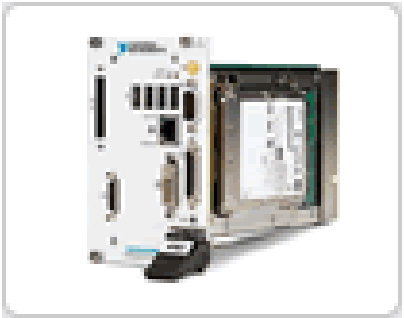
C8: Implementarea algoritmilor de reglare.

Cuprins (anterior):

- Introducere – Functii
- Achizitia datelor
- Notiunea de timp real
- Structura software a aplicatiilor de timp real
- Arhitecturi hardware si software de timp real
- Algoritmi de control
- Exemple



Cuprins (actual):

- Introducere
- Modele, filtre, implementare
- Alg. bi-pozitionali, tri-pozitionali, PID
- Alg. RST (general)
- Metode de acordare
- Exemple

C8: Implementarea algoritmilor de reglare.

1. Introducere – Tipuri algoritmi

Clasici:
- Bi-pozitionali
- Tri-pozitionali
- PID
- PID – iesire ON-OFF;

Moderni:
- RST
- Fuzzy
- Neuronali
- Etc.

C8: Implementarea algoritmilor de reglare.

2. Modele, filtre, implementare

In multe situatii se doreste ca masura achizitionata sa fie filtrata, sau ca referinta unui regulator sa urmareasca o traiectorie continua fara salturi mari. O solutie pentru aceasta situatie poate fi un sistem de tip ARX $B_m(q^{-1})/A_m(q^{-1})$ prezentat in urmatoarea ecuatie generala**.**

$$H_m\left(q^{-1}\right) \overset{def}{=} \frac{B_m\left(q^{-1}\right)}{A_m\left(q^{-1}\right)}$$

$$\begin{cases} B_m\left(q^{-1}\right) = b_{m1}q^{-1} + b_{m2}q^{-2} + \ldots + b_{ms}q^{-s} \\ A_m\left(q^{-1}\right) = 1 + a_{m1}q^{-1} + \ldots + a_{mr}q^{-r} \end{cases}$$

C8: Implementarea algoritmilor de reglare.

2. Modele, filtre, implementare

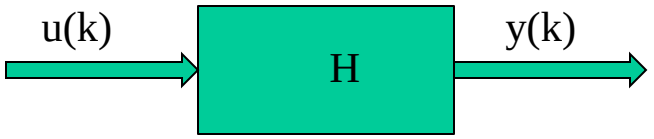
O solutie practica de proiectare a unui astfel de filtru este discretizarea (prin metoda dreptunghiului sau a trapezului) unui sistem continuu de ordinul II (sau I)

$$H_m(s) = \frac{k}{Ts+1} \qquad H_m(s) = \frac{k\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2}$$

$$q^{-1} = operatorul \text{ de intarziere cu un pas}$$

$$q^{-1}y(k) = y(k-1)$$

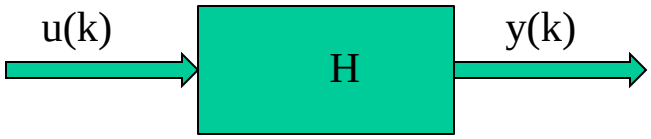
$$H_m\left(q^{-1}\right) = \frac{b_{m0} + b_{m1}q^{-1} + b_{m2}q^{-2}}{a_{m0} + a_{m1}q^{-1} + a_{m2}q^{-2}}$$



C8: Implementarea algoritmilor de reglare.

2. Modele, filtre, implementare

Implementare:



$$H_m(q^{-1}) = \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{a_0 + a_1 q^{-1} + a_2 q^{-2}} = \frac{y(k)}{u(k)}$$

$$y(k) = \frac{1}{a_0}(-a_1 y(k) q^{-1} - a_2 y(k) q^{-2} + b_0 u(k) + b_1 u(k) q^{-1} + b_2 u(k) q^{-2})$$

$$y(k) = \frac{1}{a_0}(-a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2))$$

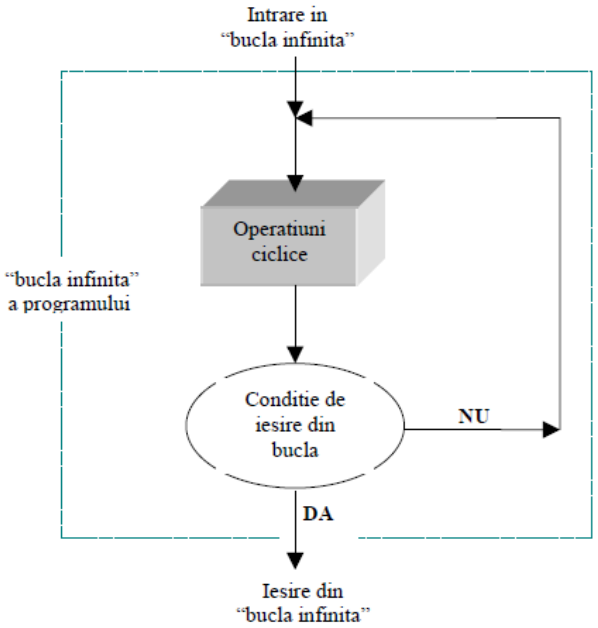
C8: Implementarea algoritmilor de reglare.

2. Modele, filtre, implementare

Implementare:

Pasi ai “buclei infinite”:

- Achizitie date
- Calcul comanda
- Trimitere comanda
- Afisare date
- Actualizare memorie algoritm



C8: Implementarea algoritmilor de reglare.

2. Modele, filtre, implementare

Implementare:

$$y(k) = \frac{1}{a_0}(-a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2))$$

!!!!!!!!!!!!!!!!!!!!!!!

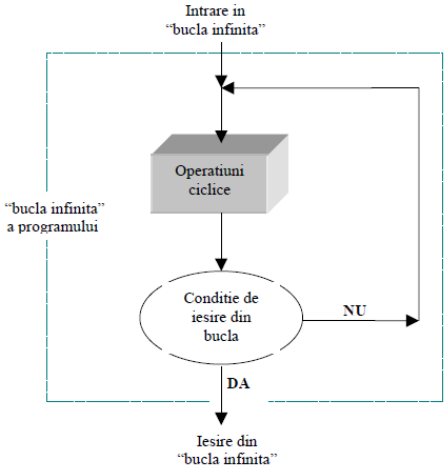
Actualizare memorie:

