ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ
Πανεπιστημιούπολη Σίνδου

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΡΟΜΠΟΤΙΚΗ, STEAM ΚΑΙ ΝΕΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΣΤΗΝ ΕΚΠΑΙΔΕΥΣΗ»

Διπλωματική Εργασία

ΕΛΕΓΧΟΣ ΣΤΡΟΦΩΝ DC ΚΙΝΗΤΗΡΑ ΜΕ ARDUINO

του

ΔΗΜΗΤΡΙΟΥ ΠΕΧΛΗΒΑΝΗ

Επιβλέπων Καθηγητής

ΧΡΗΣΤΟΣ ΥΦΟΥΛΗΣ

Αν. καθηγητής Τμ. Μηχανικών Παραγωγής & Διοίκησης, ΔΙΠΑΕ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος ειδίκευσης Ρομποτική, STEAM και Νέες Τεχνολογίες στην Εκπαίδευση

Θεσσαλονίκη, Μήνας Έτος

Υπεύθυνη Δήλωση Φοιτητή

Η παρούσα Διπλωματική Εργασία και τα συμπεράσματά της, σε οποιαδήποτε μορφή,αποτελούν συνιδιοκτησία του Τμήματος Μηχανικών Παραγωγής και Διοίκησης του Διεθνούς Πανεπιστημίου Ελλάδος και της φοιτητή. Οι προαναφερόμενοι διατηρούν το δικαίωμα ανεξάρτητης χρήσης και αναπαραγωγής (τμηματικά ή συνολικά) για διδακτικούς και ερευνητικούς σκοπούς. Σε κάθε περίπτωση πρέπει να
αναφέρεται ο τίτλος, η συγγραφέας, ο επιβλέπων και το τμήμα του ΔιΠαΕ. Η έγκριση της παρούσας Διπλωματικής Εργασίας από το Τμήμα Μηχανικών Παραγωγής και Διοίκησης δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Δηλώνω υπεύθυνα ότι η παρούσα Διπλωματική Εργασία είναι εξ ολοκλήρου δικό μου έργο και συγγράφηκε ειδικά για τις απαιτήσεις του μεταπτυχιακού προγράμματος σπουδών του Τμήματος Μηχανικών Παραγωγής και Διοίκησης με τίτλο «Ρομποτική, STEAM και νέες Τεχνολογίες στην Εκπαίδευση».

Κατά τη συγγραφή ακολούθησα την πρέπουσα ακαδημαϊκή δεοντολογία αποφυγής λογοκλοπής και έχω αποφύγει οποιαδήποτε ενέργεια που συνιστά παράπτωμα λογοκλοπής. Δεν χρησιμοποίησα ολόκληρο ή μέρος έργου άλλου δημιουργού ή τις ιδέες και αντιλήψεις άλλου δημιουργού χωρίς να γίνεται σαφής αναφορά στην πηγή προέλευσης (βιβλίο, άρθρο από επιστημονικό περιοδικό, ιστοσελίδα κλπ.).

Θεσσαλονίκη, 5 Φεβρουαρίου 2024
Ο Δηλών: Δημήτρης Πεχληβάνης

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία αφορά την πειραματική μελέτη και την αξιολόγηση του προβλήματος ελέγχου στροφών και γωνιακής θέσης ενός μικρού DC κινητήρα. Για την εκτέλεση των απαραίτητων πειραμάτων χρησιμοποιήθηκε ένας μικρός DC σερβοκινητήρας με ενσωματωμένο κωδικοποιητή, έτσι ώστε να είναι δυνατός ο ακριβής έλεγχος ταχύτητας και θέσης με βελτιστοποίηση της απόκρισης και της ευστάθειας του συστήματος μέσω της ρύθμισης των παραμέτρων ελέγχου, δηλ. των τριών κερδών ενός PID ελεγκτή τριών όρων, Αναλογικού, Ολοκληρωτικού και Διαφορικού. Ο έλεγχος επιτυγχάνεται με τον μικροελεγκτή Arduino μέσω ενός βασικού κυκλώματος ελέγχου κινητήρων. Συνολικά, παρέχουμε λεπτομερείς πληροφορίες σχετικά με τη μεθοδολογία, την πειραματική διαδικασία, την λήψη και την αξιολόγηση των αποτελεσμάτων με στόχο να δώσουμε στον αναγνώστη που δεν κατέχει εξειδικευμένες γνώσεις τη δυνατότητα να κατανοήσει και να εφαρμόσει τα ευρήματα σε παρόμοια συστήματα ή εφαρμογές.

ABSTRACT

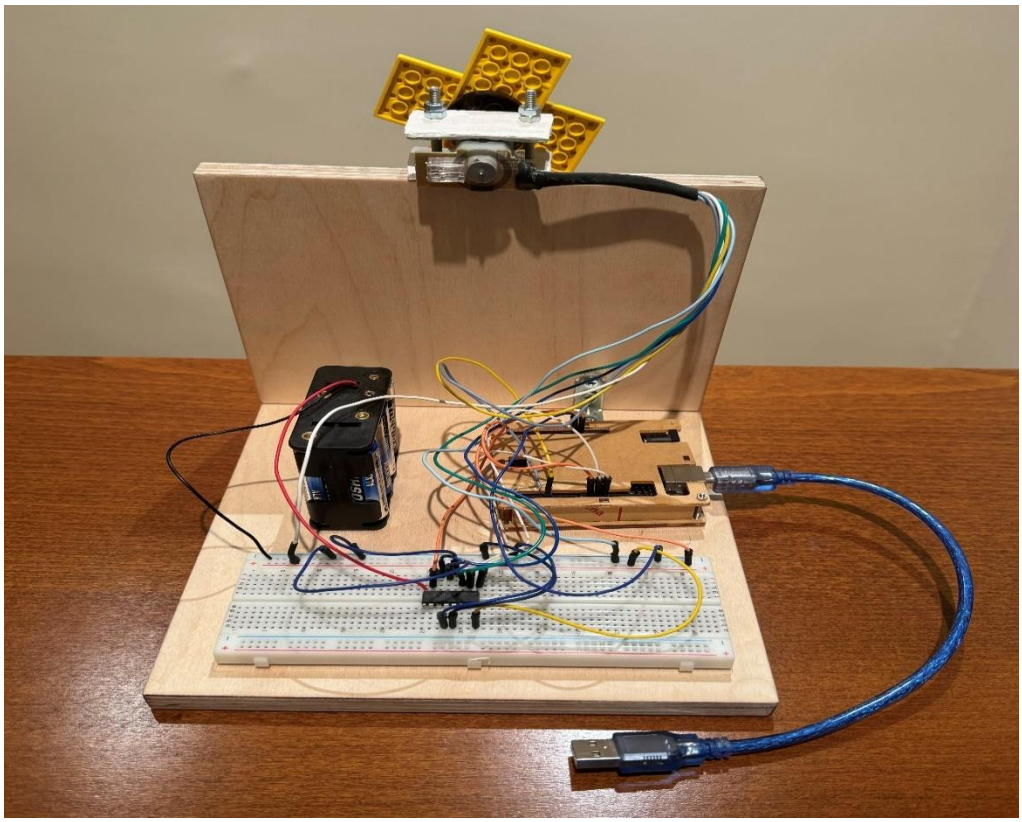
This thesis concerns the experimental study and evaluation of the speed and angular position control problem of a small DC motor. To perform the necessary experiments, a construction with cheap materials and a small DC servo motor with a built-in encoder were used, so that precise control of speed and position was possible by optimizing the response and stability of the system by adjusting the control parameters, i.e. of the gains of a three term PID controller, Analog, Integral and Differential. Control is achieved with the Arduino microcontroller via a basic motor control circuit. Overall, we provide detailed information on the methodology, experimental procedure, acquisition and evaluation of results with the aim of enabling the lay reader to understand and apply the findings to similar systems or applications.

Περιεχόμενα

ΠΕΡΙΛΗΨΗ 5
ABSTRACT 6
ΚΑΤΑΣΚΕΥΗ ΔΙΑΤΑΞΗΣ 8
Περιγραφή κατασκευής 8
Καταγραφή υλικών 9
Περιγραφή κυκλώματος 10
ΕΛΕΓΧΟΣ ΤΑΧΥΤΗΤΑΣ DC ΚΙΝΗΤΗΡΑ 12
Καταγραφή Βηματικής απόκρισης ταχύτητας 12
Υπολογισμός σταθεράς χρόνου και ρύθμιση PI ελεγκτή για έλεγχο ταχύτητας 13
Πείραμα σε κλειστό βρόχο με προσθήκη του ελεγκτή 14
Πειραματικά αποτελέσματα με μεγάλο φορτίο 15
Πειραματικά αποτελέσματα με μικρό φορτίο 20
Γενικά συμπεράσματα: 25
ΕΛΕΓΧΟΣ ΓΩΝΙΑΚΗΣ ΘΕΣΗΣ DC ΚΙΝΗΤΗΡΑ 26
Υπολογισμός και ρύθμιση PID ελεγκτή για έλεγχο θέσης γωνίας 26
Πείραμα ελέγχου θέσης γωνίας 26
Πειραματικοί υπολογισμοί Kp, Ki και Kd για τις τιμές του Tc 27
Πειραματικά αποτελέσματα με μεγάλο φορτίο 28
Πειραματικά αποτελέσματα με μικρό φορτίο 32
Γενικά συμπεράσματα: 36
ΒΙΒΛΙΟΓΡΑΦΙΑ 37
ΠΑΡΑΡΤΗΜΑΤΑ 38
ΠΑΡΑΡΤΗΜΑ Α 38
ΠΑΡΑΡΤΗΜΑ Β 40
ΠΑΡΑΡΤΗΜΑ Γ 42

ΚΑΤΑΣΚΕΥΗ ΔΙΑΤΑΞΗΣ

Περιγραφή κατασκευής

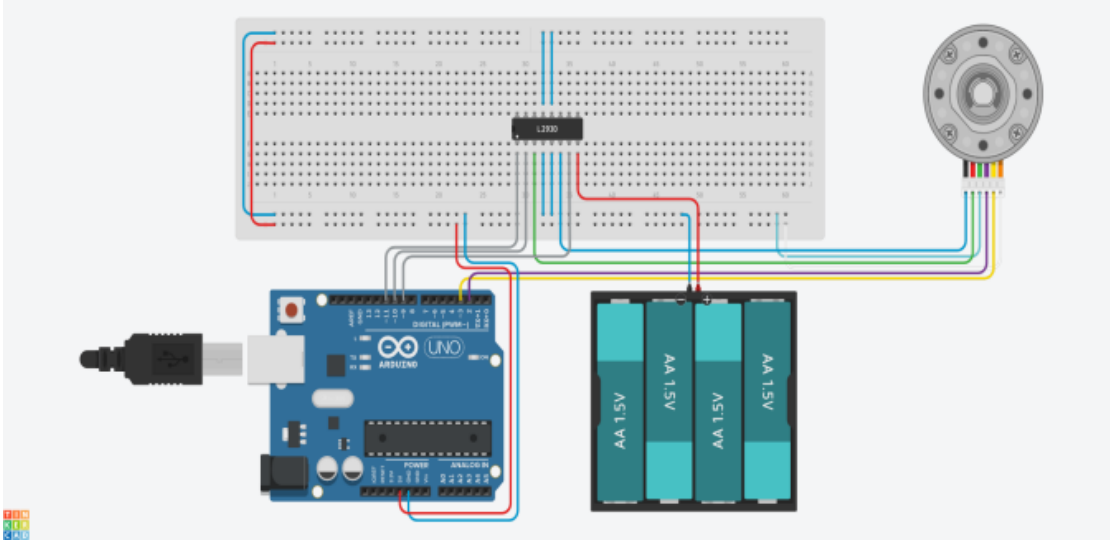


Η κατασκευή αποτελείται από δύο ξύλινές βάσεις που στηρίζουν όλα τα υλικά πάνω τους . Στην πρώτη βάση βρίσκονται σταθεροποιημένα το ράστερ με το κύκλωμα, οι μπαταρίες και ο Arduino UNO R3. Ενώ στην δεύτερη βάση σταθεροποιείται ο dc κινητήρας με τον κωδικοποιητή 2 καναλιών.

Καταγραφή υλικών

Τα υλικά που χρησιμοποιήθηκαν για την δημιουργία του κυκλώματος καταγράφονται στον παρακάτω πίνακα:

Υλικά	Τεμάχια
DC κινητήρας με κωδικοποιητή 2 καναλιών	1
Ράστερ	1
Μπαταρία 1.5V	8
Driver L293D	1
Arduino UNO R3	1
Male to male καλώδιο ράστερ	19



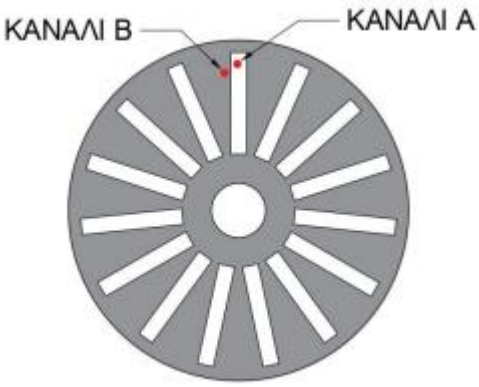
Εικόνα 1Κύκλωμα ελέγχου κινητήρα.

Περιγραφή κυκλώματος

Για να ελέγξουμε τις στροφές ενός κινητήρα, εφαρμόσαμε έναν κινητήρα DC που διαθέτει έναν ενσωματωμένο οπτικό κωδικοποιητή 2 καναλιών. Αυτός ο κωδικοποιητής παρέχει πληροφορίες σχετικά με τη θέση και τη φορά περιστροφής του κινητήρα.

Κανάλια κωδικοποιητή

Ο κωδικοποιητής αποτελείται από δύο κανάλια που βρίσκονται σε μικρή απόσταση μεταξύ τους. Κάθε κανάλι περιλαμβάνει ένα LED και ένα φωτοτρανζίστορ, μεταξύ των οποίων υπάρχει ένας δίσκος που είναι συνδεδεμένος με τον ίδιο τον κινητήρα. Ο δίσκος αυτός έχει κομμένα τμήματα που επιτρέπουν στο φως του LED να



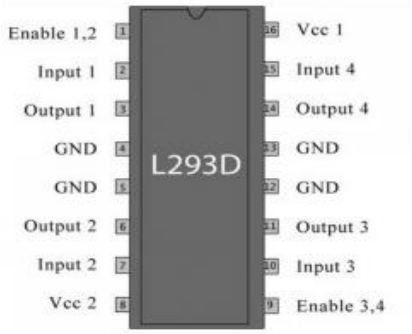
διακόπτεται καθώς ο δίσκος περιστρέφεται. Κάθε φορά που το φωτοτρανζίστορ ανιχνεύει φως, δημιουργείται ένας παλμός.

