Controller PID cu Arduino

Prezentare generala.

In acest tutorial vei descoperi cum se poate realiza un controller PID avand o placa Arduino, cum functioneaza metoda PID si ce putem realiza cu aceasta metoda.

Controlul PID (proportional, integral, derivativ) este o metoda utilizata la scara larga in reglarea proceselor, cum ar fi reglarea temperaturii, reglarea nivelului de apa intr-o incinta, controlul vitezei unui motor electric sau pozitionarea capului de imprimanta cu jet de cerneala. Acestea sunt doar cateva exemple.

Desi aplicatiile difera intre ele, metoda de abordare folosind controlul PID este asemanatoare. Ecuatia care descrie comportamentul unui controller PID exista sub diverse forme, dar forma generala este urmatoarea:

$$Control = kP*Eroare + kI*\Sigma\ Eroare + kD * dP/dT$$

Eroarea reprezinta diferenta dintre valoarea actuala a procesului (sa presupunem ca reglam viteza de rotatie a unui motor) si valoarea pe care ne-o dorim sa o atingem (referinta).

Intr-un limbaj de programare, eroarea se poate exprima astfel:

$$Eroare = Valoare – Referinta$$

Σ Eroare reprezinta suma erorilor anterioare iar dP/dT - rata de schimb a valorii procesului in raport cu timpul. Nu vei insista foarte mult asupra ecuatiilor, deoarece vei descoperi mai tarziu cum functioneaza algoritmul direct intr-un sketch Arduino.

Coeficientul proportional kP, integral kI si derivativ kD sunt responsabili cu reglarea controllerului PID corespunzator procesului pe care il regleaza. Cu alte cuvinte, in cazul unui robot, daca reglezi corect PID-ul, atunci robotul va fi foarte rapid si foarte precis.

Control este o valoare de tensiune sau curent pe care o vei utiliza atunci cand vrei sa controlezi elementele de executie. In cazul unui robot, vei utiliza valoarea Control pentru a regla tensiunea de alimentare a motoarelor (elementul de executie).

Algoritmul controllerului PID.

Iti propun urmatorul pseudocod care iti va arata ca metoda PID nu este foarte complicata.

```
PID:
 Eroare = Referinta - Eroare_Actuala
 Integral = Integral + (Eroare*dt)
 Derivativ = (Eroare - Eroare_anterioara)/dt
 Control = (Eroare*kP) + (Integral*kI) + (Derivativ*kD)
 Eroare_anterioara = Eroare
 Asteapta(dt)
GOTO PID
```

Pseudocodul iti arata pasii pe care ii urmeaza un controller PID atunci cand se afla in functiune. Poate fi util atunci cand vrei sa programezi un controller PID, intr-un limbaj diferit de catre cel folosit in Arduino.

Mai jos poti observa un controller PID realizat in Arduino. Programul nu este orientat catre un exemplu anume, ci este unul general. Ramane la decizia ta sa alegi cum citesti valoarea Pozitie si cum realizezi comanda motorului.

```
 Actual = analogRead(Pozitie);
 Eroare = Referinta - Actual;

 if (abs(Eroare) < PragIntegral){ // previne saturatia integralei
  Integral = Integral + Eroare; // acumuleaza
 }
 else {
  Integral=0; // trece in 0 daca a depasit limita
 }
 P = Eroare*kP; // termenul proportional
 I = Integral*kI; // termenul integrativ
 D = (Eroare_anterioara-Actual)*kD; // termenul derivativ
 Drive = P + I + D; // Control total = P+I+D
 Drive = Drive*FactorScalare; // scaleaza Drive in domeniul 0-255
 if (Drive < 0){ // Verifica directia
  digitalWrite (Direction,LOW); // schimba directia
 }
 else { // schimba directia
  digitalWrite (Direction,HIGH);
 }
 if (abs(Drive)>255) {
  Drive=255;
 }
```

```
 analogWrite (Motor,Drive); // transmite un semnal PWM catre Motor
 Eroare_anterioara = Actual; // pastreaza valoarea actuala
}
```

Comanda PWM.

In exemplul de mai sus s-a utilizat o iesire PWM pentru a controla tensiunea de alimentare a unui motor. Valoarea din interiorul functiei analogWrite() trebuie sa fie de tip intreg si cuprinsa intre 0 si 255. Valoarea Drive este redimensionata pentru acest interval.

Reglarea PID-ului.

Partea de reglare a controllerului PID nu este foarte simpla. Ea se realizeaza din coeficientii kP, kI si kD.

Se procedeaza astfel:

- Initial coeficientii kI si kD se egaleaza cu 0 si se va utiliza numai coeficientul kP.
- Se va creste valoarea coeficientului kP pana cand raspunsul controllerului incepe si oscileaza. Cu alte cuvinte, daca incerci sa reglezi un robot urmaritor de linie, se va mari valoarea kP pana cand robotul incepe si „se plimba“ in jurul liniei, adica oscileaza. In acest moment s-a ajuns intr-un punct in care robotul nu se comporta corect, asa ca trebuie sa miscorezi putin valoarea kP pana cand robotul tinde sa oscileze dar nu foarte mult.
- Se va mari valoarea coeficientul kD care se comporta ca si cum ar anticipa inceputul unei oscilatii.
- In final, se va mari foarte putin coeficientul kI care imbunatateste timpul de raspuns al robotului.

Concluzie.

Scopul acestui tutorial este de a-ti realiza o scurta introducere in ceea ce se numeste Regulator PID. Tutorialul acopera aspectele importante de functionare si este util atunci cand doresti sa realizezi o comanda cat mai buna a unui proces. Ca exemplu, poti sa reglezi temperatura intr-o incinta sau a unui recipient (dar nu este absolut necesar sa utilizezi metoda PID), poti sa reglezi viteza de rotatie a unui motor sau poti mentine in echilibru un pendul invers.