y(k-2)=y(k-1); // yk_2 = yk_1

y(k-1)=y(k); // yk_1 = yk

u(k-2)=u(k-1); //uk_2 = uk_1

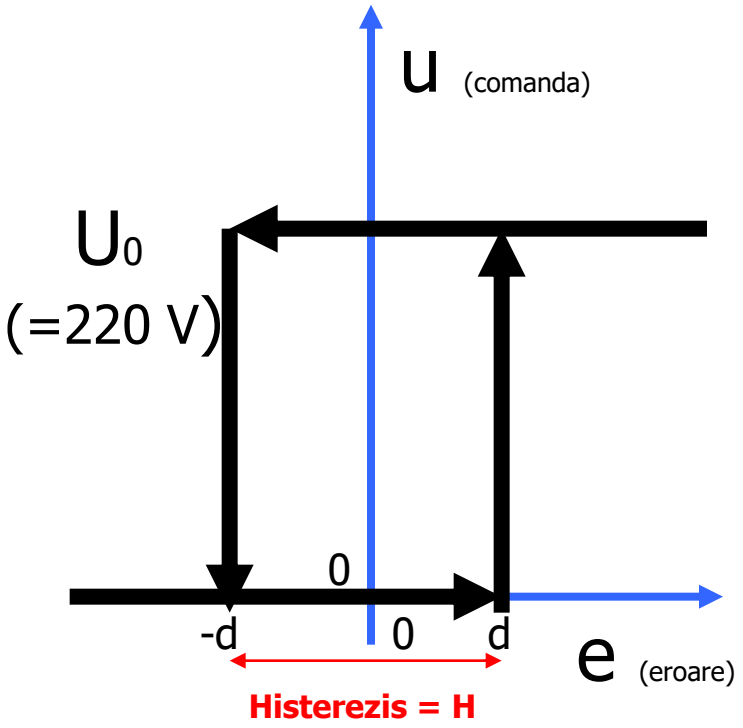
u(k-1)=u(k); // uk_1 = uk



C8: Implementarea algoritmilor de reglare.

3. Algoritmi de reglare bi-pozitionali

Reprezentarea “schematica” a algoritmului bi-pozitional si posibila sa implementare software.



$$H = 2 * d$$

$$e = VS - VP \quad (=r-y)$$

$$u = \begin{cases} U_0 & \text{pentru } e >= d \\ 0 & \text{pentru } e <= -d \\ u & \text{pentru } |e| < d \end{cases}$$

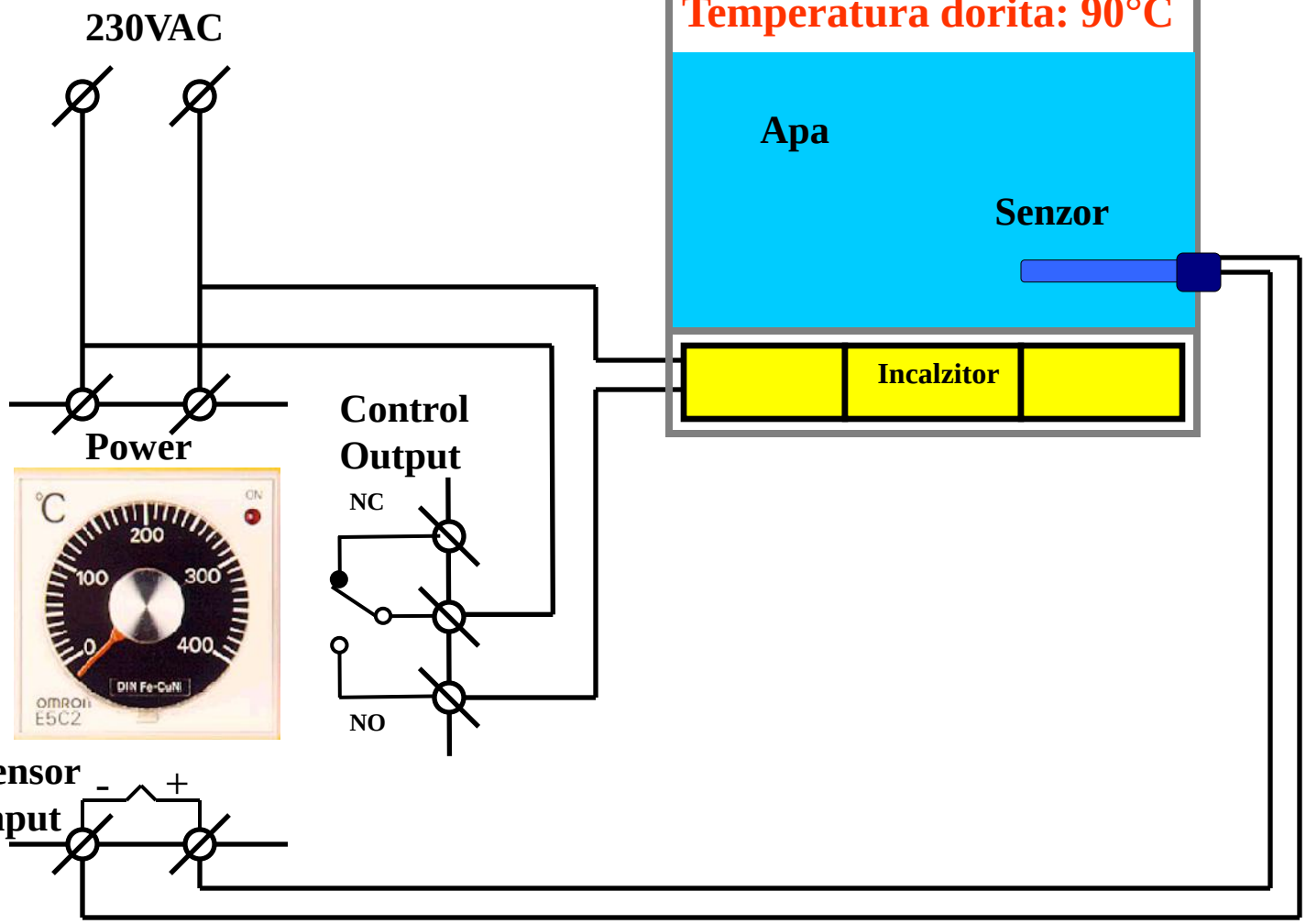
Reprezinta o “baza” pentru scrierea unui algoritm software de reglare bi-pozitionala intr-un PLC.

C8: Implementarea algoritmilor de reglare.

3. Algoritmi de reglare bi-pozitionali

Ex.: Obiectiv: incalzire si mentinere a temperaturii la 90°C.

Conexiuni:



1) Regulatorul trebuie conectat la tensiunea de alimentare.

