,055	**201**	670
0,20866	0,00149	74	207.172	**1.492**	21,055	**201**	670
0,21016	0,00149	75	208.664	**1.492**	21,055	**201**	670
0,21165	0,00149	76	210.156	**1.492**	21,055	**201**	670
0,21314	0,00149	77	211.648	**1.492**	21,055	**201**	670
0,21463	0,00149	78	213.140	**1.492**	21,055	**201**	670
0,21612	0,00149	79	214.632	**1.492**	21,055	**201**	670
0,21762	0,00149	80	216.124	**1.492**	21,055	**201**	670
0,21911	0,00149	81	217.616	**1.492**	21,055	**201**	670
0,22060	0,00149	82	219.108	**1.492**	21,055	**201**	670
0,22209	0,00149	83	220.600	**1.492**	21,055	**201**	670
0,22358	0,00149	84	222.092	**1.492**	21,055	**201**	670
0,22508	0,00149	85	223.584	**1.492**	21,055	**201**	670
0,22657	0,00149	86	225.076	**1.492**	21,055	**201**	670
0,22806	0,00149	87	226.568	**1.492**	21,055	**201**	670
0,22955	0,00149	88	228.060	**1.492**	21,055	**201**	670
0,23104	0,00149	89	229.552	**1.492**	21,055	**201**	670
0,23254	0,00149	90	231.044	**1.492**	21,055	**201**	670
0,23403	0,00149	91	232.536	**1.492**	21,055	**201**	670
0,23552	0,00149	92	234.028	**1.492**	21,055	**201**	670
0,23701	0,00149	93	235.520	**1.492**	21,055	**201**	670
0,23850	0,00149	94	237.012	**1.492**	21,055	**201**	670
0,24000	0,00149	95	238.504	**1.492**	21,055	**201**	670
0,24149	0,00149	96	239.996	**1.492**	21,055	**201**	670
0,24298	0,00149	97	241.488	**1.492**	21,055	**201**	670
0,24447	0,00149	98	242.980	**1.492**	21,055	**201**	670
0,24596	0,00149	99	244.472	**1.492**	21,055	**201**	670
0,24746	0,00149	100	245.964	**1.492**	21,055	**201**	670
0,24895	0,00149	101	247.456	**1.492**	21,055	**201**	670
0,25044	0,00149	102	248.948	**1.492**	21,055	**201**	670
0,25193	0,00149	103	250.440	**1.492**	21,055	**201**	670
0,25342	0,00149	104	251.932	**1.492**	21,055	**201**	670
0,25492	0,00149	105	253.424	**1.492**	21,055	**201**	670
0,25641	0,00149	106	254.916	**1.492**	21,055	**201**	670
0,25790	0,00149	107	256.408	**1.492**	21,055	**201**	670
0,25939	0,00149	108	257.900	**1.492**	21,055	**201**	670
0,26088	0,00149	109	259.392	**1.492**	21,055	**201**	670
0,26238	0,00149	110	260.884	**1.492**	21,055	**201**	670
0,26387	0,00149	111	262.376	**1.492**	21,055	**201**	670
0,26536	0,00149	112	263.868	**1.492**	21,055	**201**	670
0,26685	0,00149	113	265.360	**1.492**	21,055	**201**	670
0,26834	0,00149	114	266.852	**1.492**	21,055	**201**	670
0,26984	0,00149	115	268.344	**1.492**	21,055	**201**	670
0,27133	0,00149	116	269.836	**1.492**	21,055	**201**	670
0,27282	0,00149	117	271.328	**1.492**	21,055	**201**	670
0,27431	0,00149	118	272.820	**1.492**	21,055	**201**	670
0,27580	0,00149	119	274.312	**1.492**	21,055	**201**	670
0,27730	0,00149	120	275.804	**1.492**	21,055	**201**	670
0,27879	0,00149	121	277.296	**1.492**	21,055	**201**	670
0,28028	0,00149	122	278.788	**1.492**	21,055	**201**	670
0,28177	0,00149	123	280.280	**1.492**	21,055	**201**	670
0,28326	0,00149	124	281.772	**1.492**	21,055	**201**	670
0,28476	0,00149	125	283.264	**1.492**	21,055	**201**	670

