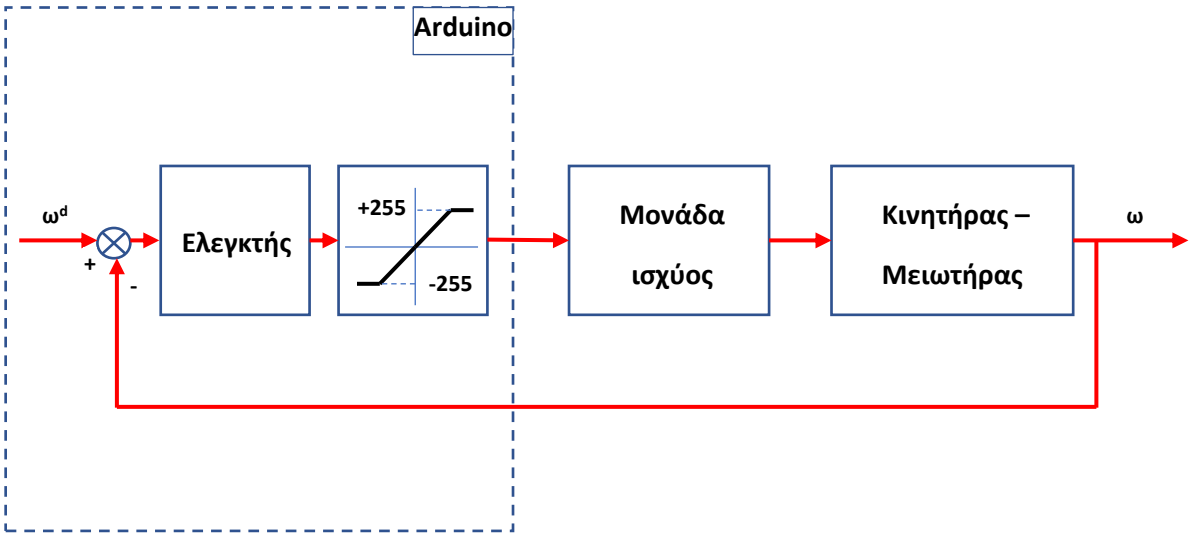
Εργαστηριακή Άσκηση 3

Έλεγχος κλειστού βρόχου γωνιακής ταχύτητας κινητήρα DC με ενσωματωμένο μειωτήρα

Υλοποίηση Αναλογικού – Ολοκληρωτικού Ελέγχου (PI)

Στην εργαστηριακή Άσκηση 1 διαπιστώσαμε ότι μπορούμε να δώσομε κάποιον αριθμό ως εντολή από τον μικρουπολογιστή και ο κινητήρας να περιστραφεί. Η συνάρτηση μεταφοράς ανοικτού βρόχου που προσδιορίσθηκε μας λέει επίσης πόση γωνιακή ταχύτητα περιμένομε ανά μονάδα εντολής αλλά και το πόσο γρήγορα φτάνει ο κινητήρας την ταχύτητα αυτή.

Για αν μπορέσομε να ελέγξομε πραγματικά (σωστά) την γωνιακή ταχύτητα, θα πρέπει να είμαστε σε θέση να ζητήσομε από το σύστημα συγκεκριμένη, επιθυμητή τιμή, ω^d και ο κινητήρας να την δώσει. Κάτι τέτοιο μπορεί να γίνει μόνο με σύστημα κλειστού βρόχου. Στο Σχήμα που ακολουθεί φαίνεται το σύστημα αυτό.



Θα προσπαθήσομε στην συνέχεια να υλοποιήσομε με την βοήθεια του μικροελεγκτή Arduino Uno κατάλληλους «Ελεγκτές», ώστε να ελέγξομε την γωνιακή ταχύτητα όσο γίνεται καλύτερα. Το «στοιχείο κορεσμού» που φαίνεται στο σχήμα (μετά τον ελεγκτή), αποτυπώνει το γεγονός ότι η μονάδα ισχύος δέχεται εντολές με τιμές μεταξύ -255 και 255. Πρόκειται για ένα στοιχείο που εισάγει όπως λέμε «μη γραμμικότητες» στο σύστημα.

Αναλογικός – Ολοκληρωτικός ελεγκτής (Proportional - Integral Control)

Θα προσθέσομε ολοκληρωτικό έλεγχο για να εξαφανίσομε τα σφάλματα μόνιμης κατάστασης. Ο ολοκληρωτικός όρος του ελεγκτή είναι το ολοκλήρωμα του σφάλματος στον χρόνο επί το «κέρδος» Ki του ολοκληρωτικού όρου.

Η εντολή :

Integral=Integral+((PreviousErr+Err)/2)*0.01;

«αθροίζει» κάθε 10 ms το σφάλμα και είναι μια καλή αριθμητική προσέγγιση του ζητούμενου ολοκληρώματος.

Τοποθετήστε την παραπάνω εντολή ως τελευταία εντολή στον ατέρμονα βρόχο void loop() του προγράμματος που υλοποιεί αναλογικό έλεγχο και διορθώστε την εντολή ελέγχου σε :

cmd=Kp*(Err)+Ki*Integral;

Φυσικά πρέπει να ορίσετε τις καινούργιες μεταβλητές στην αρχή του προγράμματος :

float Integral=0, Ki=0.0;

Ρύθμιση του PI Ελεγκτή

Το σύστημα κλειστού βρόχου, με την προσθήκη του ολοκληρωτικού όρου, έχει γίνει σύστημα 2ης τάξης, άρα μπορεί εν δυνάμει να ταλαντωθεί ή και να οδηγηθεί σε αστάθεια αν δεν επιλεγούν σωστά τα «κέρδη» Kp και Ki.