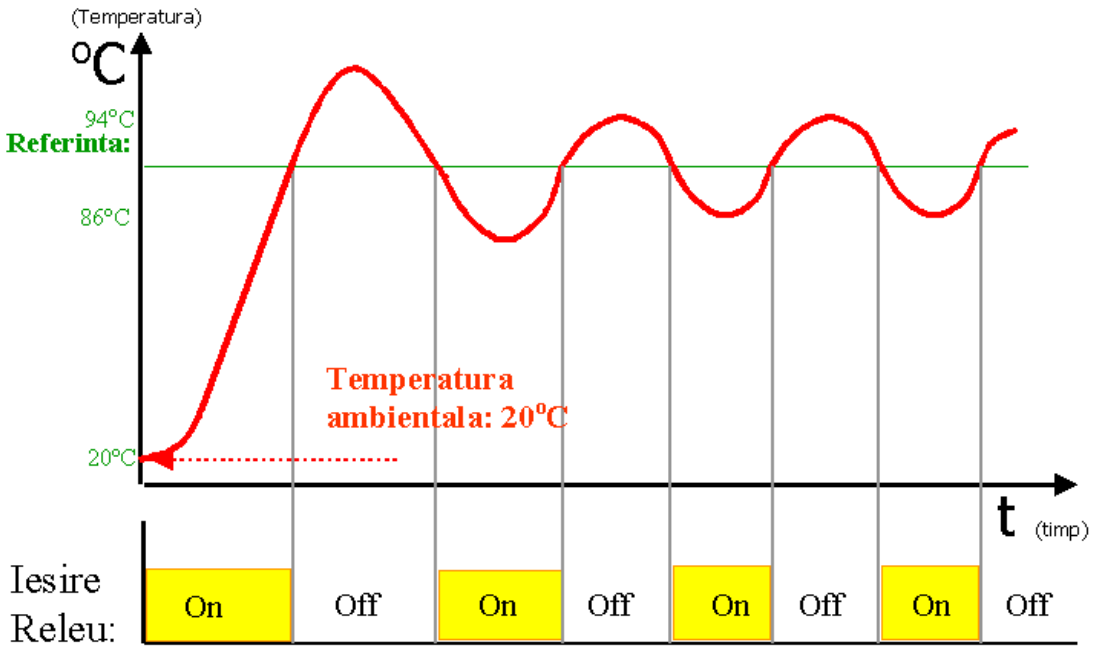
2) Incalzitorul trebuie conectat si el la tensiunea de alimentare.

3) Traductorul (senzorul) trebuie conectat la intrarea corespunzatoare a regulatorului.

C8: Implementarea algoritmilor de reglare.

3. Algoritmi de reglare bi-pozitionali

Evolutie proces si comanda:



C8: Implementarea algoritmilor de reglare.

4. Algoritmi de reglare tri-pozitionali

Reprezentarea “schematica” a algoritmului tri-pozitional.

Termeni:

e - eroare (=r-y)

u - comanda

2a - zona de insensibilitate

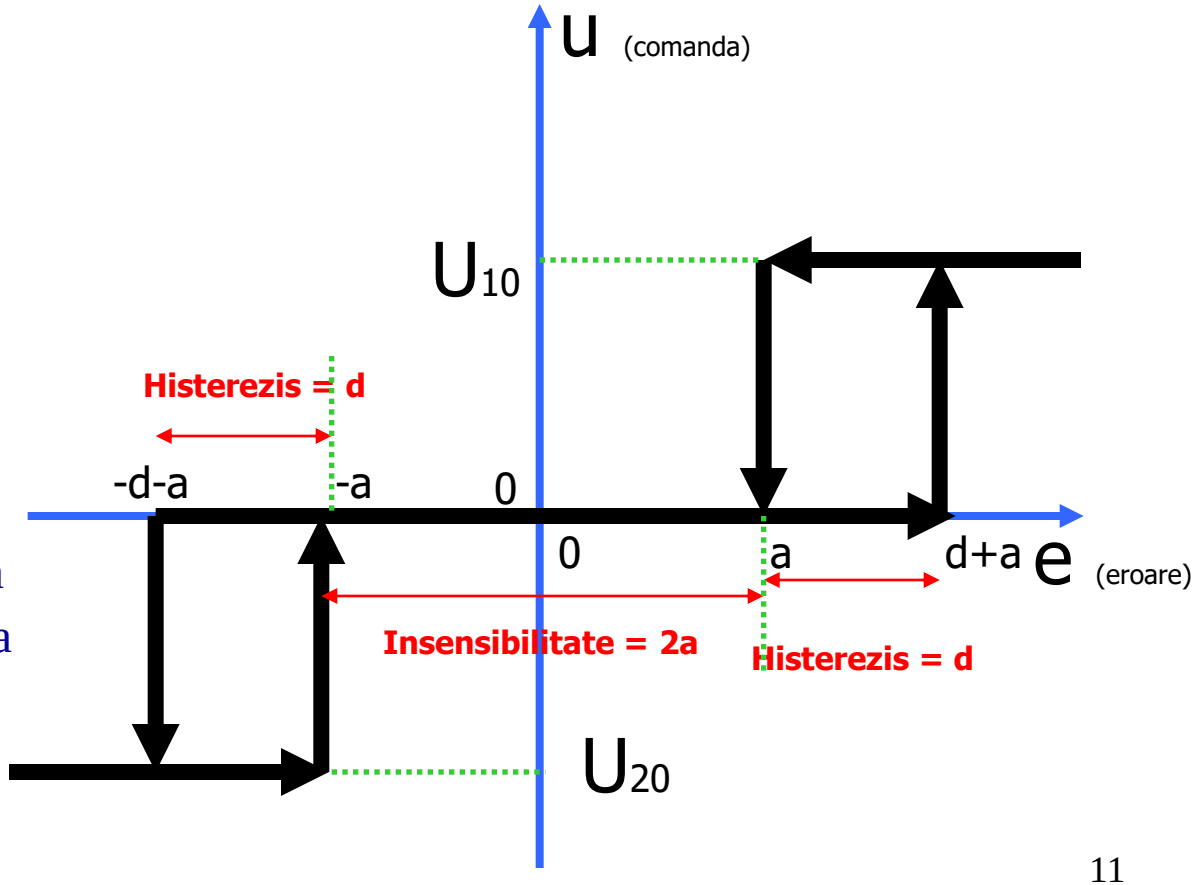
d - zona histerezis (x2)

U10 - comanda eroare pozitiva

U20 - comanda eroare negativa

Uzual:

U20 = -U10



C8: Implementarea algoritmilor de reglare.