0,28625	0,00149	126	284.756	**1.492**	21,055	**201**	670
0,28774	0,00149	127	286.248	**1.492**	21,055	**201**	670
0,28923	0,00149	128	287.740	**1.492**	21,055	**201**	670
0,29072	0,00149	129	289.232	**1.492**	21,055	**201**	670
0,29222	0,00149	130	290.724	**1.492**	21,055	**201**	670
0,29371	0,00149	131	292.216	**1.492**	21,055	**201**	670
0,29520	0,00149	132	293.708	**1.492**	21,055	**201**	670
0,29669	0,00149	133	295.200	**1.492**	21,055	**201**	670
0,29818	0,00149	134	296.692	**1.492**	21,055	**201**	670
0,29968	0,00149	135	298.184	**1.492**	21,055	**201**	670
0,30117	0,00149	136	299.676	**1.492**	21,055	**201**	670
0,30266	0,00149	137	301.168	**1.492**	21,055	**201**	670
0,30415	0,00149	138	302.660	**1.492**	21,055	**201**	670
0,30564	0,00149	139	304.152	**1.492**	21,055	**201**	670
0,30714	0,00149	140	305.644	**1.492**	21,055	**201**	670
0,30863	0,00149	141	307.136	**1.492**	21,055	**201**	670
0,31012	0,00149	142	308.628	**1.492**	21,055	**201**	670
0,31161	0,00149	143	310.120	**1.492**	21,055	**201**	670
0,31310	0,00149	144	311.612	**1.492**	21,055	**201**	670
0,31460	0,00149	145	313.104	**1.492**	21,055	**201**	670
0,31609	0,00149	146	314.596	**1.492**	21,055	**201**	670
0,31758	0,00149	147	316.088	**1.492**	21,055	**201**	670
0,31907	0,00149	148	317.580	**1.492**	21,055	**201**	670
0,32056	0,00149	149	319.072	**1.492**	21,055	**201**	670
0,32206	0,00149	150	320.564	**1.492**	21,055	**201**	670
0,32355	0,00149	151	322.056	**1.492**	21,055	**201**	670
0,32504	0,00149	152	323.548	**1.492**	21,055	**201**	670
0,32653	0,00149	153	325.040	**1.492**	21,055	**201**	670
0,32802	0,00149	154	326.532	**1.492**	21,055	**201**	670
0,32952	0,00149	155	328.024	**1.492**	21,055	**201**	670