Ανατροφοδότηση θέσης και φοράς περιστροφής κινητήρα

Μέσω των δύο καναλιών του κωδικοποιητή, μπορούμε να καθορίσουμε τη φορά περιστροφής του κινητήρα. Επιπλέον, μετρώντας τους παλμούς, μπορούμε να παρακολουθούμε τη θέση του κινητήρα σε κάθε χρονική στιγμή. Για να αυξήσουμε την ακρίβεια αυτής της μέτρησης, χρησιμοποιούμε διακοπές (interrupts) για να μετρήσουμε τις αλλαγές στους παλμούς αντί για συγκεκριμένες παρυφές. Με αυτόν τον τρόπο, διπλασιάζουμε την ανάλυση κάθε καναλιού ανά περιστροφή, προσφέροντας έτσι τετραπλάσια ακρίβεια στη συνολική μέτρηση.

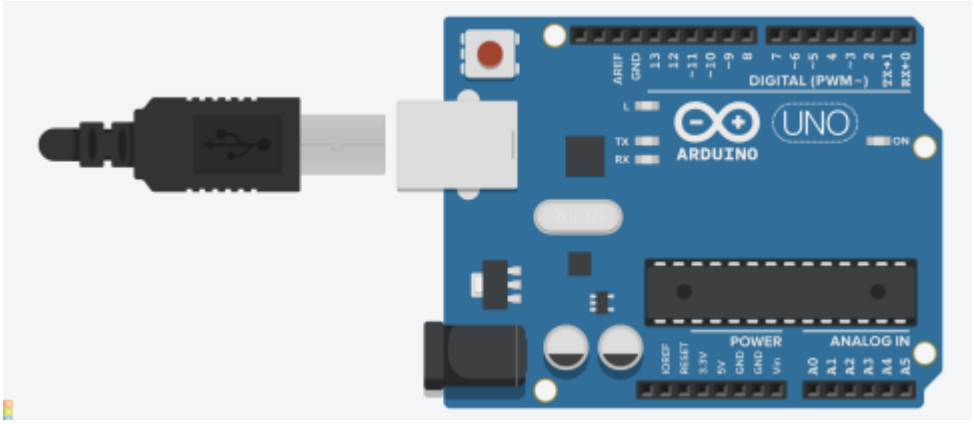
Driver L293D

Για τον έλεγχο της κατεύθυνσης και της τάσης τροφοδοσίας του κινητήρα, χρησιμοποιούμε το ολοκληρωμένο L293D (H-bridge) και μια εξωτερική τροφοδοσία 12V. Αυτό μας επιτρέπει να παρέχουμε υψηλότερη τάση από τα 5V του Arduino στον κινητήρα και να ελέγχουμε τη φορά περιστροφής αλλάζοντας την πολικότητα.



Arduino UNO R3

Για τον έλεγχο και την καταγραφή του κώδικα, χρησιμοποιούμε την πλακέτα Arduino UNO R3. Αυτή η πλακέτα διαθέτει δύο θύρες (pin 2 και 3) για χρήση διακοπών, οι οποίες χρησιμοποιούνται για την καταμέτρηση των παλμών των καναλιών του κωδικοποιητή. Επιπλέον, χρησιμοποιούμε δύο άλλες θύρες ως ψηφιακές εξόδους (pin 9 και 10) για την αλλαγή της πολικότητας και μια άλλη θύρα ως αναλογική έξοδος (pin 11) για τον έλεγχο της τάσης τροφοδοσίας με χρήση του PWM.



ΕΛΕΓΧΟΣ ΤΑΧΥΤΗΤΑΣ DC ΚΙΝΗΤΗΡΑ

Σε αυτή την ενότητα θα εξηγήσουμε λεπτομερώς την διαδικασία σχεδίασης και πειραματικής επιβεβαίωσης ενός PID ελεγκτή για τα προβλήματα ελέγχου ταχύτητας και θέσης ενός DC κινητήρα, που είναι απαραίτητη σε διάφορες εφαρμογές. Η προτεινόμενη πειραματική διαδικασία μπορεί να εφαρμοστεί στην πράξη και να δώσει καλά αποτελέσματα χωρίς να απαιτούνται εξειδικευμένες γνώσεις ή κατανόηση της θεωρίας των ελεγκτών PID ή πολύπλοκοι μαθηματικοί υπολογισμοί. Προτείνεται μια ημιεμπειρική μέθοδος που επιτρέπει με ένα απλό πείραμα σε ανοικτό βρόχο (απουσία ελεγκτή) να ρυθμίσουμε τα κέρδη του PID ελεγκτή προσαρμόζοντας τα στο εκάστοτε φορτίο του κινητήρα μας.

Καταγραφή Βηματικής απόκρισης ταχύτητας

Εφόσον έχει ολοκληρωθεί η κατασκευή μας, επιλέγουμε το επιθυμητό φορτίο και το τοποθετούμε στο DC κινητήρα μας. Στη συνέχεια ανοίγουμε το πρόγραμμα του Arduino. Πατάμε «Αρχείο» , «Άνοιγμα» και επιλέγουμε το πρόγραμμα «DC speed open.ino» που παρατίθεται στο Παράρτημα Α, και κάνουμε «ανέβασμα». Το πρόγραμμα αυτό μας δίνει την δυνατότητα να κάνουμε ένα πείραμα καταγραφής της βηματικής απόκρισης του κινητήρα μας, δηλ. να καταγράψουμε το μεταβατικό φαινόμενο που λαμβάνει χώρα όταν παρέχουμε απότομα την μέγιστη διαθέσιμη ισχύ στον κινητήρα μας. και να πάρουμε τις μέγιστες τιμές ταχύτητας του φορτίου στην μέγιστη ισχύ που θα του δώσουμε.

Στην συνέχεια πηγαίνουμε στα «Εργαλεία», «Σχεδιογράφος σειριακής» και παρατηρούμε τα γραφήματα. Εφόσον πάρουμε μια εικόνα για το πως πρέπει να είναι το τελικό γράφημα μας το κλείνουμε και επιλέγουμε στα «Εργαλεία» το «Παρακολούθηση σειριακής». Κάνουμε αντιγραφή τα αποτελέσματα και επικόλληση σε ένα σημειωματάριο. Έπειτα ανοίγουμε ένα Excel πατάμε «Αρχείο» , «Άνοιγμα» και επιλέγουμε το σημειωματάριο που έχουμε αποθηκευμένα τα αποτελέσματα από την προηγούμενη διαδικασία. Τέλος επιλέγουμε τις στήλες με τα αποτελέσματα που μας ενδιαφέρουν και εμφανίζουμε το τελικό μας γράφημα.

Υπολογισμός σταθεράς χρόνου και ρύθμιση PI ελεγκτή για έλεγχο ταχύτητας

Με την ολοκλήρωση της παραπάνω διαδικασίας θα μπορέσουμε να υπολογίσουμε το κέρδος μας **K** που προκύπτει από την παρακάτω εξίσωση $K = \frac{\Delta Y}{\Delta U}$
Όπου ΔΥ είναι η μέγιστη ταχύτητα και ΔU η μεγίστη ισχύς (255 σε PWM)

Στην συνέχεια για να υπολογίσουμε την σταθερά του χρόνου μας **T** πρέπει να λάβουμε γνώση ότι η «ταχύτητα» της απόκρισης, είναι ο χρόνος που απαιτείται για να φτάσει η έξοδος στο **63%** της τελικής της τιμής. Συνεπώς
ΔΥ * 0,63
Από το αποτέλεσμα αυτής της εξίσωσης θα μετρήσουμε πόσα δείγματα (**e**) χρειάστηκαν ώστε να φτάσουμε στο **63%** της ταχύτητας απόκρισης. Οπότε προκύπτει
T= e * 0,02
όπου 0,02 είναι τα δευτερόλεπτα ανά δείγμα (20 msec) που έχουμε ορίσει από το πρόγραμμα «DC speed open.ino» , δηλ. η περίοδος δειγματοληψίας του συστήματος.

Εφόσον ολοκληρώσουμε αυτή την διαδικασία και βρούμε τα **K** και **T** στο φορτίο που επιλέξαμε, μπορούμε να συνεχίσουμε με την ρύθμιση του PID ελεγκτή. Πιο συγκεκριμένα, στον έλεγχο ταχύτητας προτείνεται η χρήση PI ελεγκτή, δηλ. ο διαφορικός όρος D παραλείπεται. Για τον υπολογισμό των δύο παραμέτρων του **PID** ελεγκτή **Kp** και **Ki** εφαρμόστηκε με επιτυχία η μέθοδος IMC SKOGESTAD [1] όπου:

$$K_p = \frac{1}{K} \times \frac{T}{T_c}$$

$$K_i = \frac{Kp}{\min(T, 4T_c)}$$

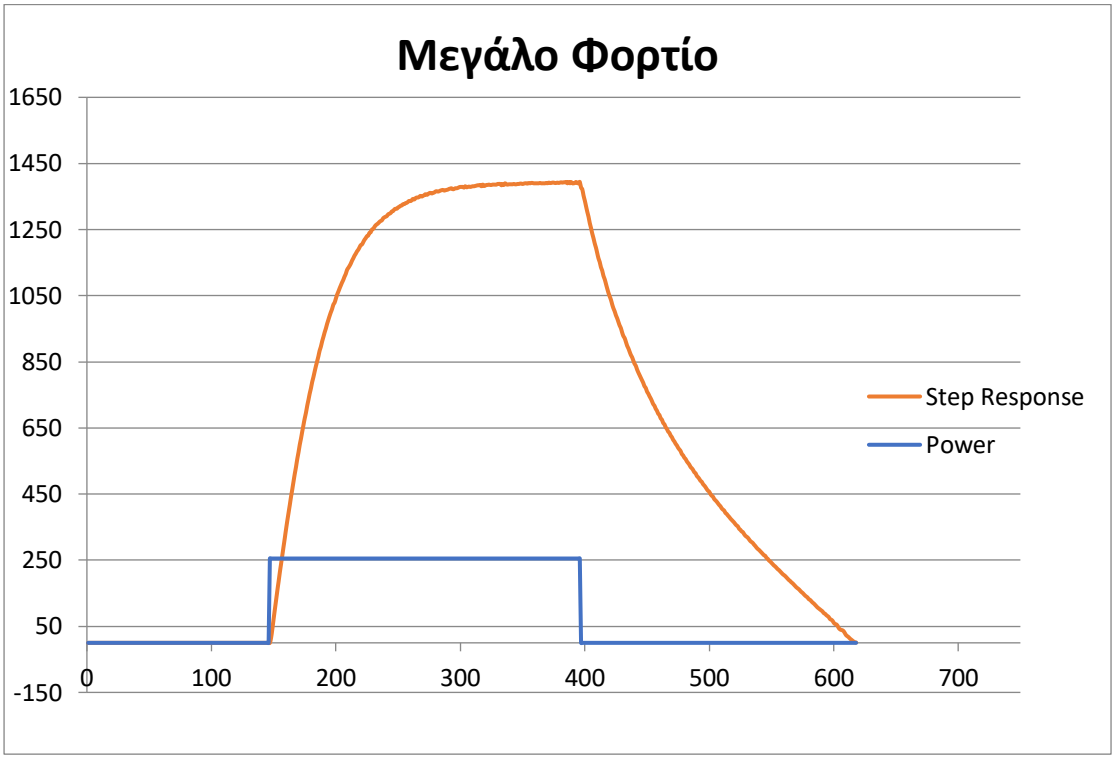
Ανάλογα με την τιμή που θα δώσουμε στο **Tc** θα προκύψουν οι τιμές των **Kp** και **Ki** , όπου το Tc (σταθερά χρόνου κλειστού βρόχου) είναι μια τιμή που επιλέγουμε με βάση τις απαιτήσεις μας από την προσθήκη του ελεγκτή. Η σταθερά χρόνου είναι ένα μέτρο της ταχύτητας αποκρισης ενός συστήματος. Όσο μικρότερη είναι τόσο γρηγορότερα αποκρίνεται το σύστημα στις εντολές που του δίνουμε. Επιλέγοντας

Tc=T αναφερόμαστε σε ελεγκτή που δεν θα αλλάξει την ταχύτητα απόκρισης του συστήματος, ενώ αν π.χ. επιλέξουμε Tc=T/2 δίνουμε το μήνυμα ότι επιθυμούμε υποδιπλασιασμό της σταθεράς χρόνου του συστήματος με την προσθήκη του ελεγκτή, άρα διπλασιασμό της ταχύτητας απόκρισης του.

Πείραμα σε κλειστό βρόχο με προσθήκη του ελεγκτή

Έπειτα θα ανοίξουμε το πρόγραμμα του Arduino. Πατάμε «Αρχείο», «Άνοιγμα» και αυτή την φορά θα επιλέξουμε το πρόγραμμα «DC speed closed.ino» που παρατίθεται στο ΠΑΡΑΡΤΗΜΑ Β και να προσαρμόσουμε τις τιμές του **Kp** και **Ki** με βάση τους υπολογισμούς που βρήκαμε, να κάνουμε «ανέβασμα» και να καταγράψουμε τα καινούργια αποτελέσματα. Στη συνέχεια περιγράφουμε την πειραματική διαδικασία και τα αποτελέσματα σε 2 διαφορετικές περιπτώσεις φορτίων που δοκιμάστηκαν, ένα μεγάλο (βαρύτερο) και ένα μικρότερο (ελαφρύτερο).

Πειραματικά αποτελέσματα με μεγάλο φορτίο



Σε αυτή την εικόνα βλέπουμε τα αποτελέσματα από το πρόγραμμα του DC speed open, που στην ουσία είναι οι μέγιστες τιμές ταχύτητας που μπορούμε να πάρουμε για το μεγάλο μας φορτίο. Με αυτό το γράφημα θα μπορέσουμε να υπολογίσουμε το στατικό κέρδος (**K**) και την σταθερά χρόνου (**T**).