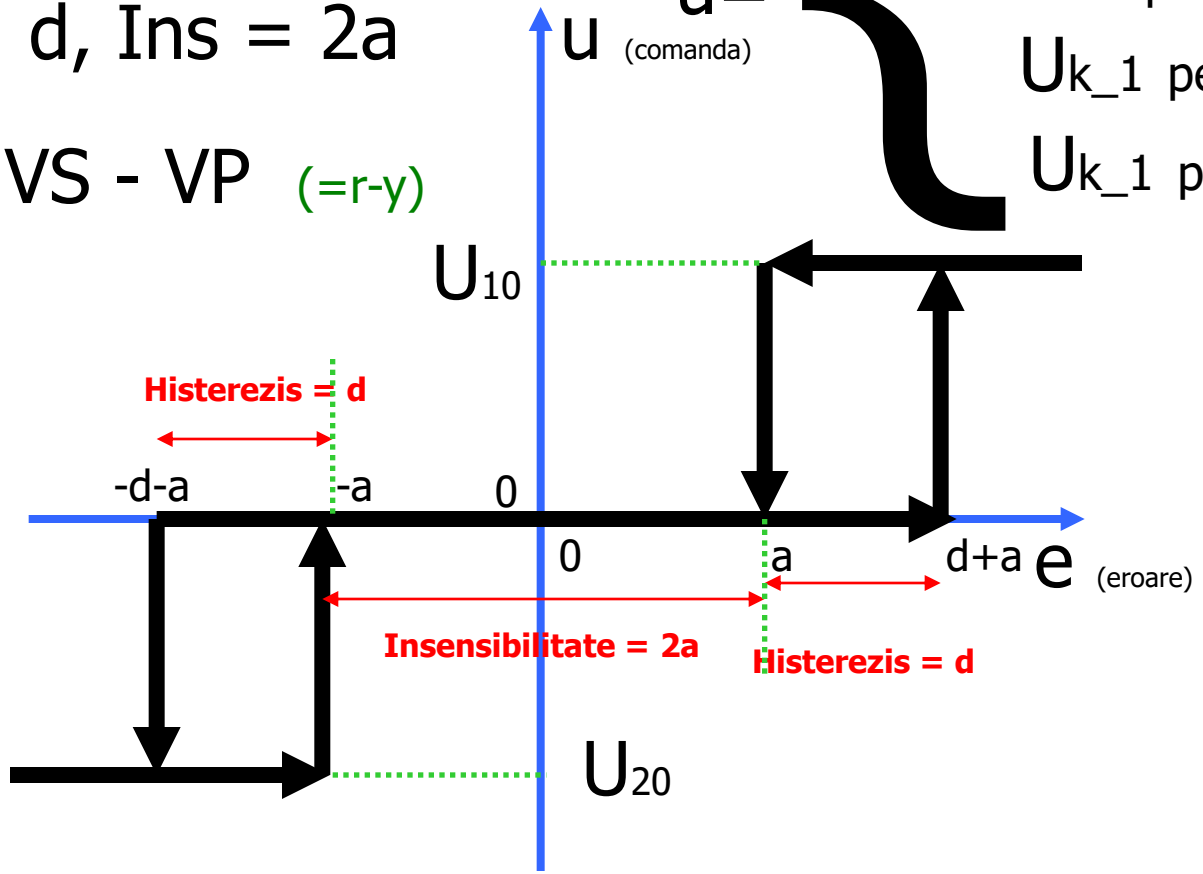
4. Algoritmi de reglare tri-pozitionali

Reprezentarea "schematica" a algoritmului tri-pozitional si posibila sa implementare software.

$$u = \begin{cases} U1_0 \text{ pentru } e >= a+d \\ U2_0 \text{ pentru } e <= -a-d \\ 0 \text{ pentru } -a <= e <= a \\ U_{k\_1} \text{ pentru } a < e < a+d \\ U_{k\_1} \text{ pentru } -a-d < e < -a \end{cases}$$

H= d, Ins = 2a

e= VS - VP (=r-y)



C8: Implementarea algoritmilor de reglare.

4. Algoritmi de reglare tri-pozitionali

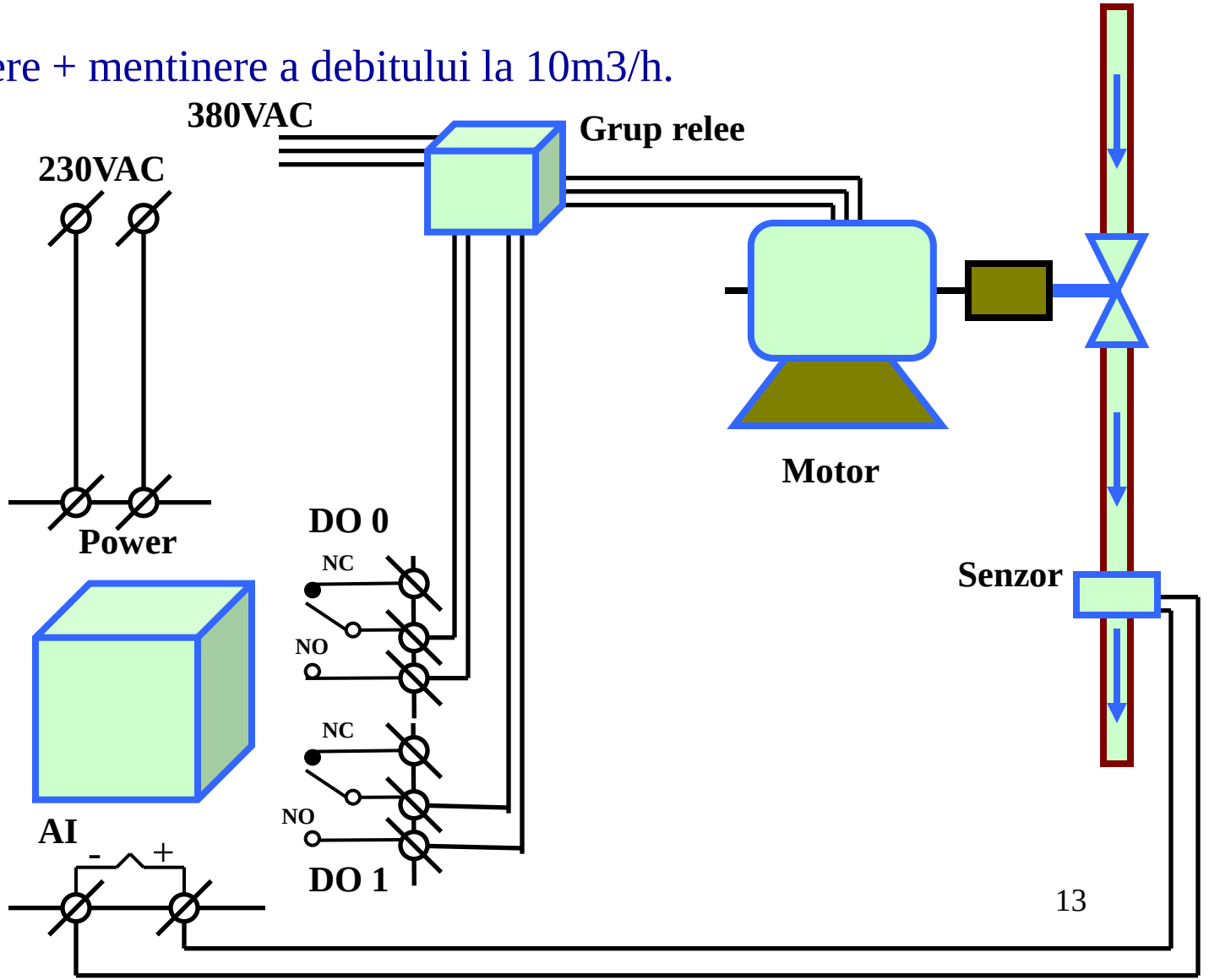
Obiectiv: Deschidere + mentinere a debitului la 10m3/h.

Conexiuni:

1) Automatul trebuie conectat la tensiunea de alimentare.