Η περιστροφή του κινητήρα γίνεται με σταθερή ταχύτητα 200rpm

0,54735	0,00149	301	545.856	**1.492**	21,055	**201**	670
0,54884	0,00149	302	547.348	**1.492**	21,055	**201**	670
0,55033	0,00149	303	548.840	**1.492**	21,055	**201**	670
0,55182	0,00149	304	550.332	**1.492**	21,055	**201**	670
0,55332	0,00149	305	551.824	**1.492**	21,055	**201**	670
0,55481	0,00149	306	553.316	**1.492**	21,055	**201**	670
0,55630	0,00149	307	554.808	**1.492**	21,055	**201**	670
0,55779	0,00149	308	556.300	**1.492**	21,055	**201**	670
0,55928	0,00149	309	557.792	**1.492**	21,055	**201**	670
0,56078	0,00149	310	559.284	**1.492**	21,055	**201**	670
0,56227	0,00149	311	560.776	**1.492**	21,055	**201**	670
0,56376	0,00149	312	562.268	**1.492**	21,055	**201**	670
0,56525	0,00149	313	563.760	**1.492**	21,055	**201**	670
0,56674	0,00149	314	565.252	**1.492**	21,055	**201**	670
0,56824	0,00149	315	566.744	**1.492**	21,055	**201**	670
0,56973	0,00149	316	568.236	**1.492**	21,055	**201**	670
0,57122	0,00149	317	569.728	**1.492**	21,055	**201**	670
0,57271	0,00149	318	571.220	**1.492**	21,055	**201**	670
0,57420	0,00149	319	572.712	**1.492**	21,055	**201**	670
0,57570	0,00149	320	574.204	**1.492**	21,055	**201**	670
0,57719	0,00149	321	575.696	**1.492**	21,055	**201**	670
0,57868	0,00149	322	577.188	**1.492**	21,055	**201**	670
0,58017	0,00149	323	578.680	**1.492**	21,055	**201**	670
0,58166	0,00149	324	580.172	**1.492**	21,055	**201**	670
0,58316	0,00149	325	581.664	**1.492**	21,055	**201**	670
0,58465	0,00149	326	583.156	**1.492**	21,055	**201**	670
0,58614	0,00149	327	584.648	**1.492**	21,055	**201**	670
0,58763	0,00149	328	586.140	**1.492**	21,055	**201**	670
0,58912	0,00149	329	587.632	**1.492**	21,055	**201**	670
0,59063	0,00150	330	589.124	**1.503**	20,905	**200**	665
0,59214	0,00151	331	590.627	**1.514**	20,754	**198**	661
0,59367	0,00152	332	592.140	**1.525**	20,602	**197**	656
0,59520	0,00154	333	593.665	**1.536**	20,449	**195**	651
0,59675	0,00155	334	595.201	**1.548**	20,294	**194**	646
0,59831	0,00156	335	596.749	**1.560**	20,139	**192**	641
0,59988	0,00157	336	598.309	**1.572**	19,982	**191**	636
0,60147	0,00158	337	599.881	**1.585**	19,824	**189**	631