Υπολογισμός στατικού κέρδους (**K**) :

$$K = \frac{\Delta Y}{\Delta U} = \frac{1400}{255} = 5{,}5$$

Υπολογισμός Σταθεράς χρόνου (**T**) :

Η «ταχύτητα» της απόκρισης. Είναι ο χρόνος που απαιτείται για να φτάσει η έξοδος στο **63%** της τελικής της τιμής.

$$1400 \times 0{,}63 = 882$$

που ισοδυναμεί με **38** δείγματα

$$T = 38 \times 0{,}02 = 0{,}76\ sec$$

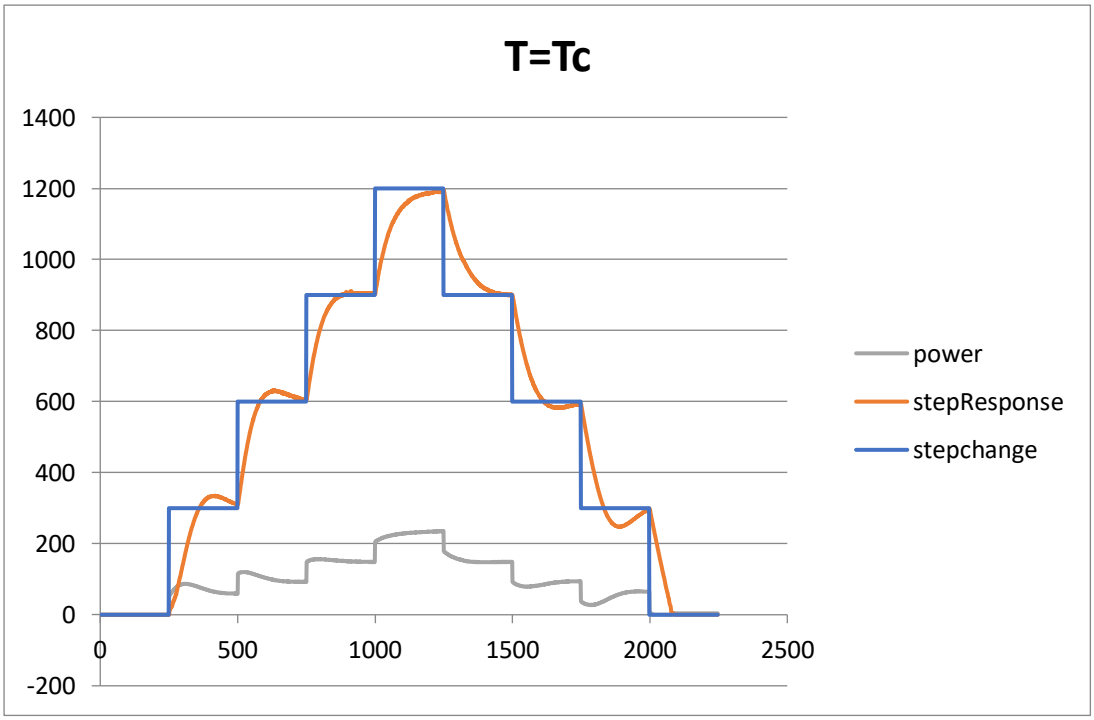
Υπολογισμός παραμέτρων PI ελεγκτή (**Kp,Ki**) :

$$K_p = \frac{1}{K} \times \frac{T}{T_c}$$

$$K_i = \frac{K_p}{\min(T, 4T_c)}$$

Για **Tc=T**

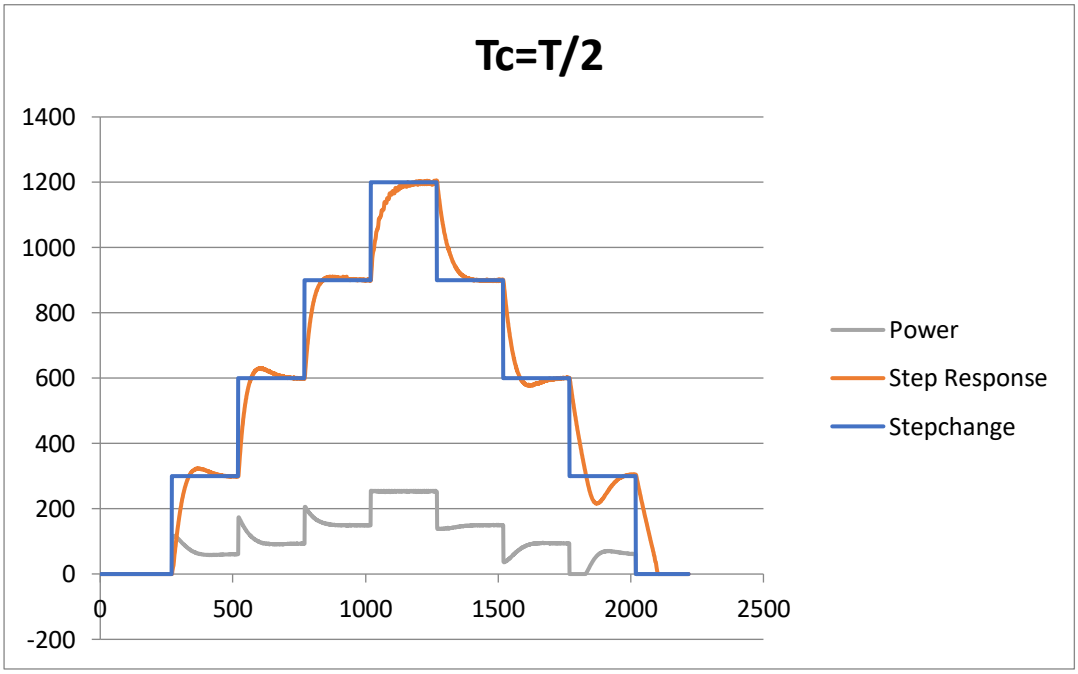
Kp_1=0,18 Ki_1=0,24



Αργή βηματική απόκριση. Υπάρχει καθυστέρηση στην αντίδραση του συστήματος

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{2}}$

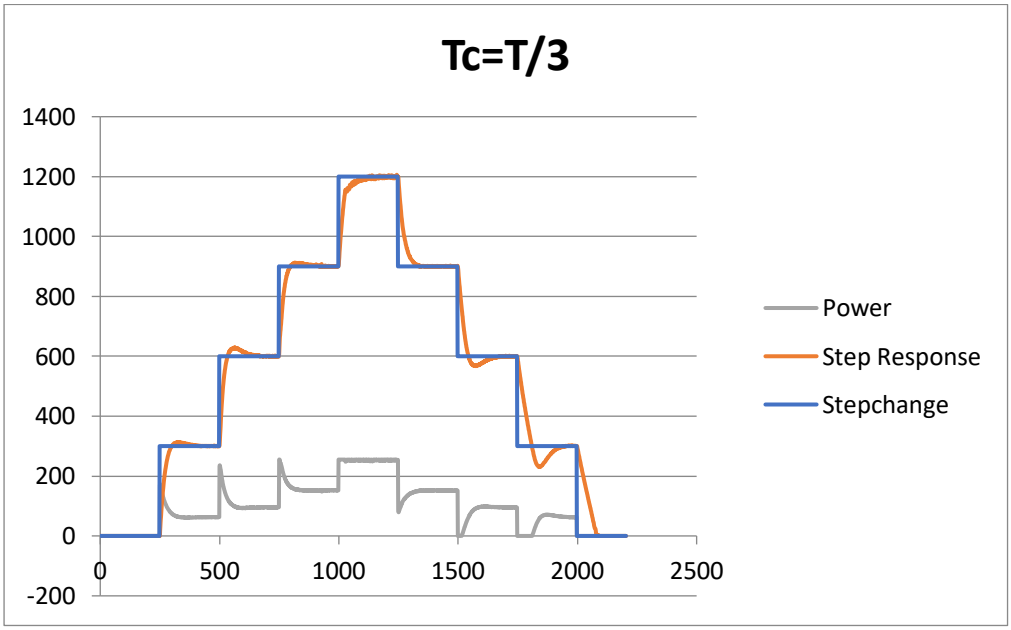
Kp_2=0,36 Ki_2=0,48



Ελαφρώς πιο βελτιωμένη βηματική απόκριση σε σχέση με Tc=T.

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{3}}$

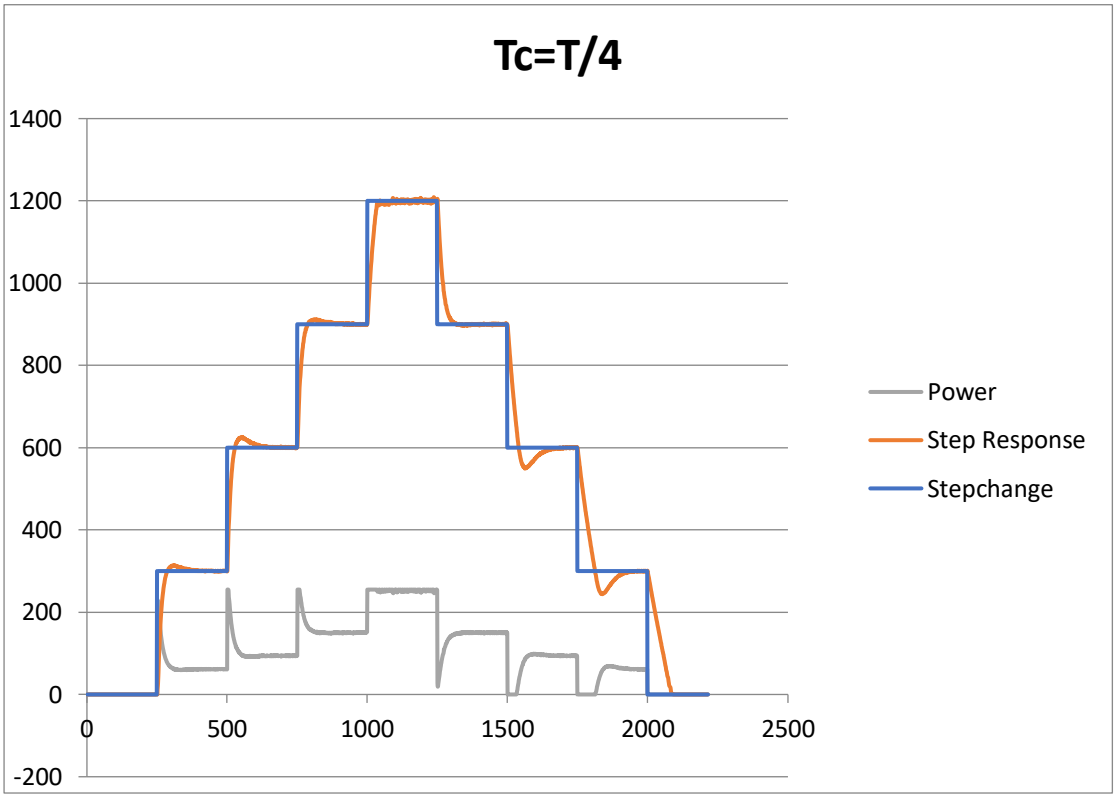
Kp_3=0,55 Ki_3=0,72



Ακόμα πιο βελτιωμένη απόκριση. Η μικρότερη τιμή του T επιτρέπει στον ελεγκτή να αντιλαμβάνεται και να ανταποκρίνεται πιο γρήγορα.

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{4}}$

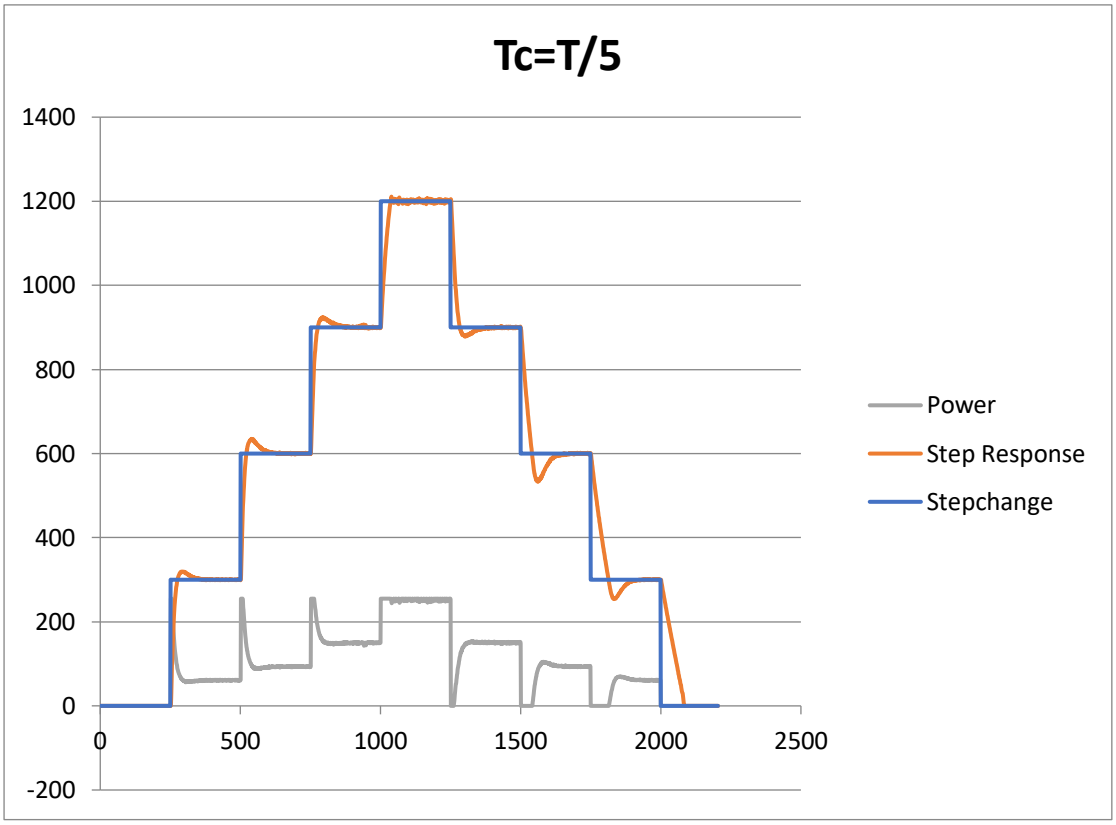
Kp_4=0,73 Ki_4=0,96



Η καλύτερη απόκριση σε σχέση με όλες τις προηγούμενες περιπτώσεις. Το σύστημα μπορεί να αποκριθεί γρήγορα σε όλες τις βηματικές αλλαγές.

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{5}}$

Kp_5=0,9 Ki_5=1,5



Επίσης πολύ καλή απόκριση ανάλογη με την προηγούμενη περίπτωση.

Σε αυτό το πείραμα παρατηρούμε ότι:

1. Όταν η τιμή του Tc γίνεται μικρότερη, οι τιμές του Kp και Ki μεγαλώνουν.
2. Όσο μικραίνει η τιμή Tc και μεγαλώνουν οι τιμές του Kp και Ki βλέπουμε στα γραφήματα μας να υπάρχει γρηγορότερη βηματική απόκριση

Πειραματικά αποτελέσματα με μικρό φορτίο

Επαναλαμβάνουμε το ίδιο πείραμα και με μικρότερο φορτίο στο πρόγραμμα «DC speed open». Όπως είναι λογικό μικρότερο φορτίο ισοδυναμεί με μεγαλύτερες τιμές στην ταχύτητα που αυτό αντιστοιχεί σε μεγαλύτερο κέρδος K και μικρότερο χρόνο T σε σχέση με το μεγαλύτερο φορτίο.