2) Motorul trebuie conectat si el la tensiunea de alimentare 380V.

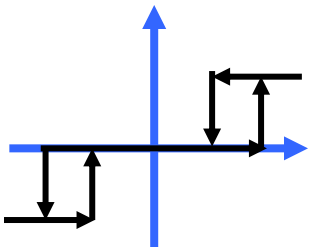
3) Traductorul (senzorul) trebuie conectat la intrarea corespunzatoare a AP.



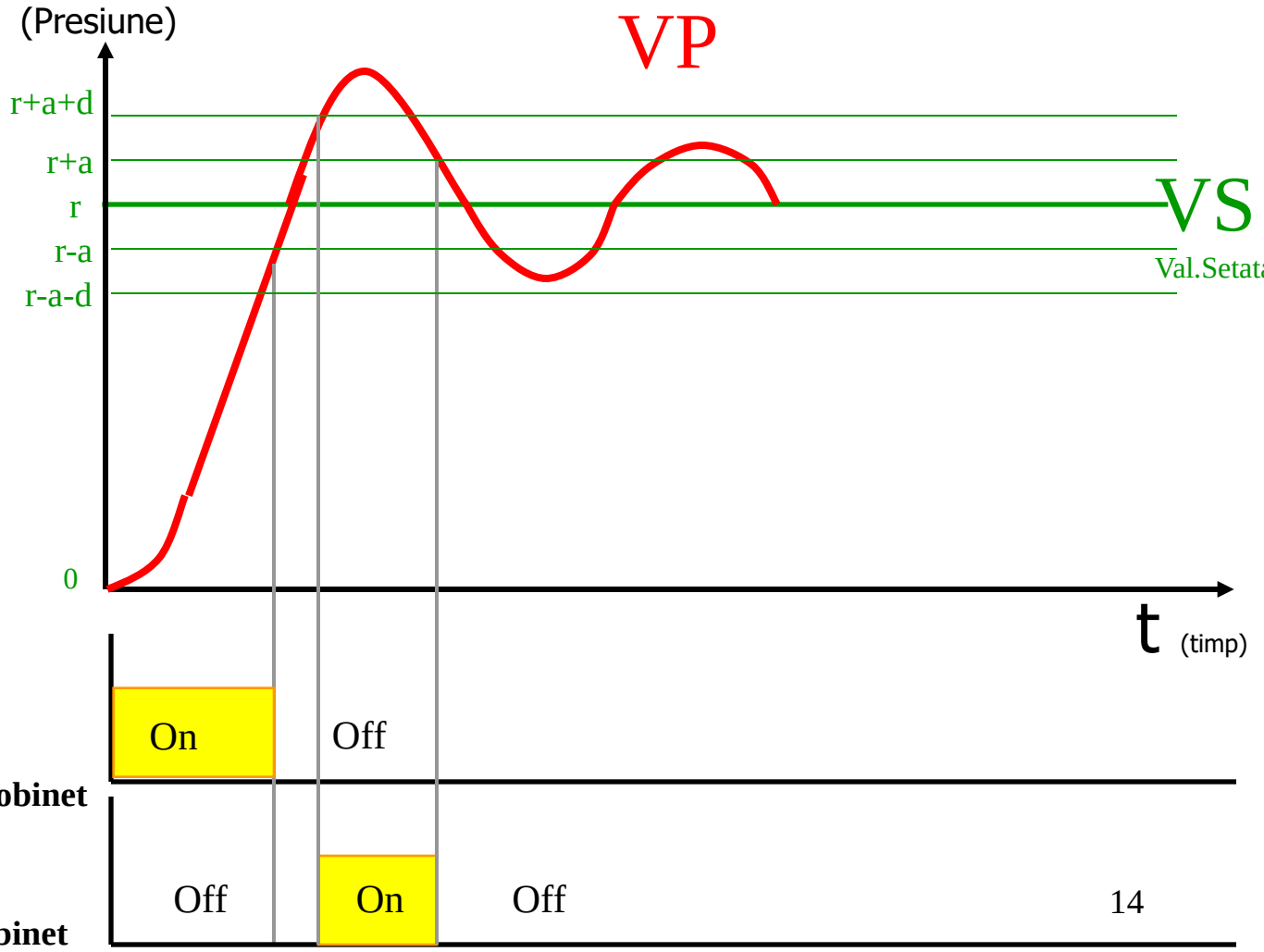
C8: Implementarea algoritmilor de reglare.

4. Algoritmi de reglare tri-pozitionali

Functionarea reala a unui regulator tri-pozitional.



Se observa cele trei pozitii



C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PID pozitional):

T – perioada de esantionare;
u_0 – comanda initiala fata de care se calculeaza u_k;
T_I – constanda componentei integratoare;
T_d – constanda componentei derivative;
K_r – constanda componentei proportionale.

$$u_k = K_R \cdot \left[\varepsilon_k + \frac{T}{T_i} \sum_{i=0}^{k} \varepsilon_i + \frac{T_d}{T} \left(\varepsilon_k - \varepsilon_{k-1} \right) \right] + u_0$$

C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PID pozitional):

$$u_k = K_R \cdot \left[\varepsilon_k + \frac{T}{T_i} \sum_{i=0}^{k} \varepsilon_i + \frac{T_d}{T} \left(\varepsilon_k - \varepsilon_{k-1} \right) \right] + u_0$$

$$u_{k-1} = K_R \cdot \left[\varepsilon_{k-1} + \frac{T}{T_i} \sum_{i=0}^{k-1} \varepsilon_i + \frac{T_d}{T} \left(\varepsilon_{k-1} - \varepsilon_{k-2} \right) \right] + u_0$$

$$u_k - u_{k-1} = K_R \cdot \left[1 + \frac{T}{T_i} + \frac{T_d}{T} \right] \cdot \varepsilon_k - \left[K_R + 2K_R \frac{T_d}{T} \right] \cdot \varepsilon_{k-1} + K_R \frac{T}{T_i} \cdot \varepsilon_{k-2}$$

C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PID pozitional -> incremental):

$$u_k = u_{k-1} + q_0 \cdot \varepsilon_k + q_1 \cdot \varepsilon_{k-1} + q_2 \cdot \varepsilon_{k-2}$$

$$q_0 = K_R \left[1 + \frac{T}{T_i} + \frac{T_d}{T} \right]$$

$$q_1 = -K_R \left[1 + 2\frac{T_d}{T} \right]$$

$$q_2 = K_R \frac{T}{T_i}$$

!!!!!!!!!!!!!!!!!!!!!!