Από την θεωρία γνωρίζομε ότι η συνάρτηση μεταφοράς κλειστού βρόχου, με τον PI ελεγκτή είναι :

$$\boldsymbol{G_c(s) = \frac{\frac{(K_p s + K_i)K_{tot}}{T}}{s^2 + (\frac{K_p K_{tot} + 1}{T})s + \frac{K_i K_{tot}}{T}}}$$

Παρά το ότι η συνάρτηση αυτή δεν είναι η τυπική συνάρτηση μεταφοράς ενός δευτεροταξίου συστήματος (Βλέπε Θεωρητική Άσκηση), θα χρησιμοποιήσομε τους τύπους που γνωρίζομε :

$$\omega_n^2 = \frac{K_i K_{tot}}{T} \qquad \kappa\alpha\iota \quad 2\zeta\omega_n = \frac{K_p K_{tot} + 1}{T}$$

ώστε από τις «προδιαγραφές» για την κυκλική συχνότητα ω_n και την απόσβεση ζ που θα θέσομε, να υπολογίσομε στη συνέχεια τα κέρδη Kp και Ki.

Η κυκλική συχνότητα σχετίζεται άμεσα με την «ταχύτητα απόκρισης» του συστήματος. Η τελευταία, εκτιμάται με την βοήθεια του «χρόνου ανόδου» (rise time, Tr). Για τον χρόνο ανόδου όμως, στην περίπτωση των δευτεροτάξιων συστημάτων, δεν διαθέτομε αναλυτική σχέση. Θα χρησιμοποιήσομε εναλλακτικά τον χρόνο κορυφής, Tp, για τον οποίο (και πάλι για το τυπικό δευτεροτάξιο σύστημα με υποαπόσβεση) ξέρομε :

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$$

Θέσετε προδιαγραφές. Π.χ :

Tp=0.1 sec (σημαίνει ότι το σύστημα σε 0.1 sec θα έχει πιάσει την υψηλότερη τιμή του, αρκετά γρήγορα δηλαδή)

ζ = 0.6 (μια αξιοπρεπής απόσβεση)

και υπολογίσετε στην συνέχεια, με βάση τους παραπάνω τύπους και τα χαρακτηριστικά της εγκατάστασης που βρήκατε κατά την ταυτοποίηση (δηλαδή Ktot και T), τα κέρδη Kp και Ki του ελεγκτή.

Δώστε τις τιμές αυτές στο πρόγραμμα στον Arduino και δείτε το αποτέλεσμα : Η γωνιακή ταχύτητα πρέπει να είναι αυτή που ζητήσατε! Το εξασφαλίζει αυτό ο ολοκληρωτικός όρος.

Υπολογίσετε εκ νέου Kp και Ki τέτοια που να δίνουν τον ίδιο χρόνο κορυφής αλλά λόγο απόσβεσης ζ=0.2 και παρατηρείστε ότι το σύστημα είναι τώρα πολύ ταλαντωτικό.

Και για τις δύο παραπάνω περιπτώσεις, σχεδιάσετε την καμπύλη απόκρισης της γωνιακής ταχύτητας σε είσοδο βαθμίδας 3000 παλμούς/sec.

«Απόρριψη» διαταραχών ροπής

Φρενάρετε τον άξονα με «λογική» ροπή και παρατηρείστε την γωνιακή ταχύτητα να υποχωρεί αρχικά, αλλά στην συνέχεια να επανέρχεται στην τιμή που θέλετε. Φυσικά πέρα από κάποιο όριο ροπής φρεναρίσματος, δεν μπορεί να συμβεί αυτό : Ο κινητήριος μηχανισμός έχει φτάσει στα όρια ισχύος του και κανένα σύστημα ελέγχου δεν μπορεί να τον κάνει να τα υπερβεί.

Κώδικας

```
#include <Encoder.h>
Encoder Enc(2, 3);      //O encoder είναι συνδεδεμένος στις ψηφιακές εισόδους 2 και 3 του arduino
long set_point,vel=0,pos=0,newP=0, Err=0, PreviousErr=0, cmd=0;
float Integral=0, Kp=0.28, Ki=10;

void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);              //DIRECTION Motor Channel B
  pinMode(8, OUTPUT);               //BRAKE Motor Channel B
  digitalWrite(13, HIGH);           //Establishes forward direction of Channel B
  digitalWrite(8, LOW);             //Disengage the Brake for Channel B
  set_point=3000;                   //Ζητούμενη ταχύτητα (παλμοί/sec)
  Enc.write(0);
        }

void loop()                         //Επαναλάμβανε συνεχώς :
{
```

```
  newP = Enc.read();                              //Διάβασε την γωνία στροφής
  vel = (newP-pos) * 100;                         //Υπολόγισε την γωνιακή ταχύτητα (παλμοί/sec)
  pos = newP;
  Err=set_point-vel;                              //Υπολόγισε το σφάλμα ταχύτητας
  Integral=Integral+(PreviousErr+Err)*0.005;  //Υπολόγισε το ολοκλήρωμα του σφάλματος στο χρόνο
  PreviousErr=Err;

  cmd=Kp*(Err)+Ki*Integral;                       //Υπολόγισε την εντολή υλοποιώντας PI Έλεγχο

    //Στείλε την εντολή στον κινητήρα
    if (cmd<0)  {cmd=-cmd; digitalWrite(13, LOW);}
        else {digitalWrite(13, HIGH);}
    if (cmd>255)  cmd=255;
    analogWrite(11, cmd);

  Serial.println (vel);
  delay(10);                          //περίμενε 10 ms
}
```