Υπολογισμός στατικού κέρδους (**K**) :

$$K = \frac{\Delta Y}{\Delta U} = \frac{1720}{255} = 6{,}75$$

Υπολογισμός Σταθεράς χρόνου (**T**) :

Η «ταχύτητα» της απόκρισης. Είναι ο χρόνος που απαιτείται για να φτάσει η έξοδος στο **63%** της τελικής της τιμής.

$$1720 \times 0{,}63 = 1083$$

που ισοδυναμεί με **22** δείγματα

$$T = 22 \times 0{,}02 = 0{,}44 \text{ sec}$$

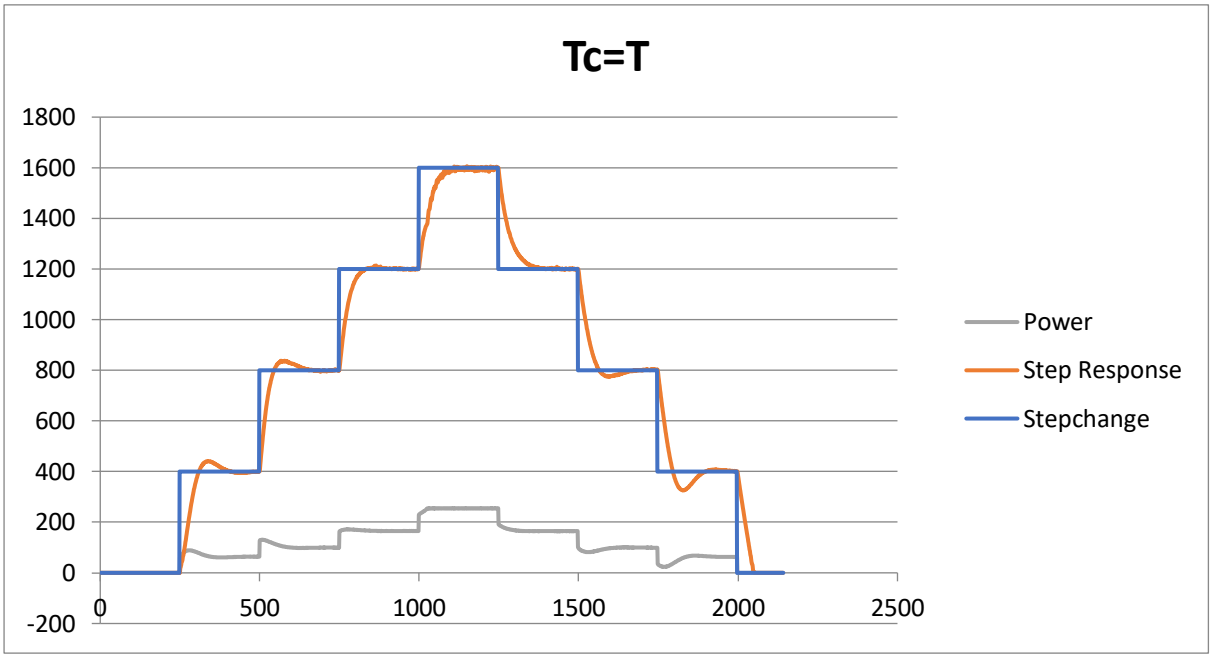
Υπολογισμός παραμέτρων PI ελεγκτή (**Kp,Ki**) :

$$K_p = \frac{1}{K} \times \frac{T}{T_c}$$

$$K_i = \frac{Kp}{\min(T, 4T_c)}$$

Για **Tc=T**

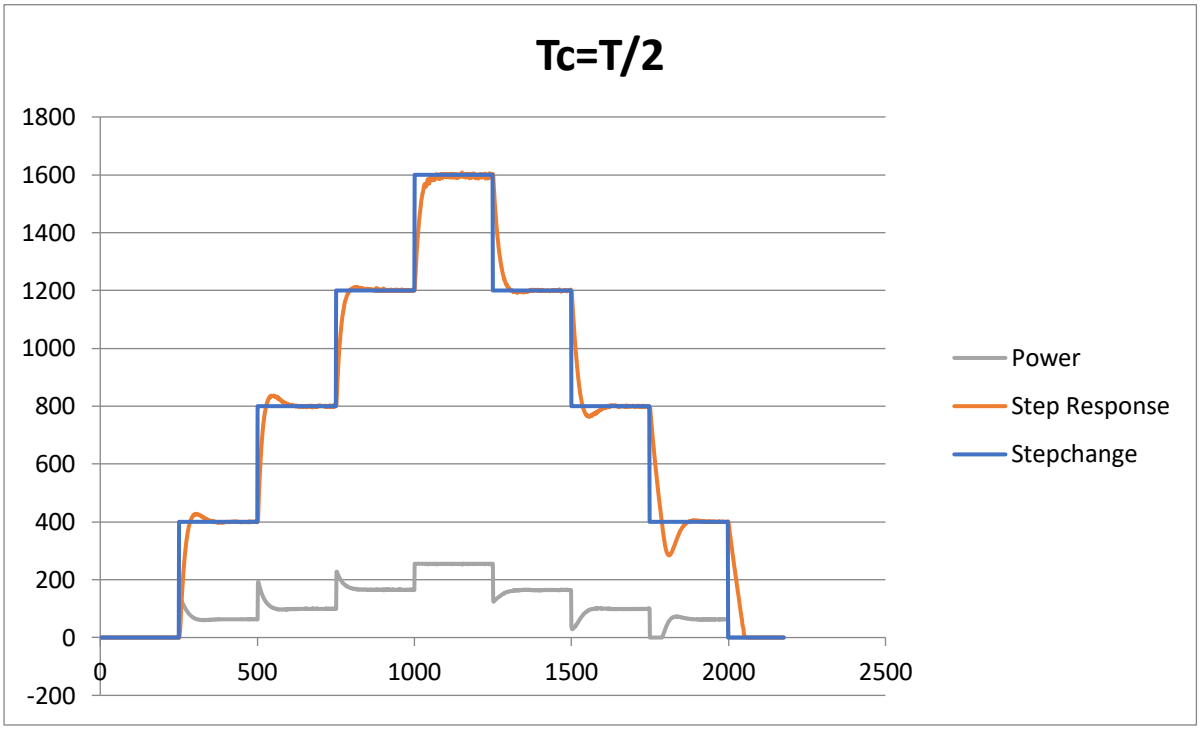
Kp_1=0,15 Ki_1=0,34



Βελτίωση σε σχέση με την περίπτωση με μεγάλο φορτίο. Ακόμα όμως υπάρχει αργή απόκριση

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{2}}$

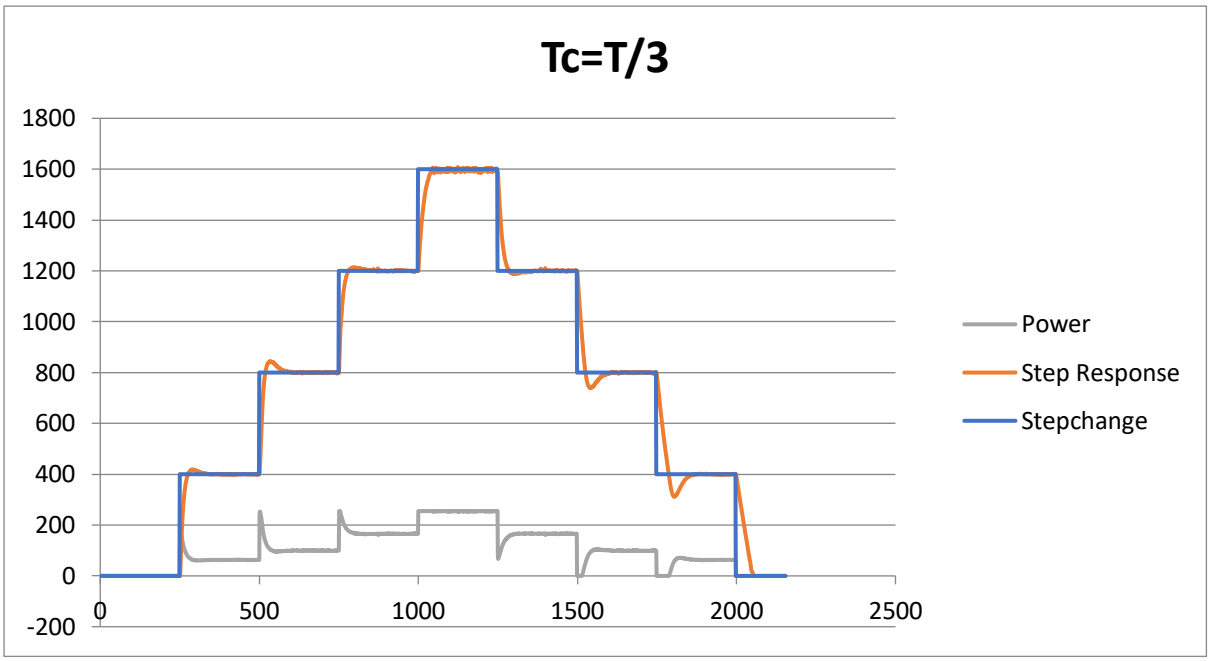
Kp_2=0,3 Ki_2=0,67



Πολύ καλύτερη απόκριση σε σχέση με την προηγούμενη περίπτωση. Εμφανώς πιο γρήγορη απόκριση.

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{3}}$

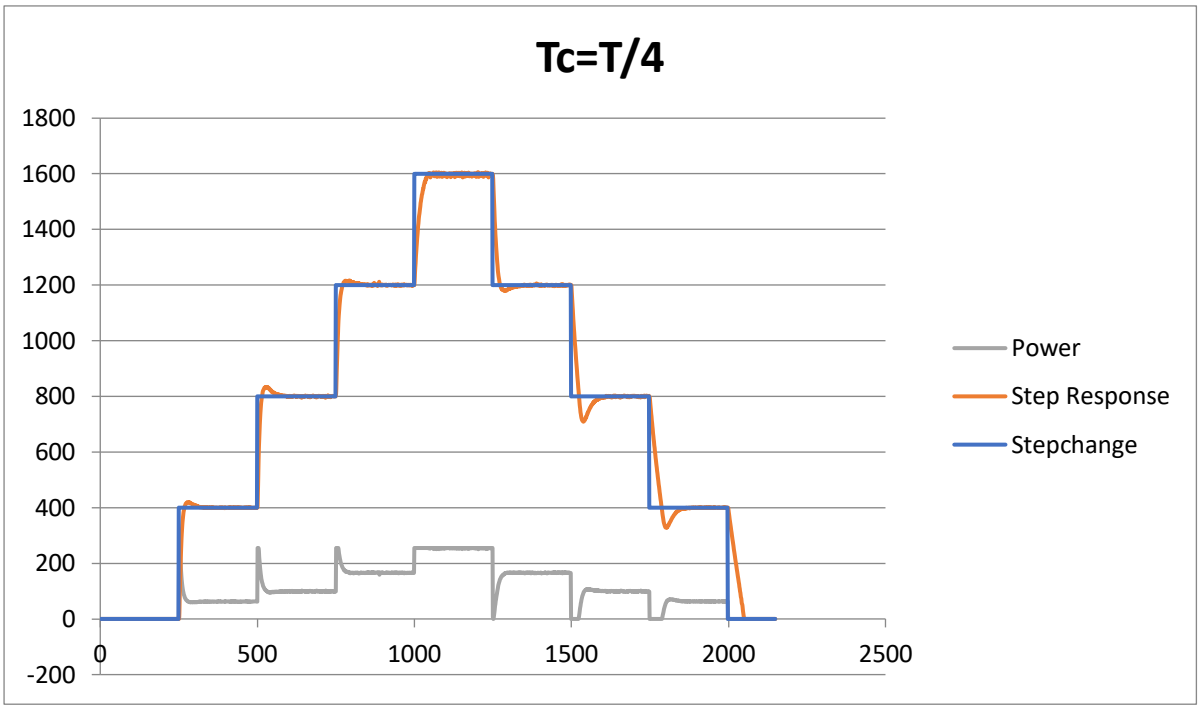
Kp_3=0,44 Ki_3=1



Καλή και γρήγορη απόκριση

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{4}}$

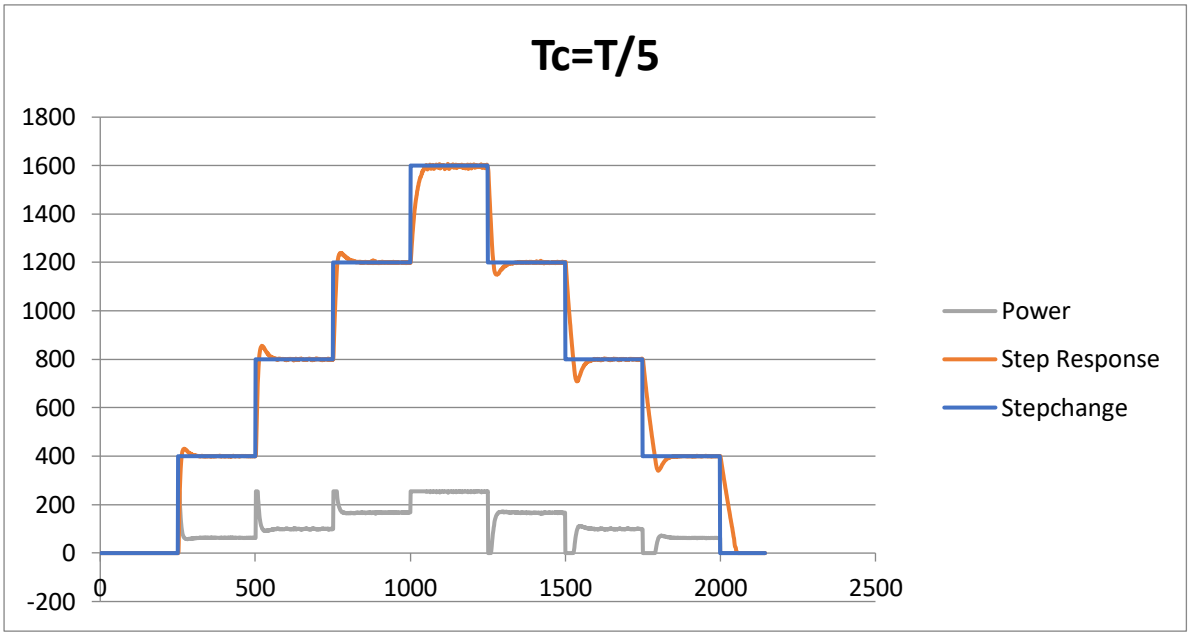
Kp_4=0,59 Ki_4=1,35



Καλύτερη και πιο γρήγορη απόκριση σε σχέση με την προηγούμενη περίπτωση

Για $\mathbf{Tc} = \frac{\mathbf{T}}{\mathbf{5}}$

Kp_5=0,74 Ki_5=2,1



Εξίσου πολύ καλή απόκριση.