Actualizare memorie:

e(k-2)=e(k-1);

e(k-1)=e(k);

u(k-1)=u(k);

C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Ex.: implementare PID:

```
#include <stdio.h>
#include <dos.h>
#include <time.h>
#define CLK             0x1Ch
#define PORT            220h
void interrupt (*oldvect)(void);
int contor = 0;
int canal = 0;
float Te = 1.0;
float Kr = 1.0;
float Ti = 50.0;
float Td = 10.0;
float yk = 0.0;
float rk = 50.0;
float uk = 0.0,uk_1 = 0.0;
float ek = 0.0,ek_1 = 0.0,ek_2 = 0.0;
float q0 = 0.0,q1 = 0.0,q2 = 0.0;
void interrupt planific(void)
{
    contor++;
}
float ai(int can);
void ao(float valcom);

void main(void)
{
        oldvect = getvect(CLK);
        setvect(CLK, planific);
        canal = 0;
        q0 = Kr*(1+Te/Ti+Td/Te);
        q1 = -Kr*(1+2*Td/Te);
        q2 = Kr*Te/Ti;
```

C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Ex.: implementare PID:

```
while(!kbhit())
{
        if(contor >= 18)
        {
                contor = 0;
                yk = ai(canal);
        ek = rk-yk;
        uk = uk_1+q0*ek+q1*ek_1+q2*ek_2;
        ao(yk);
        // actualizare variabile
        uk_1 = uk;
        ek_2 = ek_1;
        ek_1 = ek;
        // afisare
        gotoxy(2,2);
        printf("rk = %6.2f  yk = %6.2f  uk =
                %6.2f", rk, yk, uk);
        }// enf if
} // end while
        setvect(CLK, oldvect);
} // end main
float ai(int can)
{
        int val_ucan = 0;
        float y_proc = 0.0;
        ……..
        y_proc =
        =(val_ucan/4095.0)*100.0;
        return y_proc;
}
void ao(float u_proc)
{
        int u_can = 0;
        u_can =     (u_proc/100)*4095;
        …….
}
```

C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PIDf):

$$H_R(s) = K_R \cdot \left[1 + \frac{1}{T_i s}\right] \frac{T_d s + 1}{\alpha T_d s + 1}$$

Discretizare (Te):

$$\frac{1}{s} \sim \frac{T}{1 - z^{-1}}$$ - metoda dreptunghiului;

$$\frac{1}{s} \sim \frac{T}{2} \frac{1 + z^{-1}}{1 - z^{-1}}$$ - metoda trapezelor.

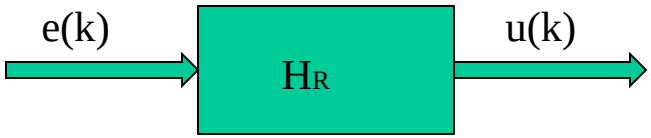
C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PIDf):

$$H_R\left(z^{-1}\right) = K_R \cdot \left[1 + \frac{1}{T_i}\frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}\right]\left[\frac{T_d + \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}}{\alpha T_d + \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}}\right]$$

$$H_R\left(z^{-1}\right) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 + p_1 z^{-1} + p_2 z^{-2}} \qquad \text{Sau: !!!} \quad H_R(q^{-1}) = \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{a_0 + a_1 q^{-1} + a_2 q^{-2}} = \frac{y(k)}{e(k)}$$



C8: Implementarea algoritmilor de reglare.

5. Algoritmi de reglare PID

Prezentare (PIDf):

$$H_R\left(z^{-1}\right)=\frac{q_0+q_1z^{-1}+q_2z^{-2}}{1+p_1z^{-1}+p_2z^{-2}}$$

$$q_0=K_R\frac{\left(2T_i+T\right)\left(2T_d+T\right)}{2T_i\left(2\alpha T_d+T\right)}$$

$$q_1=\frac{\left(2T_i+T\right)\left(T-2T_d\right)+\left(2T_d+T\right)\left(T-2T_i\right)}{2T_i\left(2\alpha T_d+T\right)}$$

$$q_2=\frac{\left(T-2T_i\right)\left(T-2T_d\right)}{2T_i\left(2\alpha T_d+T\right)}$$

$$p_1=\frac{T-4\alpha T_d}{2\alpha T_d+T}$$

$$p_2=\frac{2\alpha T_d-T}{2\alpha T_d+T}$$

C8: Implementarea algoritmilor de reglare.

6. Algoritmi de reglare RST

Prezentare (PIDf – in varianta RST):

$$H_{PID}(s) \overset{def}{=} K_R \cdot \left[1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N} s + 1} \right]$$

Notam prin: K_R – castigul sau amplificarea regulatorului, T_i, T_d — constante pentru actiunea integrala, respectiv derivativa, N - constanta de filtraj. Intrarea in algoritmul de comanda este eroarea de reglare, adica diferenta dintre marimea de referinta si marimea de iesire reglata, iar marimea de la iesirea din regulator este comanda numerica $u(k)$.

C8: Implementarea algoritmilor de reglare.

6. Algoritmi de reglare RST

Prezentare (PIDf – in varianta RST):

$$s \approx \frac{1-q^{-1}}{T_e}$$

$$H_{PID}(q^{-1}) = K_R \left[1 + \frac{\frac{T_e}{T_i}}{1-q^{-1}} + \frac{\frac{NT_d}{T_d+NT_e}(1-q^{-1})}{1-\frac{T_d}{T_d+NT_e}q^{-1}} \right] = \frac{R_0(q^{-1})}{S_0(q^{-1})}$$

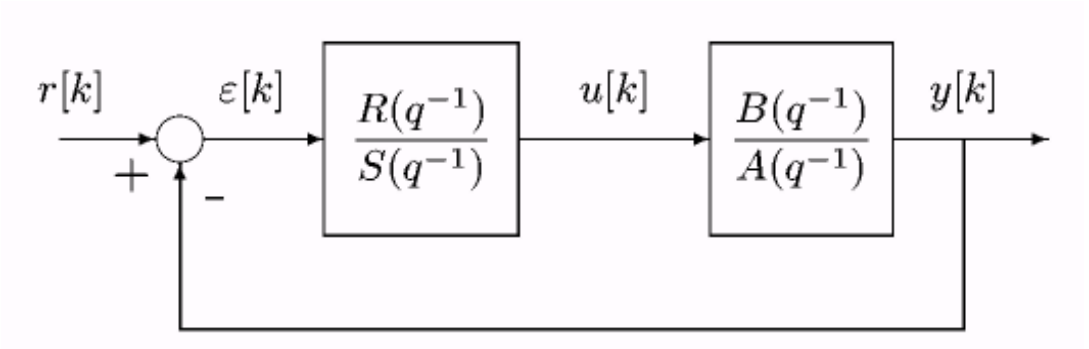
$$\begin{cases} R_0(q^{-1}) = K_R(1-q^{-1})\left(1-\frac{T_d}{T_d+NT_e}q^{-1}\right) + \frac{K_R T_e}{T_i}\left(1-\frac{T_d}{T_d+NT_e}q^{-1}\right) + \frac{K_R NT_d}{T_d+NT_e}(1-q^{-1})^2 \\ S_0(q^{-1}) = (1-q^{-1})\left(1-\frac{T_d}{T_d+NT_e}q^{-1}\right) \end{cases}$$