0,60306	0,00160	338	601.466	**1.597**	19,665	**188**	626
0,60467	0,00161	339	603.063	**1.611**	19,504	**186**	621
0,60630	0,00162	340	604.674	**1.624**	19,343	**185**	616
0,60794	0,00164	341	606.298	**1.638**	19,179	**183**	611
0,60959	0,00165	342	607.936	**1.652**	19,015	**182**	605
0,61125	0,00167	343	609.588	**1.667**	18,849	**180**	600
0,61294	0,00168	344	611.254	**1.682**	18,681	**178**	595
0,61463	0,00170	345	612.936	**1.697**	18,512	**177**	589
0,61635	0,00171	346	614.633	**1.713**	18,341	**175**	584
0,61807	0,00173	347	616.346	**1.729**	18,169	**174**	578
0,61982	0,00175	348	618.075	**1.746**	17,995	**172**	573
0,62158	0,00176	349	619.820	**1.763**	17,820	**170**	567
0,62336	0,00178	350	621.583	**1.781**	17,642	**168**	562
0,62516	0,00180	351	623.364	**1.799**	17,463	**167**	556
0,62698	0,00182	352	625.162	**1.818**	17,282	**165**	550
0,62882	0,00184	353	626.980	**1.837**	17,099	**163**	544
0,63067	0,00186	354	628.817	**1.857**	16,915	**162**	538
0,63255	0,00188	355	630.674	**1.878**	16,728	**160**	532
0,63445	0,00190	356	632.552	**1.899**	16,539	**158**	526
0,63637	0,00192	357	634.452	**1.922**	16,348	**156**	520
0,63832	0,00194	358	636.373	**1.945**	16,154	**154**	514
0,64029	0,00197	359	638.318	**1.968**	15,958	**152**	508
0,64228	0,00199	360	640.287	**1.993**	15,760	**151**	502
0,64430	0,00202	361	642.280	**2.019**	15,559	**149**	495
0,64634	0,00205	362	644.299	**2.046**	15,356	**147**	489
0,64842	0,00207	363	646.344	**2.074**	15,150	**145**	482
0,65052	0,00210	364	648.418	**2.103**	14,941	**143**	476
0,65265	0,00213	365	650.521	**2.133**	14,729	**141**	469
0,65482	0,00216	366	652.653	**2.164**	14,514	**139**	462
0,65702	0,00220	367	654.818	**2.197**	14,296	**137**	455
0,65925	0,00223	368	657.015	**2.232**	14,074	**134**	448
0,66152	0,00227	369	659.247	**2.268**	13,849	**132**	441
0,66382	0,00231	370	661.516	**2.306**	13,620	**130**	434
0,66617	0,00235	371	663.822	**2.347**	13,387	**128**	426
0,66856	0,00239	372	666.169	**2.389**	13,150	**126**	419
0,67099	0,00243	373	668.557	**2.434**	12,909	**123**	411
0,67347	0,00248	374	670.991	**2.481**	12,663	**121**	403
0,67600	0,00253	375	673.472	**2.531**	12,412	**119**	395
0,67859	0,00258	376	676.003	**2.584**	12,156	**116**	387
0,68123	0,00264	377	678.587	**2.641**	11,895	**114**	379
0,68393	0,00270	378	681.228	**2.702**	11,628	**111**	370
0,68670	0,00277	379	683.929	**2.767**	11,354	**108**	361
0,68953	0,00284	380	686.696	**2.837**	11,074	**106**	353
0,69245	0,00291	381	689.533	**2.912**	10,786	**103**	343
0,69544	0,00299	382	692.445	**2.994**	10,490	**100**	334
0,69852	0,00308	383	695.440	**3.084**	10,186	**97**	324
0,70171	0,00318	384	698.524	**3.182**	9,873	**94**	314
0,70500	0,00329	385	701.705	**3.290**	9,549	**91**	304
0,70840	0,00341	386	704.995	**3.409**	9,214	**88**	293
0,71195	0,00354	387	708.404	**3.543**	8,866	**85**	282
0,71564	0,00369	388	711.947	**3.694**	8,505	**81**	271
0,71951	0,00387	389	715.641	**3.866**	8,127	**78**	259
0,72357	0,00406	390	719.507	**4.064**	7,730	**74**	246
0,72787	0,00430	391	723.570	**4.296**	7,312	**70**	233
0,73244	0,00457	392	727.866	**4.573**	6,869	**66**	219
0,73735	0,00491	393	732.439	**4.912**	6,395	**61**	204
0,74269	0,00534	394	737.351	**5.339**	5,884	**56**	187
0,74859	0,00590	395	742.690	**5.901**	5,323	**51**	169
0,75528	0,00669	396	748.591	**6.688**	4,697	**45**	150
0,76318	0,00790	397	755.279	**7.904**	3,975	**38**	127
0,77334	0,01016	398	763.183	**10.162**	3,091	**30**	98
0,79028	0,01694	399	773.345	**16.937**	1,855	**18**	59
0,77334	-0,01694	400	790.282	**-16.937**	-1,855	**-18**	-59

Εάν διπλασιάσουμε την επιτάχυνση σε 200 rad / sec / sec τότε πετυχαίνουμε την προδιαγραφή που έχουμε θέσει ως προς τη χρονική διάρκεια της κίνησης μιας και ο κινητήρας ολοκληρώνει την πορεία του μέχρι το στόχο σε χρόνο ίσο με 0,6856 sec . Τώρα η ράμπα της επιβράδυνσης έχει διάρκεια 34 βήματα από 70 προηγουμένως.