Όπως και στο προηγούμενο πείραμα παρατηρούμε ότι:

1. Όταν η τιμή του Tc γίνεται μικρότερη, οι τιμές του Kp και Ki μεγαλώνουν.
2. Όσο μικραίνει η τιμή Tc και μεγαλώνουν οι τιμές του Kp και Ki βλέπουμε στα γραφήματα μας υπάρχει γρηγορότερη βηματική απόκριση.

Γενικά συμπεράσματα:

1. Και στα δυο πειράματα παρατηρήθηκε ότι όσο μικραίνουμε την τιμή του Tc τα αποτελέσματα γίνονται και καλύτερα, με καλύτερη ευστάθεια και με γρηγορότερη βηματική απόκριση.
2. Το μικρότερο φορτίο παρουσίασε πιο βελτιωμένα αποτελέσματα στις πέντε περιπτώσεις του Tc σε σχέση με το μεγαλύτερο φορτίο.

ΕΛΕΓΧΟΣ ΓΩΝΙΑΚΗΣ ΘΕΣΗΣ DC ΚΙΝΗΤΗΡΑ

Σε αυτή την ενότητα θα αναλύσουμε τον τρόπο σχεδίασης και επιβεβαίωσης πειραματικά ενός PID ελεγκτή για το πρόβλημα ελέγχου γωνιακής θέσης ενός DC κινητήρα. Η διαδικασία που προτείνουμε μπορεί να εφαρμοστεί στην πράξη και να παρέχει ικανοποιητικά αποτελέσματα χωρίς να απαιτούνται εξειδικευμένες γνώσεις ή κατανόηση της θεωρίας των ελεγκτών PID ή περίπλοκοι μαθηματικοί υπολογισμοί.

Υπολογισμός και ρύθμιση PID ελεγκτή για έλεγχο θέσης γωνίας

Για τον έλεγχο θέσης γωνίας προτείνεται η χρήση **PID** ελεγκτή. Για τον υπολογισμο των τριών παραμέτρων του **PID** ελεγκτή **Kp**, **Ki** και **Kd** εφαρμόσαμε το εξής τυπολόγιο (μέθοδος βέλτιστων κερδών ITAE [2]):

Υπολογισμός κέρδους Αναλογικού όρου (**Kp**):

$$\boldsymbol{K_p = \frac{0,5}{T_c^2}}$$

Υπολογισμός κέρδους Ολοκληρωτικού όρου (**Ki**):

$$\boldsymbol{K_i = \frac{1,25}{T_c^3}}$$

Υπολογισμός κέρδος Διαφορικού όρου (**Kd**):

$$\boldsymbol{K_d = \frac{0,1}{T_c}}$$

Πείραμα ελέγχου θέσης γωνίας

Στην ίδια κατασκευή που είχαμε χρησιμοποιήσει και στο προηγούμενο πείραμα μας, επιλέγουμε το επιθυμητό φορτίο και το τοποθετούμε στο DC κινητήρα μας. Στη συνέχεια ανοίγουμε το πρόγραμμα του Arduino. Πατάμε «Αρχείο» , «Άνοιγμα» και επιλέγουμε το πρόγραμμα «**Position_PID.ino**» που παρατίθεται στο ΠΑΡΑΡΤΗΜΑ Γ και προσαρμόζουμε τις τιμές του **Kp**, **Ki** και **Kd** με βάση τους υπολογισμούς που βρήκαμε, κάνουμε «ανέβασμα» και καταγράφουμε τα καινούργια αποτελέσματα. Στη συνέχεια περιγράφουμε την πειραματική διαδικασία και τα αποτελέσματα σε 2 διαφορετικές περιπτώσεις φορτίων που δοκιμάστηκαν, ένα μεγάλο (βαρύτερο) και ένα μικρότερο (ελαφρύτερο).

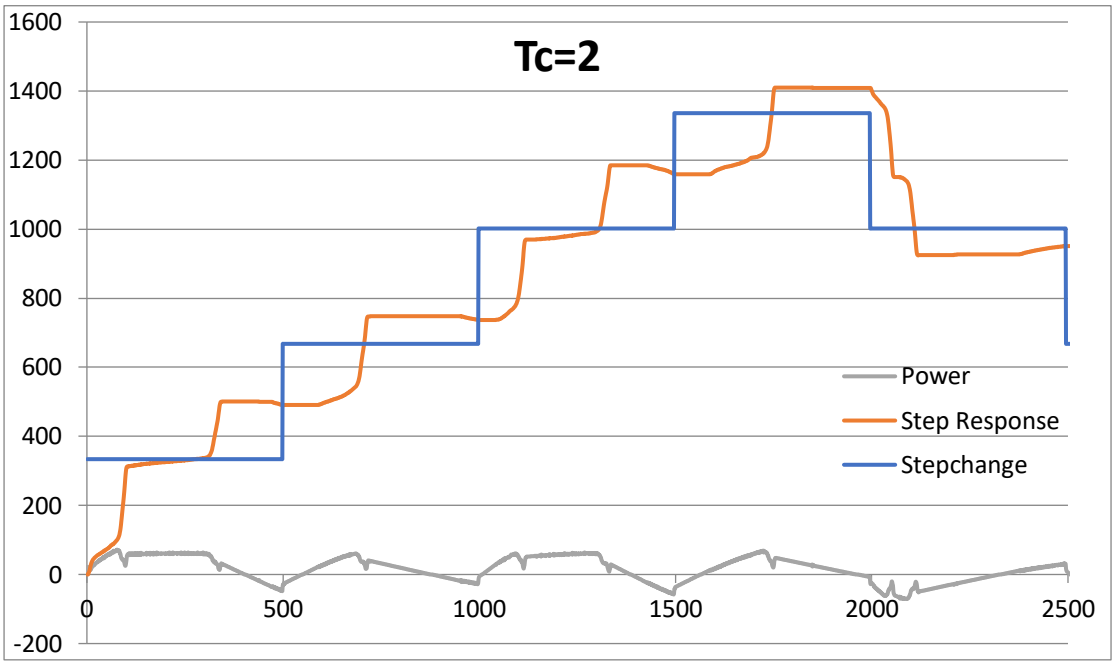
Πειραματικοί υπολογισμοί Kp, Ki και Kd για τις τιμές του Tc

Με βάση το παραπάνω τυπολόγιο θα υπολογίσουμε τις πειραματικές τιμές του **Kp**, **Ki** και **Kd** στις πέντε διαφορετικές τιμές του **Tc** που θα χρησιμοποιηθούν για τα πειραματικά αποτελέσματα που θα ακολουθήσουν στο μεγάλο (βαρύτερο) και στο μικρότερο (ελαφρύτερο) φορτίο για διαδοχικές βηματικές αλλαγές 90 μοιρών.

Tc	Kp	Ki	Kd
2	0,13	0,16	0,05
1	0,5	1,25	0,1
0,75	0,9	3	0,13
0,5	2	10	0,2
0,25	8,3	80	0,4

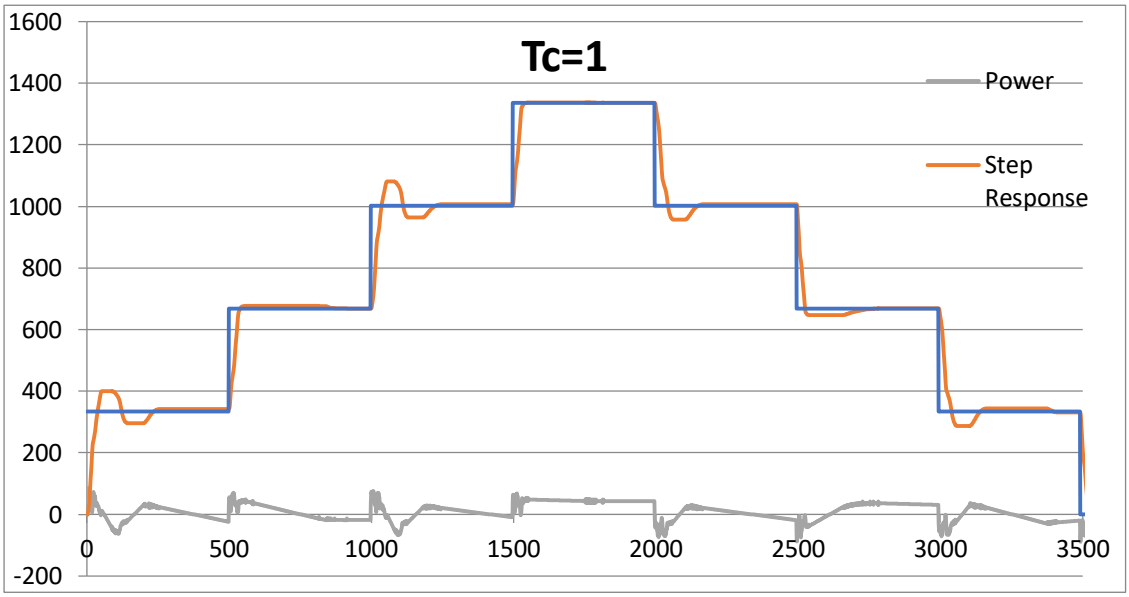
Πειραματικά αποτελέσματα με μεγάλο φορτίο

Για **Tc=2** προκύπτει:



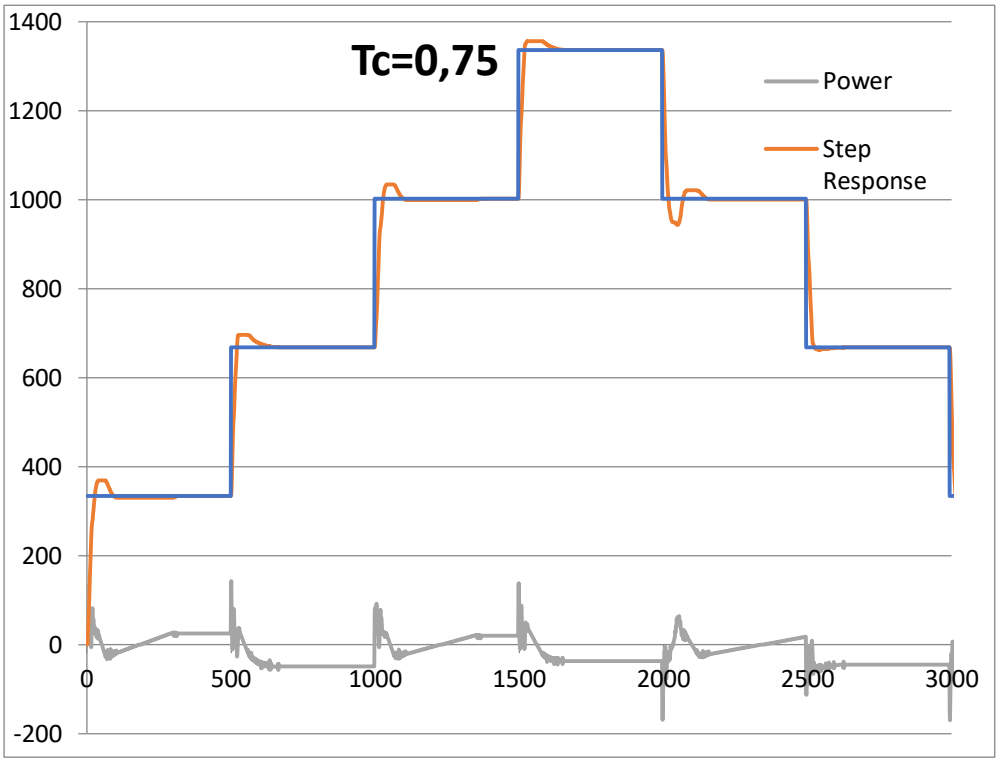
Αργή απόκριση στις βηματικές αλλαγές με μεγάλο μόνιμο σφάλμα λόγω υποτονικού σήματος ελέγχου (μικρά κέρδη).

Για **Tc=1** προκύπτει:



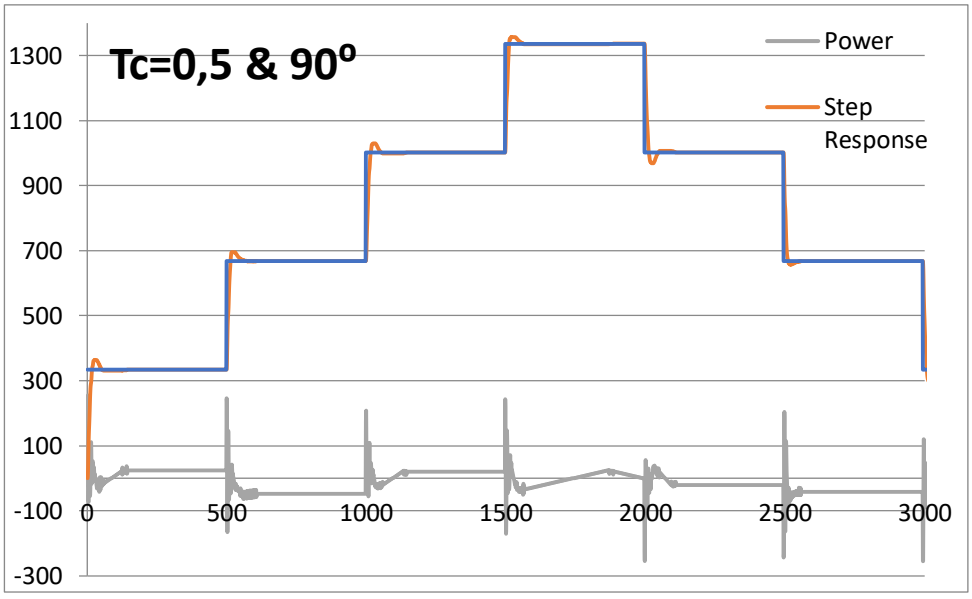
Αρκετά καλή απόκριση στις βηματικές αλλαγές, με μικρή υπερύψωση και χωρίς μόνιμο σφάλμα.