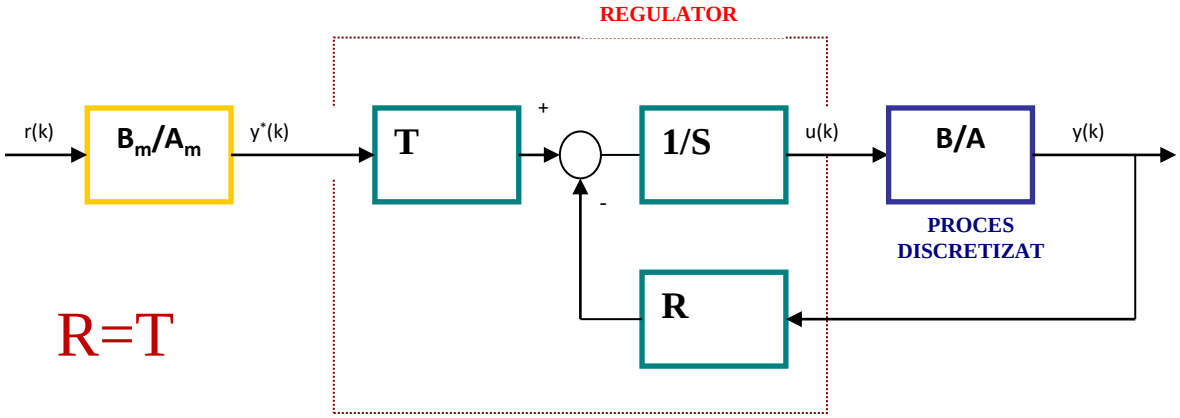
C8: Implementarea algoritmilor de reglare.

6. Algoritmi de reglare RST

Prezentare (PIDf – in varianta RST):

$$\begin{cases} R\left(q^{-1}\right) = T\left(q^{-1}\right) = r_0 + r_1 q^{-1} + r_2 q^{-2} \equiv R_0\left(q^{-1}\right) \\ S\left(q^{-1}\right) = \left(1 - q^{-1}\right)\left(1 + s_1 q^{-1}\right) \equiv S_0\left(q^{-1}\right) \end{cases}$$



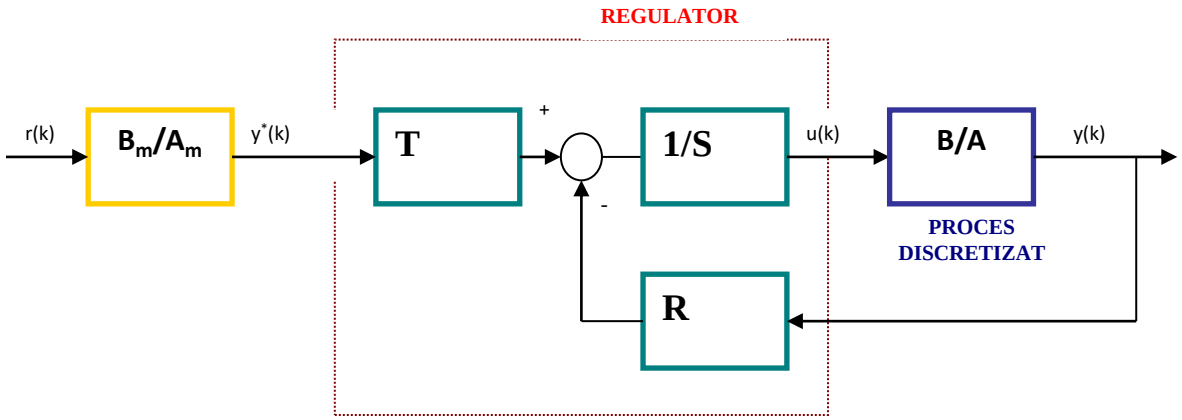


R=T

C8: Implementarea algoritmilor de reglare.

6. Algoritmi de reglare RST

Prezentare (RST - general):



$$S(q^{-1}) = s_0 + s_1 q^{-1} + \ldots + s_{n_S} q^{-n_S}$$

$$R(q^{-1}) = r_0 + r_1 q^{-1} + \ldots + r_{n_R} q^{-n_R}$$

$$T(q^{-1}) = t_0 + t_1 q^{-1} + \ldots + t_{n_T} q^{-n_T}$$

$$S(q^{-1})u(k) + R(q^{-1})y(k) = T(q^{-1})y^*(k)$$

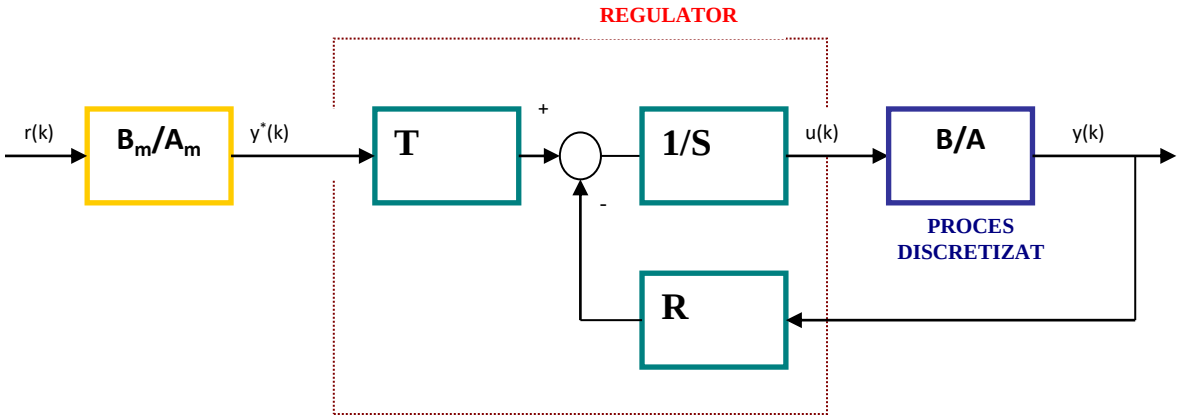
$$\sum_{i=0}^{n_S} s_i u(k-i) + \sum_{i=0}^{n_R} r_i y(k-i) = \sum_{i=0}^{n_T} t_i y^*(k-i)$$

$$u(k) = \frac{1}{s_0}\left[-\sum_{i=1}^{n_S} s_i u(k-i) - \sum_{i=0}^{n_R} r_i y(k-i) + \sum_{i=0}^{n_T} t_i y^*(k-i) \right]$$

C8: Implementarea algoritmilor de reglare.