ΠΑΡΑΡΤΗΜΑ 2 - Προγραμματιστικές Εφαρμογές

sketch_01 – περιστροφή κινητήρα με διέγερση one phase on full step

Με το sketch αυτό ο κινητήρας περιστρέφεται συνεχώς με κατεύθυνση clockwise

```
//sketch_01
// ο κινητήρας περιστρέφεται συνεχώς σε διέγερση one phase on full step
// με ταχύτητα 20 rpm


#include <AFMotor.h>

AF_Stepper motor(200, 2);


void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  motor.setSpeed(20);  // 20 rpm

}

void loop() {
 motor.step(1,FORWARD, SINGLE);

}
```

sketch_02 – περιστροφή κινητήρα με διέγερση two phase on full step

Με αυτό το sketch ο κινητήρας περιστρέφεται με κατεύθυνση counterclockwise. Χαρακτηριστικό του προγράμματος είναι η χρήση του for loop

```
//sketch 02
// ο κινητήρας περιστρέφεται για ορισμένο αριθμό σε διέγερση two phase on full step
// με ταχύτητα 20 rpm

#include <AFMotor.h>

AF_Stepper motor(200, 1);

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  motor.setSpeed(20);  // 20 rpm
}

void loop() {
  for (int i=0; i<5; i++){
 motor.step(200,BACKWARD , DOUBLE);
 motor.release();
  }
  delay(10000);
}
```

sketch_03 – περιστροφή κινητήρα με όλους τους δυνατούς τρόπους διέγερσης

Εδώ γίνεται επίδειξη όλων των δυνατών τρόπων διέγερσης των τυλιγμάτων του κινητήρα. Επίσης ο κινητήρας κινείται και προς τις δυο κατευθύνσεις. Έχουμε διέγερση full step με one phase on και two phase on, αλλά και διέγερση half step και microstepping. Παρατηρούμε πως η διέγερση half step έχει σαν αποτέλεσμα το κάθε βήμα του κινητήρα να είναι το μισό σε σχέση με το κανονικό και έτσι ο κινητήρας για ίδιο αριθμό βημάτων να καλύπτει την μισή απόσταση. Επίσης παρατηρούμε και την κίνηση του άξονα με το microstepping που έχει ως πλεονέκτημα την ομαλότερη κίνηση του άξονα του κινητήρα και την απαλοιφή των συντονισμών.

```
#include <AFMotor.h>


AF_Stepper motor(200, 2);


void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper test!");

  motor.setSpeed(10);  // 10 rpm

  motor.step(100, FORWARD, SINGLE);
  motor.release();
  delay(1000);
}

void loop() {
  motor.step(100, FORWARD, SINGLE);
  motor.step(100, BACKWARD, SINGLE);

  motor.step(100, FORWARD, DOUBLE);
  motor.step(100, BACKWARD, DOUBLE);

  motor.step(100, FORWARD, INTERLEAVE);
  motor.step(100, BACKWARD, INTERLEAVE);

  motor.step(100, FORWARD, MICROSTEP);
  motor.step(100, BACKWARD, MICROSTEP);

  delay(2000);
}
```

sketch_04 – λειτουργία βήμα προς βήμα

Εδώ ο κινητήρας λειτουργεί βήμα προς βήμα εκτελώντας μια πλήρη περιστροφή και κατόπιν σταματά. Εάν θέλουμε μπορούμε να επαναλάβουμε πιέζοντας το μπουτόν reset στο Motor Shield

```
#include <AFMotor.h>

AF_Stepper Stepper1(200,1);

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

  for(int i=0;i<200;i++){
    Stepper1.onestep(FORWARD, DOUBLE);

    delay(500);         //500 ms
  }
  Stepper1.release();
}

void loop(){
}
```

sketch_05 – προκαθορισμένος αριθμός περιστροφών

Μπορούμε να προκαθορίσουμε τον αριθμό των περιστροφών που θα εκτελέσει ο κινητήρας. Εδώ ο κινητήρας περιστρέφεται για όσες φορές έχουμε προκαθορίσει μέσα από το sketch και κατόπιν σταματά. Και εδώ μπορούμε να επαναλάβουμε πιέζοντας το μπουτόν reset.

```
#include <AFMotor.h>

AF_Stepper motor1(200,2);

void setup(){

  Serial.begin(9600);
  motor1.setSpeed(20);

  for (int j=0;j<5;j++){
    for(int i=0; i<200;i++){
      motor1.step(1,BACKWARD,DOUBLE);

  }
    motor1.release();
  }
}

 void loop(){}
```