Για **Tc=0,75** προκύπτει:



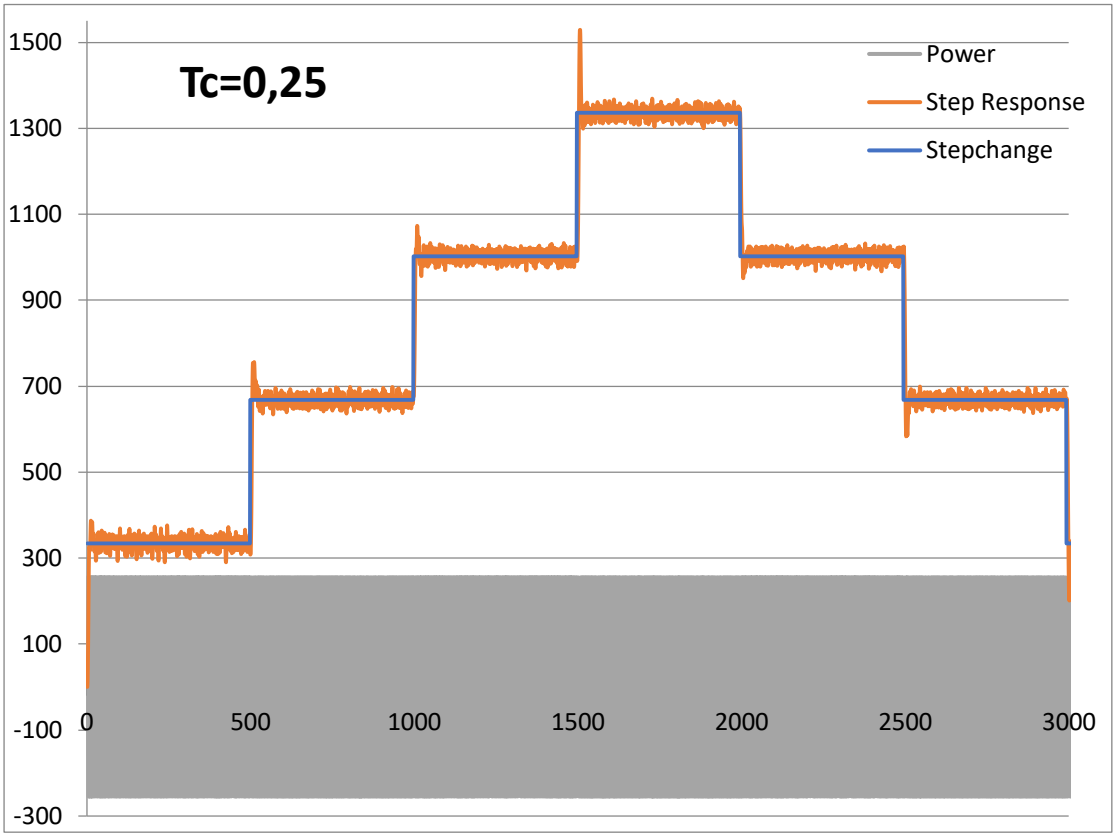
Καλύτερη απόκριση στις βηματικές αλλαγές, με μικρότερη υπερύψωση και χωρίς μόνιμο σφάλμα.

Για **Tc=0,5** προκύπτει:



Καλύτερη απόκριση σε σύγκριση με τις προηγούμενες περιπτώσεις. Το σύστημα είναι πιο ευέλικτο και μπορεί να ανταποκριθεί γρήγορα σε αλλαγές. Βέλτιστη απόκριση στις βηματικές αλλαγές, με ελάχιστη υπερύψωση και χωρίς μόνιμο σφάλμα. Παρά τα αυξημένα κέρδη το σήμα ελέγχου είναι ακόμα εντός ορίων.

Για **Tc=0,25** προκύπτει:

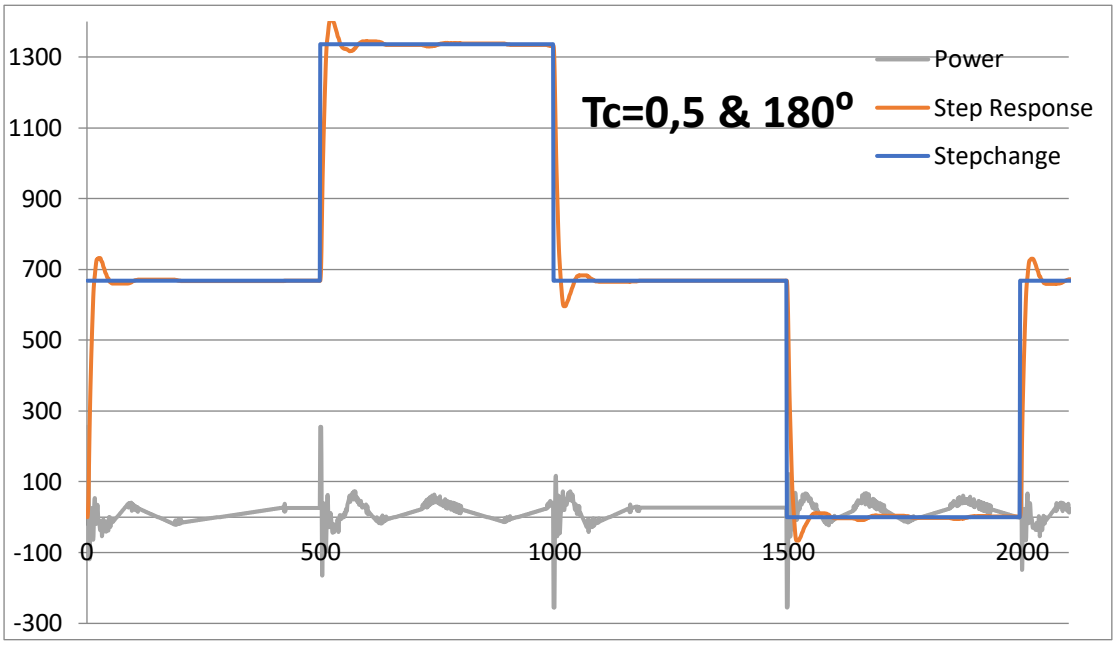


Αν και το σύστημα αποκρίνεται πολύ γρήγορα και ελέγχεται άμεσα και με σχετική ακρίβεια, τα πολύ μεγάλα κέρδη οδηγούν το σήμα ελέγχου σε έντονο κορεσμό (και άλματα ανάμεσα στην ελάχιστη (-255) και την μέγιστη (255) τιμή του με μεγάλη συχνότητα), με αποτέλεσμα την εισαγωγή έντονων κραδασμών-ταλαντώσεων, που υποβαθμίζουν σημαντικά την λειτουργία του στην μόνιμη κατάσταση.

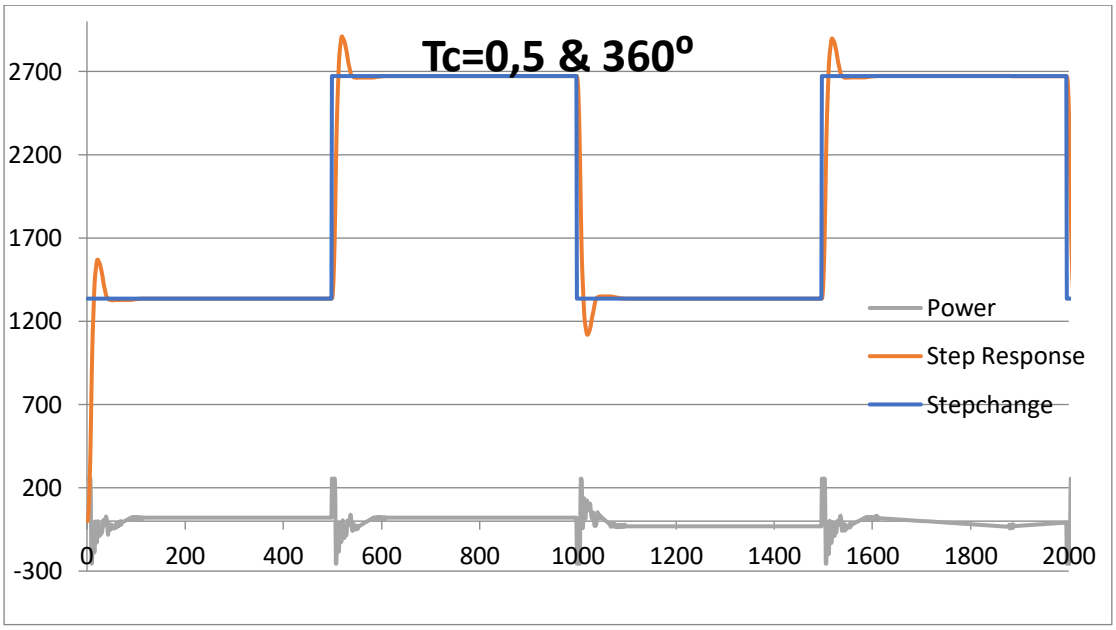
Παρατηρούμε οτι :

a) Το δείγμα που είναι πιο σταθερό είναι οταν το **Tc=0,5**
b) Όταν οι τιμές του κέρδους είναι μικρές έχουμε μεγάλες αποκλίσεις (Tc=2)
c) Όταν οι τιμές του κέρδους είναι μεγάλες έχουμε πολλές ταλαντώσεις (Tc=0,25)

Για **Tc=0,5** στις **180º** προκύπτει:

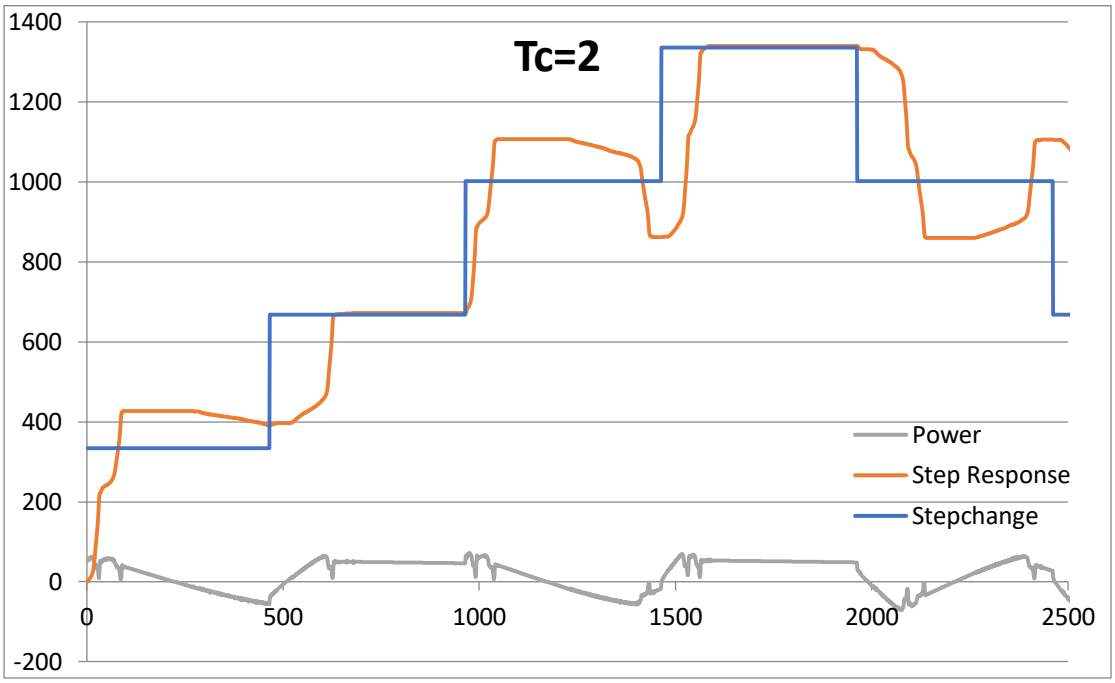


Για **Tc=0,5** στις **360º** προκύπτει:



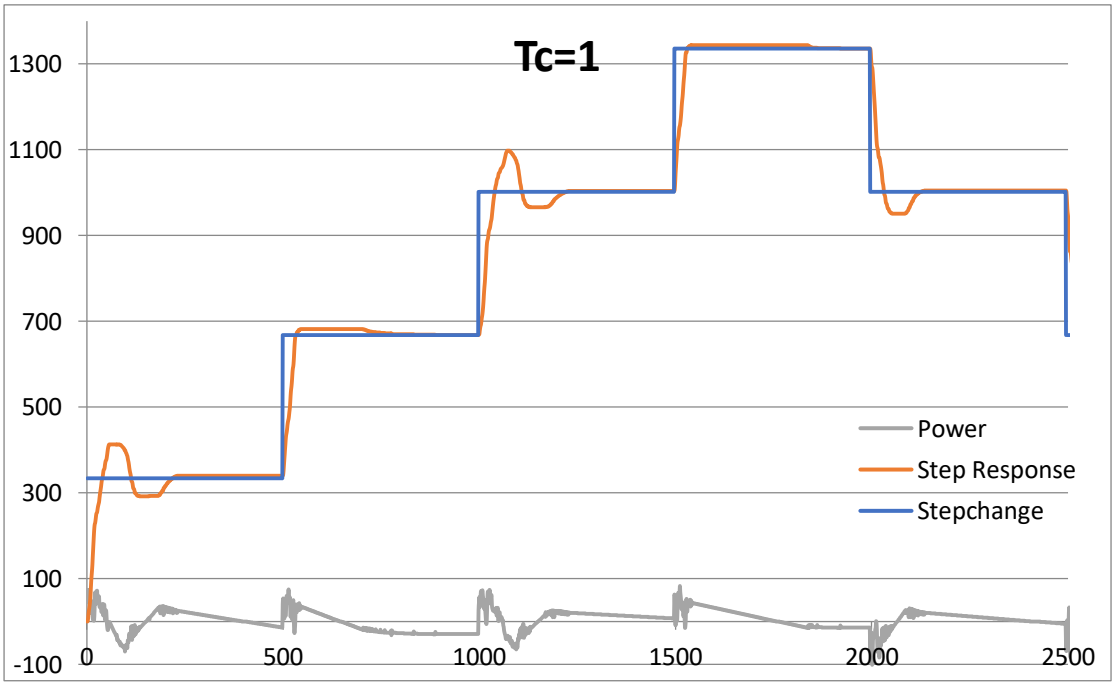
Πειραματικά αποτελέσματα με μικρό φορτίο

Για **Tc=2** προκύπτει:



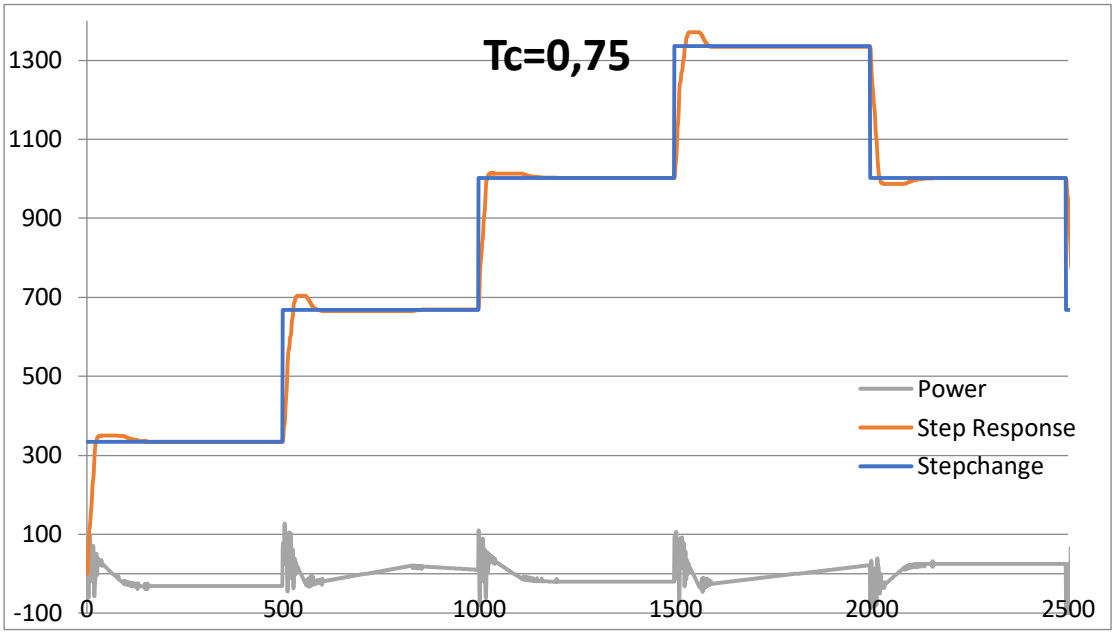
Βελτιωμένη εικόνα η σε σχέση με την περίπτωση με μεγαλύτερο φορτίο. Ακόμα όμως υπάρχει αργή απόκριση, μεγάλη υπερύψωση και μόνιμα σφάλματα.