sketch_05a – προκαθορισμένος αριθμός περιστροφών

Μια διαφορετική υλοποίηση του προηγούμενου sketch

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 2);

void forwardstep() {
motor1.onestep(FORWARD, DOUBLE);
}
void backwardstep() {
motor1.onestep(BACKWARD, DOUBLE);
}


AccelStepper stepper(backwardstep, forwardstep);

void setup()
{
Serial.begin(9600);
Serial.println("Stepper test!");

}
void loop()
{
stepper.moveTo(1800);
stepper.setSpeed(150);      // steps per second = 45 rpm

stepper.run();
}
```

sketch_06 – μέγιστη ταχύτητα περιστροφής (maximum pull out rate)

Σαν μέγιστη ταχύτητα περιστροφής ορίζεται η ταχύτητα εκείνη (σε rpm) την οποία ο κινητήρας μπορεί να φτάσει έχοντας εκκινήσει εν κενώ, από στάση και χωρίς να πραγματοποιούνται εναλλαγές στην περιστροφή του, στάσεις και επανεκκινήσεις. Είναι δηλαδή το όριο πέρα από το οποίο ο κινητήρας αρχίζει να χάνει βήματα, να παρουσιάζει ανωμαλίες στην περιστροφή του και τελικά να παύει να λειτουργεί (stall).
Κατά τη διάρκεια λειτουργίας του κινητήρα, σε κάποιες συχνότητες (ταχύτητες) θα λάβουν χώρα μηχανικοί συντονισμοί. Οι συχνότητες αυτές αποκαλύπτονται κάνοντας έναν ισχυρό θόρυβο. Εδώ μπορεί να προκληθεί χάσιμο βημάτων αλλά καθόσον αυξάνεται η ταχύτητα περιστροφής ο κινητήρας επανέρχεται σε ομαλή λειτουργία έως την στιγμή που θα σταματήσει οριστικά. Για τον συγκεκριμένο κινητήρα και για διέγερση τύπου two phase on η ταχύτητα αυτή είναι περίπου 200 rpm ενώ κατά την διέγερση one phase on είναι περίπου 150 rpm.

```
#include <AFMotor.h>

AF_Stepper motor(200, 2);


void setup() {
  int i=50;
  int j=51;
  Serial.begin(9600);
  Serial.println("Stepper test!");

  for(i=50;i<j;i++){

  motor.setSpeed(i);

  motor.step(200, FORWARD, DOUBLE);
  motor.release();
  j=j+1;}

}

void loop(){}
```

sketch_07 – μέγιστη ταχύτητα εκκίνησης (start / stop rate)

Έτσι ορίζουμε την μέγιστη ταχύτητα στην οποία μπορούμε να προγραμματίσουμε τον κινητήρα ώστε να μπορεί να εκκινεί και να σταματάει χωρίς να χάνει βήματα. Είναι δηλαδή η ταχύτητα εκείνη πέρα από την οποία δεν θα έχουμε ακριβή τοποθέτηση του φορτίου του κινητήρα αφού θα έχουν χαθεί βήματα. Η διαδικασία είναι ίδια με προηγουμένως με την διαφορά ότι τώρα σταματάμε και στη συνέχεια επανεκκινούμε τον κινητήρα αυξάνοντας κάθε φορά την ταχύτητα. Για τον συγκεκριμένο κινητήρα και για διέγερση one phase on η ταχύτητα αυτή είναι 100 rpm ενώ για διέγερση two phase on είναι 170 rpm. Ο κινητήρας είναι εν κενώ.

```
#include <AFMotor.h>

AF_Stepper motor(200, 2);


void setup() {
  int i=50;
  int j=51;
  Serial.begin(9600);
  Serial.println("Stepper test!");

  for(i=50;i<j;i++){

  motor.setSpeed(i);

  motor.step(200, FORWARD, SINGLE);
  motor.release();
  delay(3000);
  j=j+1;}

}

void loop(){}
```

sketch_08 – Εισαγωγή παραμέτρων μέσω της σειριακής θύρας

```
#include <AFMotor.h>

AF_Stepper motor(200,2);
long spd;
boolean done, printed;
int state;

void setup()      //set up
{
  Serial.begin(9600);
  motor.setSpeed(100);
  motor.release();
}
void loop()
{
  if (state == 0)
  {
    if (printed == false)
    {
      Serial.println("Enter Speed");
      printed = true;
    }
    if (Serial.available())
    {
      spd = Serial.parseInt();   //input spd (speed of the motor)
      done = true;
    }
    if (done)
    {
      Serial.println(spd);   //output the value of spd
      printed = false;
      done = false;
      state++;
    }
  }
if (state == 1)
  {
    if (spd > 0)     // if the input number is higher than 0, the motor will spin clockwise
    {
      motor.setSpeed(abs(spd)); //set speed
      motor.step(1000,FORWARD,DOUBLE);
    }
    else if (spd < 0)  // if the input number is lower than 0, the motor will spin anticlockwise
    {
      motor.setSpeed(abs(spd)); //set speed
      motor.step(1000,BACKWARD,DOUBLE);
    }
    else motor.release();
    state = 0;
  }
}
```

sketch_09 – γραμμικός έλεγχος ταχύτητας (linear accelaration / deceleration)

Το πρόγραμμα αυτό δημιουργεί ένα προφίλ ταχύτητας στο οποίο η επιτάχυνση και η επιβράδυνση είναι γραμμικές συναρτήσεις του χρόνου.
Η επιτάχυνση συνεχίζεται μέχρις ότου επιτευχθεί η επιθυμητή ταχύτητα.

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 1);


void forwardstep1() {
  motor1.onestep(FORWARD, DOUBLE);
}
void backwardstep1() {
  motor1.onestep(BACKWARD, DOUBLE);
}

AccelStepper stepper1(forwardstep1, backwardstep1);
void setup()
{
   stepper1.setMaxSpeed(3000.0);
   stepper1.setAcceleration(300.0);
   stepper1.moveTo(3000);

}
void loop()
{

   stepper1.run();
}
```

sketch_10 – έλεγχος της κατεύθυνσης περιστροφής από το πληκτρολόγιο

Εδώ χρησιμοποιούμε ένα πλήκτρο του υπολογιστή μας για να αλλάξουμε την φορά περιστροφής. Πατώντας το πλήκτρο μια φορά ο κινητήρας στρέφεται προς μια κατεύθυνση. Ξαναπατώντας το ίδιο πλήκτρο άλλη μια φορά ο κινητήρας στρέφεται στην αντίθετη κατεύθυνση

```
#include <AFMotor.h>
AF_Stepper motor1(200,2);
byte dir;

int count;
void setup(){
  Serial.begin(9600);
  Serial.println("motor_set.");
}
void loop(){

  while (Serial.available()){

      dir = Serial.read();

      Serial.println(dir);

    if(dir == 90){

      switch(count){

      case 0:

        motor1.setSpeed(100);
        motor1.step(2000,FORWARD,DOUBLE);
        count=1;
        motor1.setSpeed(0);
        motor1.release();
        break;

       case 1:

        motor1.setSpeed(100);
        motor1.step(2000,BACKWARD,DOUBLE);
        count=0;
        motor1.setSpeed(0);
        motor1.release();
        break;
      }

    }
  }}
```

sketch_11 – πέρασμα παραμέτρων για ταχύτητα, επιτάχυνση και απόσταση από το πληκτρολόγιο

Εδώ επειδή το πρόγραμμα αυτό στηρίζεται σε μια state machine, για να επικοινωνήσουμε μέσω της σειριακής οθόνης του interface χρειάζεται να περάσουμε τις τιμές για ταχύτητα, επιτάχυνση και απόσταση, όλες μαζί ως εξής: **S**xxx**A**xxx**D**xxx**S**. Το τελευταίο S μπαίνει για να επιστρέψουμε στην αρχική / τελική κατάσταση.