Για **Tc=1** προκύπτει:



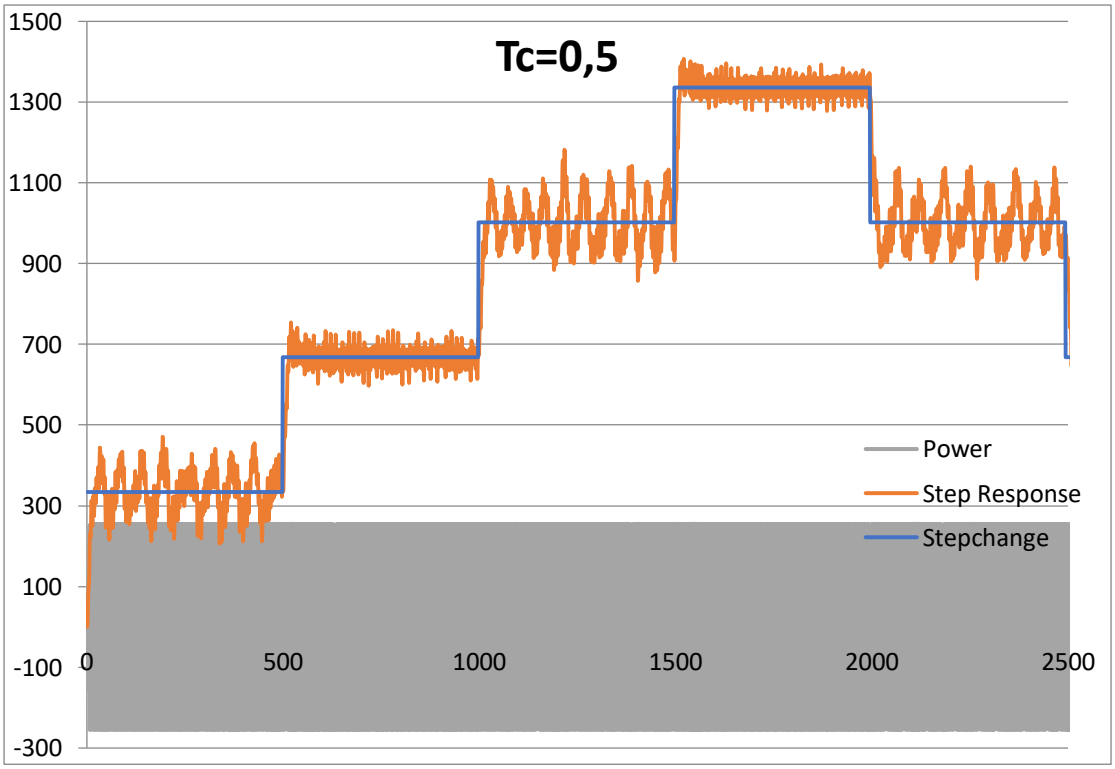
Καλύτερη απόκριση σε σχέση με Tc=2, αλλά εξακολουθεί να έχει μεγάλη υπερύψωση και χρόνο αποκατάστασης. Τα μόνιμα σφάλματα εξαλείφονται.

Για **Tc=0,75** προκύπτει:



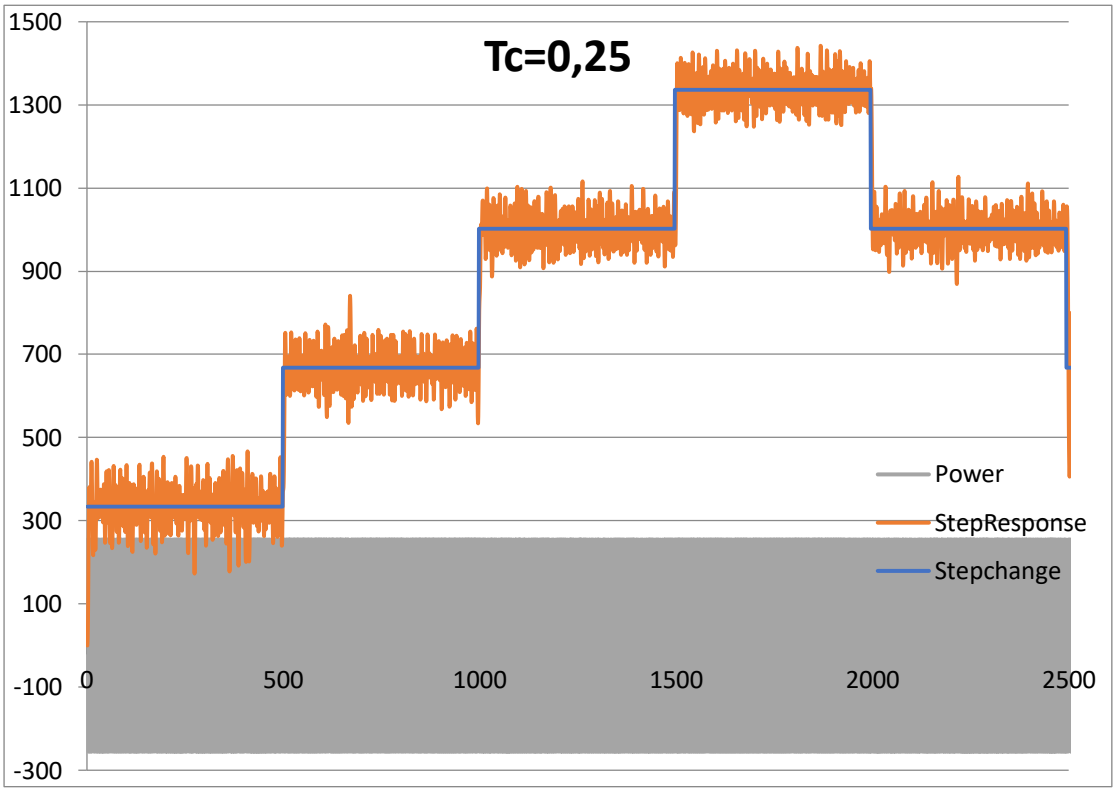
Καλύτερη απόκριση στις βηματικές αλλαγές, με μικρότερη υπερύψωση και χωρίς μόνιμο σφάλμα.

Για **Tc=0,5** προκύπτει:



Παρατηρούμε ακόμα πιο έντονα το φαινόμενο του κορεσμού και τις αρνητικές επιδράσεις του, όπως και στο προηγούμενο πείραμα.

Για **Tc=0,25** προκύπτει:

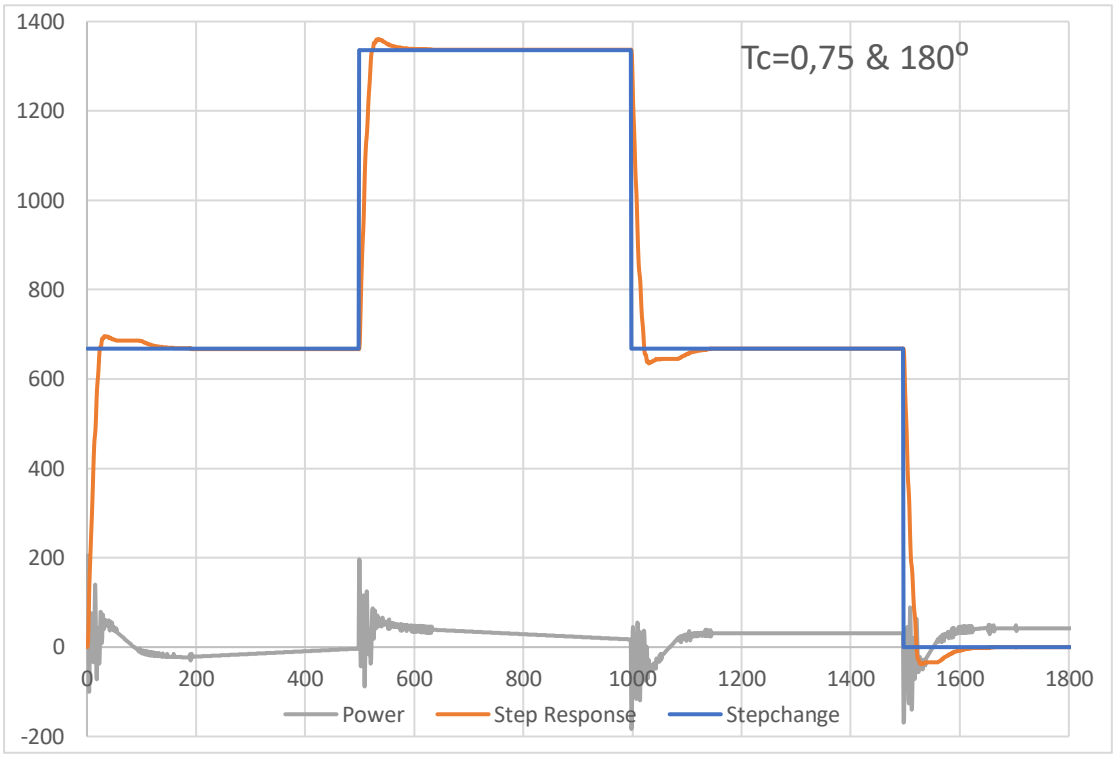


Με ακόμα μεγαλύτερα κέρδη ο έλεγχος καθίσταται ακόμα πιο προβληματικός.

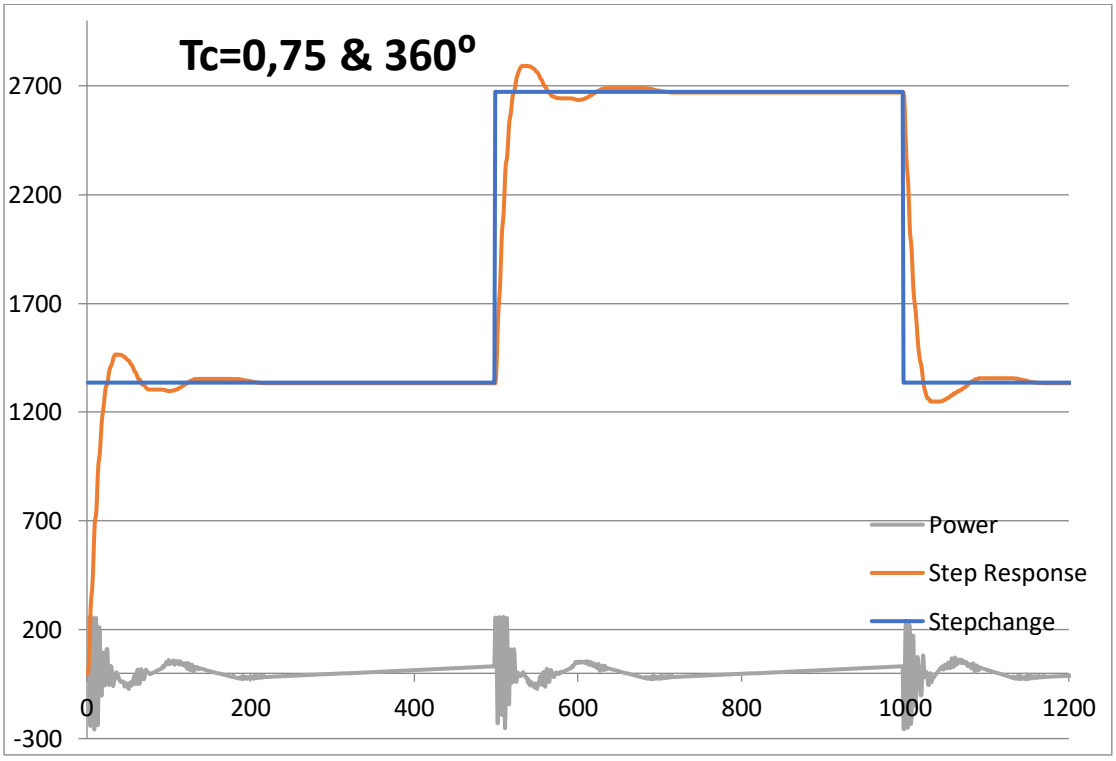
Παρατηρούμε οτι :

d) Το δείγμα που είναι πιο σταθερό είναι οταν το **Tc=0,75**
e) Όταν οι τιμές του κέρδους είναι μικρές έχουμε μεγάλες αποκλίσεις (Tc=2)
f) Όταν οι τιμές του κέρδους είναι μεγάλες έχουμε πολλές ταλαντώσεις (Tc=0,25& Tc=0,5)

Για **Tc=0,75** στις **180º** προκύπτει:



Για **Tc=0,75** στις **360º** προκύπτει:



Γενικά συμπεράσματα:

1. Διαφορετικά Tc για διάφορες ανάγκες:
   - Η επιλογή της επιθυμητής σταθεράς χρόνου (Tc) εξαρτάται από τις απαιτήσεις του συστήματος. Σε γενικές γραμμές, μικρότερα Tc οδηγούν σε γρηγορότερη απόκριση (μεγάλα κέρδη), ενώ μεγαλύτερα Tc σε πιο αργή απόκριση (μικρά κέρδη).

2. Ανεπάρκεια ελέγχου :
   - Πάντοτε υπάρχει ένα άνω όριο Tcmax, που αν ξεπεραστεί οδηγεί σε ανεπαρκή έλεγχο λόγω αδυναμίας υπερνίκησης των στατικών τριβών (πολύ μικρά κέρδη).
 - Φυσικά υπάρχει και ένα κάτω όριο Tcmin που αν ξεπεραστεί οδηγεί σε ανεπαρκή έλεγχο λόγω έντονου κορεσμού του σήματος ελέγχου που γεννά έντονες ταλαντώσεις στην μόνιμη κατάσταση (πολύ μεγάλα κέρδη).