Παράδειγμα:

S450**A**300**D**800**S**

για (μέγιστη)ταχύτητα 450 [steps / sec], επιτάχυνση (= επιβράδυνση) 300[steps/sec/sec] και απόσταση 800 [steps].

```
#include <AccelStepper.h>
#include <AFMotor.h>

AF_Stepper motor1(200, 2);
void forwardstep1() {
  motor1.onestep(FORWARD, DOUBLE);
}
void backwardstep1() {
  motor1.onestep(BACKWARD, DOUBLE);
}

AccelStepper stepper1(forwardstep1, backwardstep1);
// the possible states of the state-machine
typedef enum {
  NONE, GOT_S, GOT_A, GOT_D }
states;

// current state-machine state
states state = NONE;
// current partial number
unsigned int currentValue;

void setup()
{
  Serial.begin(9600);
  state=NONE;
}



float processSPEED (const unsigned int value)
{
  // do something with speed
  stepper1.setMaxSpeed(value);
  Serial.print ("MAXSPEED = ");
  Serial.println (value);

} // end of processRPM
```

```
void processACCEL (const unsigned int value)
{
  // do something with acel
  stepper1.setAcceleration(value);
  Serial.print ("ACCEL = ");
  Serial.println (value);
} // end of processSpeed

void processDIST (const unsigned int value)
{
  // do something with distance
  stepper1.moveTo(value);
  Serial.print ("DIST = ");
  Serial.println (value);
} // end of processGear

void handlePreviousState ()
{
  switch (state)
  {
  case GOT_S:
    processSPEED (currentValue);
    break;
  case GOT_A:
    processACCEL (currentValue);
    break;
  case GOT_D:
    processDIST (currentValue);
    break;
  }  // end of switch

  currentValue = 0;
}  // end of handlePreviousState


void processIncomingByte (const byte c)
{
  if (isdigit (c))
  {
    currentValue *= 10;
    currentValue += c - '0';
  }  // end of digit
  else
  {

    // The end of the number signals a state change
    handlePreviousState ();

    // set the new state, if we recognize it
    switch (c)
    {
    case 'S':
      state = GOT_S;
      break;
    case 'A':
```

```
      state = GOT_A;
      break;
    case 'D':
      state = GOT_D;
      break;
    default:
      state = NONE;
      break;
    }  // end of switch on incoming byte
  } // end of not digit

} // end of processIncomingByte



void loop()
{

  if (Serial.available ())
    processIncomingByte (Serial.read ());


  stepper1.run();
}
```

Βιβλιογραφία

Acarnley P.P : STEPPING MOTORS: a guide to modern theory and practice

Atmel Corporation: AVR446 – Linear speed control of stepper motor

Austin David: Generate stepper motor speed profiles in real time

Kenjo Takashi and Sugawara Akira: Stepping motors & their microprocessor controls

Koehrsen Michael: Understanding the Adafruit Motor Shield Library

Pisi Daniel: Generating stepper motor speed profiles using FPGA

Quinones I. Jose: Applying acceleration and deceleration profiles to bipolar stepper motors

https://learn.adafruit.com/adafruit-motor-shield/overview

http://www.airspayce.com/mikem/arduino/AccelStepper/index.html

http://arduino.cc/

Αφιερωμένο

στηνΠ
στηνΚ
jeepwranglersavedmylife