3. Προσαρμογή στο φορτίο:
   - Η βέλτιστη τιμή Tc και οι τιμές Tcmin , Tcmax εξαρτώνται από τον κινητήρα και από το φορτίο στον άξονα του. Στην περίπτωση μεγάλου φορτίου, μπορεί να χρειαστεί μεγαλύτερο Tc για να διαχειριστεί τη διακύμανση.

4. Σημασία δοκιμών και προσαρμογής:
   - Η βέλτιστη τιμή Tc μπορεί να απαιτεί δοκιμές και προσαρμογή στο συγκεκριμένο σύστημα και τις συνθήκες λειτουργίας.

Συνεπώς, η επιλογή του Tc είναι ένα ευαίσθητο θέμα που πρέπει να προσαρμόζεται ανάλογα με τις ανάγκες, τις συνθήκες λειτουργίας και τον χαρακτήρα του συστήματος.
Σε γενικές γραμμές, για μεγαλύτερη ευστάθεια, ισορροπία μεταξύ ταχύτητας απόκρισης και κορεσμού σήματος ελέγχου, ένα Tc γύρω στο 0.75 φαίνεται να λειτουργεί καλά και για τα δύο φορτία που δοκιμάστηκαν στα πειράματα μας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] PID Tuning using the SIMC rules, Sigurd Skogestad, NTNU, https://folk.ntnu.no

[2] Tuning PID controllers using the ITAE criterion, https://www.ijee.ie

[3] PID controllers, Theory, Design and Tuning, Astrom & Hagglund

[4] Ανάπτυξη εφαρμογών με το Arduino , Παπάζογλου & Λιωνής. Εκδ. Τζιόλα

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Α

```
#include <PID_v1.h>
const int in1Pin = 10;
const int in2Pin = 9;
const int pot = A1;           // Potentiometer input
const int motor = 11;          // motor output pin
boolean dirmotor;             // motor direction
double stepchange, stepchange1, stepchange2,r;
unsigned long sampletime=0,steptime = 0;
unsigned long now2=0, lastMessage2=0;
unsigned long count=0;
unsigned long timep,time,etime;


const int sampleRate = 20;        // Περίοδος δειγματοληψίας σε msec
const int sampleRate2 = 20000;        // Περίοδος δειγματοληψίας σε μsec

void setMotor(int speed, boolean reverse)   // ρουτίνα ελέγχου στροφών
{             // κινητήρα με PWM και ελέγχου
  analogWrite(motor, speed);        // φοράς περιστροφής
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}


void setup() {
    // Παράμετροι αυτόματης αλλαγής σημείου αναφοράς (Setpoint)
    r=0.0;

  stepchange1 = 0.0;
  stepchange2 = 255.0;
  stepchange = 255.0;
  steptime = 5000;    // χρονική διάρκεια βηματικής αλλαγής σε  millis

  dirmotor=LOW;       // φορά περιστροφής κινητήρα

  Serial.begin(9600);           // Σύνδεση Channel A στο  pin 3
//  pinMode(3,INPUT);
//  attachInterrupt(1,transition,CHANGE);              // ορισμός διακοπής στο pin 3
//  timep=micros();         // set the initial time

  pinMode(2,INPUT);
  attachInterrupt(0,transition,CHANGE); // ορισμός διακοπής στο pin 2
  pinMode(3,INPUT);
  attachInterrupt(1,transition,CHANGE); // ορισμός διακοπής στο pin 3
  timep=micros();
```

```
}

void transition()       // ρουτίνα εξυπηρέτησης της διακοπής
{
 count++;
}

void loop() {

  setMotor(r, dirmotor); // αποστολή της εξόδου (σήμα δράσης) του PID


 time=micros();
 etime=time-timep;
 if (etime > sampleRate2)    // polling για την ενημέρωση τιμών μετρητή και
 {         // εμφάνιση στην σειριακή θύρα
          // ανάγνωση νέας τιμής μετρητή
   Serial.print(r);
   Serial.print('\t');
   Serial.println(count);

   count=0;          // reset counter
   timep=time;         // reset timer
  }
// κώδικας αυτόματης αλλαγής τιμής setpoint με polling
 now2=millis();
 if (now2 - lastMessage2 > steptime)
 {
  r = r + stepchange;
  if (r >= stepchange2)
    stepchange = -stepchange;
  if (r <= stepchange1)
    stepchange = -stepchange;
  lastMessage2 = now2;
 }
}
```

ΠΑΡΑΡΤΗΜΑ Β

```
#include <PID_v1.h>
const int in1Pin = 10;
const int in2Pin = 9;
const int pot = A1;          // Potentiometer input
const int motor = 11;          // motor output pin
boolean dirmotor;            // motor direction
double stepchange, stepchange1, stepchange2, r;
unsigned long sampletime = 0, steptime = 0;
unsigned long now2 = 0, lastMessage2 = 0;
unsigned long count = 0;
unsigned long timep, time, etime;
double Setpoint, Input, Output;
double Output2, precomp;

// Tuning parameters  PID controller
float Kp = 0.74;  // Κέρδος     Αναλογικού       όρου   (P=proportional)
float Ki = 2.1; // Κέρδος Ολοκληρωτικού  όρου  (I = Integral)
float Kd = 0.0;   // Κέρδος  Διαφορικού        όρου (D=differential)



PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
const int sampleRate = 20;        // Περίοδος δειγματοληψίας σε msec
const int sampleRate2 = 20000;        // Περίοδος δειγματοληψίας σε μsec

void setMotor(int speed, boolean reverse)   // ρουτίνα ελέγχου στροφών
{ // κινητήρα με PWM και ελέγχου
  analogWrite(motor, speed);        // φοράς περιστροφής
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}


void setup() {
  // Παράμετροι αυτόματης αλλαγής σημείου αναφοράς (Setpoint)
  r=0.0;
  stepchange1 = 0.0;
  stepchange2 = 1600.0;
  stepchange = 400.0;
  //stepchange = 600.0;
  steptime = 5000; // χρονική διάρκεια βηματικής αλλαγής σε millis



  count = 0.0;
  dirmotor = LOW;     // φορά περιστροφής κινητήρα
  Setpoint = r;
```

```
  Input = count;
  myPID.SetMode(AUTOMATIC);     //Turn on the PID loop
  myPID.SetSampleTime(sampleRate);    //Sets the sample rate
  myPID.SetOutputLimits(0, 255);
  Serial.begin(9600);           // Σύνδεση Channel A στο  pin 3
//  pinMode(3, INPUT);
//  attachInterrupt(1, transition, CHANGE);            // ορισμός διακοπής στο pin 3
//  timep = micros();       // set the initial time
  pinMode(2, INPUT);
  attachInterrupt(0, transition, CHANGE); // ορισμός διακοπής στο pin 2
  pinMode(3, INPUT);
  attachInterrupt(1, transition, CHANGE); // ορισμός διακοπής στο pin 3
  timep = micros(); // set the initial time

}

void transition()       // ρουτίνα εξυπηρέτησης της διακοπής
{
  count++;
}

void loop() {

  Setpoint = r;
  myPID.Compute();        // εκτέλεση βρόχου PID

  setMotor(Output, dirmotor); // αποστολή της εξόδου (σήμα δράσης) του PID

  // ελεγκτή σε μορφή PWM στον κινητήρα
  time = micros();
  etime = time - timep;
  if (etime > sampleRate2)    // polling για την ενημέρωση τιμών μετρητή και
  { // εμφάνιση στην σειριακή θύρα
    Input = count;      // ανάγνωση νέας τιμής μετρητή
    Serial.print((int)Setpoint);
    Serial.print('\t');
    Serial.print((int)Input);
    Serial.print('\t');
    Serial.println((int)Output);
    //Serial.print('\t');
    //Serial.println((int)etime);

    count = 0;        // reset counter
    timep = time;       // reset timer
  }
  // κώδικας αυτόματης αλλαγής τιμής setpoint με polling
  now2 = millis();
  if (now2 - lastMessage2 > steptime)
  {
    r = r + stepchange;
```

```
    if (r >= stepchange2)
      stepchange = -stepchange;
    if (r <= stepchange1)
      stepchange = -stepchange;
    lastMessage2 = now2;
  }
}
```

ΠΑΡΑΡΤΗΜΑ Γ

```
const int in1Pin = 10;
const int in2Pin = 9;
const int motor = 11; // motor output
#define MOTOR_ENCODER_PIN_1  2
#define MOTOR_ENCODER_PIN_2  3
volatile signed long count=0;
boolean A,B;
boolean dirmotor;    // motor direction
byte state,statep,index;
int QEM[16]={0,-1,0,1,1,0,-1,0,0,1,0,-1,-1,0,1,0};

void setMotor(int speed, boolean reverse)
{
  analogWrite(motor, speed);
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}


void Achange()
{
  A=digitalRead(MOTOR_ENCODER_PIN_1);
  B=digitalRead(MOTOR_ENCODER_PIN_2);
  // determine state value
  if ((A==HIGH) && (B==HIGH)) state=0;
  if ((A==HIGH) && (B==LOW))  state=1;
  if ((A==LOW) && (B==LOW))   state=2;
  if ((A==LOW) && (B==HIGH))  state=3;
  index = 4*state + statep;
  count = count + QEM[index];
  statep = state;
}
void Bchange()
{
  A=digitalRead(MOTOR_ENCODER_PIN_1);
  B=digitalRead(MOTOR_ENCODER_PIN_2);
  // determine state value
  if ((A==HIGH) && (B==HIGH)) state=0;
```

```
  if ((A==HIGH) && (B==LOW))  state=1;
  if ((A==LOW) && (B==LOW))   state=2;
  if ((A==LOW) && (B==HIGH))  state=3;
  index = 4*state + statep;
  count = count + QEM[index];
  statep = state;
}

double Input1, Input2;
double lastTime;


// PID
float Kd=0.1;
float Kp=0.5;
float Ki=1.25;

float Kii,Kdd;

double e,u,uu,x,x1,y,y1,ulow,uhigh;
double b;

double f,a1,f1,a2,rf,a;  // prefilter

const int sampletime = 20000; // sampletime in micros
double stepchange, stepchange1, stepchange2,r,T;
unsigned long now = 0, lastMessage=0, steptime = 0;

void setup() {
  a=100.0;  // prefilter removed
  //a=10.0; // three pole
  //a=20.0; // ITAE wn=20 for b=0.5
  //a=8.0; // ITAE wn=15 for normal stick and b=1.0
  //a=6.0;   // ITAE wn=10 longer stick for b=1.0
  //a=4.0;   // ITAE wn=8 propeller for b=1.0
  y=0.0;y=0.0;y1=0.0;u=0.0;x=0.0;x1=0.0;e=0.0;
  ulow=-255.0;uhigh=255.0;
  rf=0.0;f=0.0;f1=0.0;

  T=sampletime/1000000.0;

  // prefilter coefficients
  a1=1.0/(1.0+a*T);a2=(a*T)/(1.0+a*T);

  // backward difference
  Kdd = Kd/T;
  Kii = Ki*T;

  b=0.5;
  //b=1.0;
```

```
//  r=0.0;
//  count=0;
//  stepchange1 =  0.0;
//  stepchange2 = 2400.0;
//  stepchange  =  60.0;
//  steptime = 500000;   // 3 sec in micros
////
//  r=500.0;
//  count=0;
//  stepchange1 =  500.0;
//  stepchange2 = 2000.0;
//  stepchange  =  500.0;
//  steptime = 3000000;   // 3 sec in micros
////
  r=334.0;
  count=0;
  stepchange1 =  0.0;
  stepchange2 =  1336.0;
  stepchange  =  334.0;
  steptime = 10000000;   // 10 sec in micros

//   r=250.0;
//  count=0;
//  stepchange1 =  250.0;
//  stepchange2 = 2000.0;
//  stepchange  =  250.0;
//  steptime = 3000000;   // 3 sec in micros

 //setPwmFrequency(11, 1024);
 // setPwmFrequency(11, 256);
 //setPwmFrequency(11, 32);  // 980 Hz
 //setPwmFrequency(11, 64);  // default 490 Hz
    //setPwmFrequency(11, 8);
    //setPwmFrequency(11, 1);  // 32 KHz

  dirmotor=LOW;
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);

  pinMode(MOTOR_ENCODER_PIN_1, INPUT);
  pinMode(MOTOR_ENCODER_PIN_2, INPUT);
  // Use CHANGE instead of RISING for more resolution.
  attachInterrupt(0,Achange,CHANGE);
  attachInterrupt(1,Bchange,CHANGE);

  Serial.begin(115200);       //Start a serial session


 // read the initial value of A & B
```

```
  A=digitalRead(2);
  B=digitalRead(3);

  // set initial state value
  if ((A==HIGH) && (B==HIGH)) statep=1;
  if ((A==HIGH) && (B==LOW))  statep=2;
  if ((A==LOW) && (B==LOW))   statep=3;
  if ((A==LOW) && (B==HIGH))  statep=4;
}


void loop() {

  if (u > 0) dirmotor=LOW;    // CCW direction of movement
  else dirmotor=HIGH;               // CW  direction of movement

  if (u < 0) uu=-u;
  else uu=u;

  setMotor(uu, dirmotor);


 if(micros()-lastTime >= sampletime)
   {
    lastTime = micros();

    // prefilter setpoint
    f=a1*f1+a2*r;
    f1=f;
    rf=f;

    y=count;
    e=rf-y;


    u=Kp*(b*rf-y)+x+Kdd*(y1-y);

    if (u > uhigh)  u = uhigh;
    else if (u < ulow) u = ulow;

    x=x1+Kii*e;   // integral term
//    if (x > uhigh)  x = uhigh;
//    else if (x < ulow) x = ulow;

    x1=x;
    y1=y;

    Serial.print((int)r);
    Serial.print('\t');
    Serial.print(y);
```

```
    Serial.print('\t');
    Serial.println((int)u);

  }

  now=micros();
  if (now - lastMessage > steptime)
  {
    r = r + stepchange;
    if (r >= stepchange2)
      stepchange = -stepchange;
    if (r <= stepchange1)
      stepchange = -stepchange;
    lastMessage = now;
  }
}



void setPwmFrequency(int pin, int divisor) {
  byte mode;
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 64: mode = 0x03; break;
      case 256: mode = 0x04; break;
      case 1024: mode = 0x05; break;
      default: return;
    }
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
    switch(divisor) {
      case 1: mode = 0x01; break;
      case 8: mode = 0x02; break;
      case 32: mode = 0x03; break;
      case 64: mode = 0x04; break;
      case 128: mode = 0x05; break;
      case 256: mode = 0x06; break;
      case 1024: mode = 0x07; break;
      default: return;
    }
    TCCR2B = TCCR2B & 0b11111000 | mode;
  }
}
```