6. Algoritmi de reglare RST

Prezentare (RST - general):



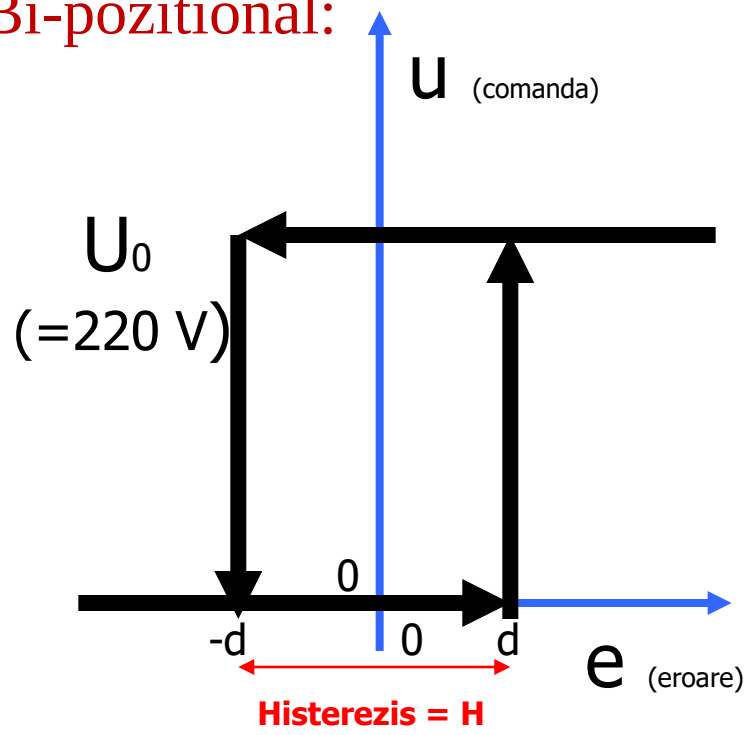
Patricularizand, pentru un algoritm de reglare fara generator de referita ($y^*(k)=r(k)$), in care $n_R = n_S = n_T = 2$ generarea comenzii se face:

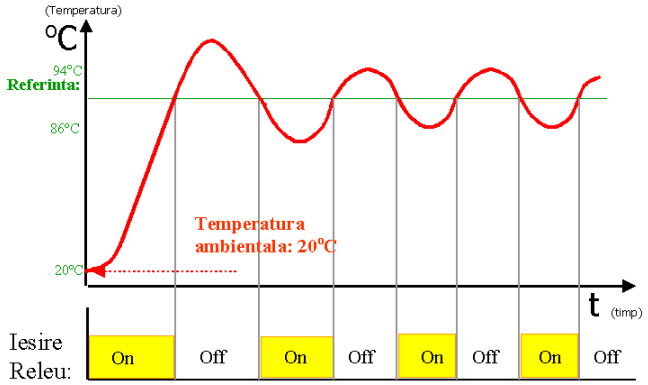
$$u(k)=\frac{1}{s_0}\left(-s_1u(k-1)-s_2u(k-2)-r_0y(k)-r_1y(k-1)-r_2y(k-2)+t_0y^*(k)+t_1y^*(k-1)+t_2y^*(k-2)\right)$$

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

Bi-pozitional:



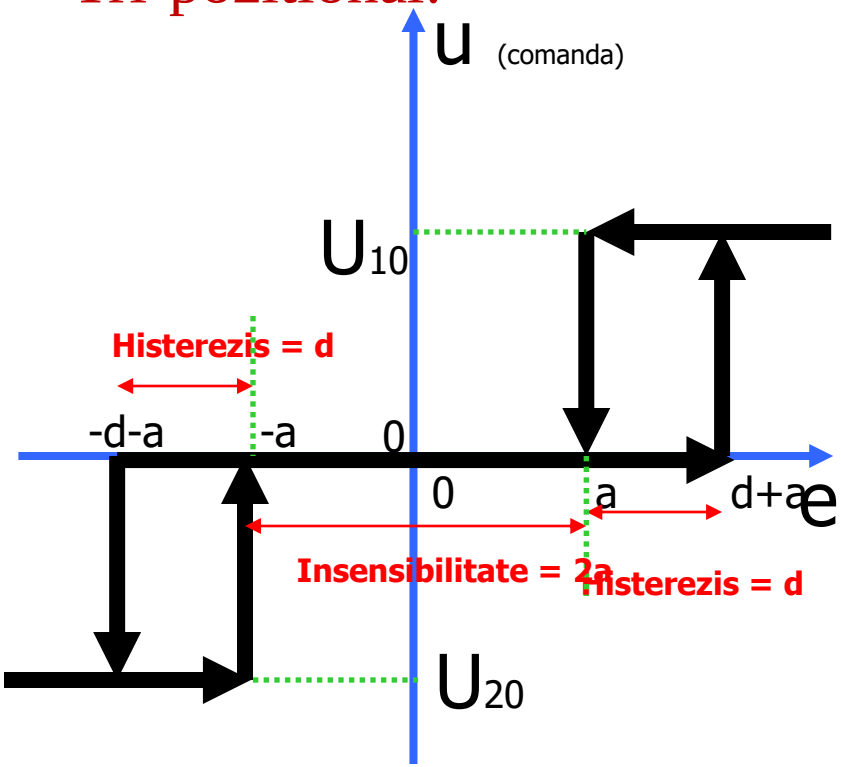


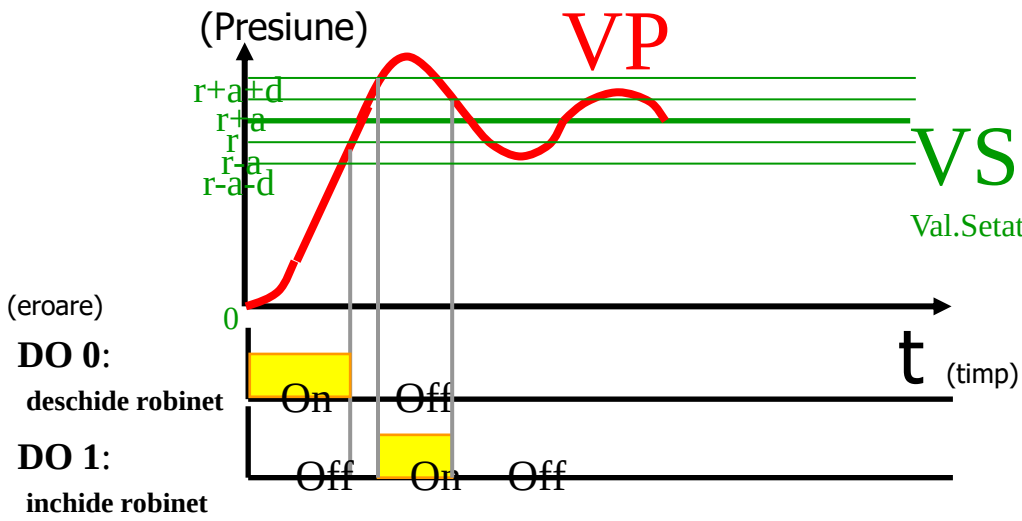
Histerezisul se va dimensiona in functie de constantele de timp ale procesului si timpul mort.

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

Tri-pozitional:





Histerezisul si zona de insensibilitate se vor dimensiona in functie de constantele de timp ale procesului si timpul mort.

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

PID – metode experimentale:

Metode bazate pe raspunsul indicial al procesului.

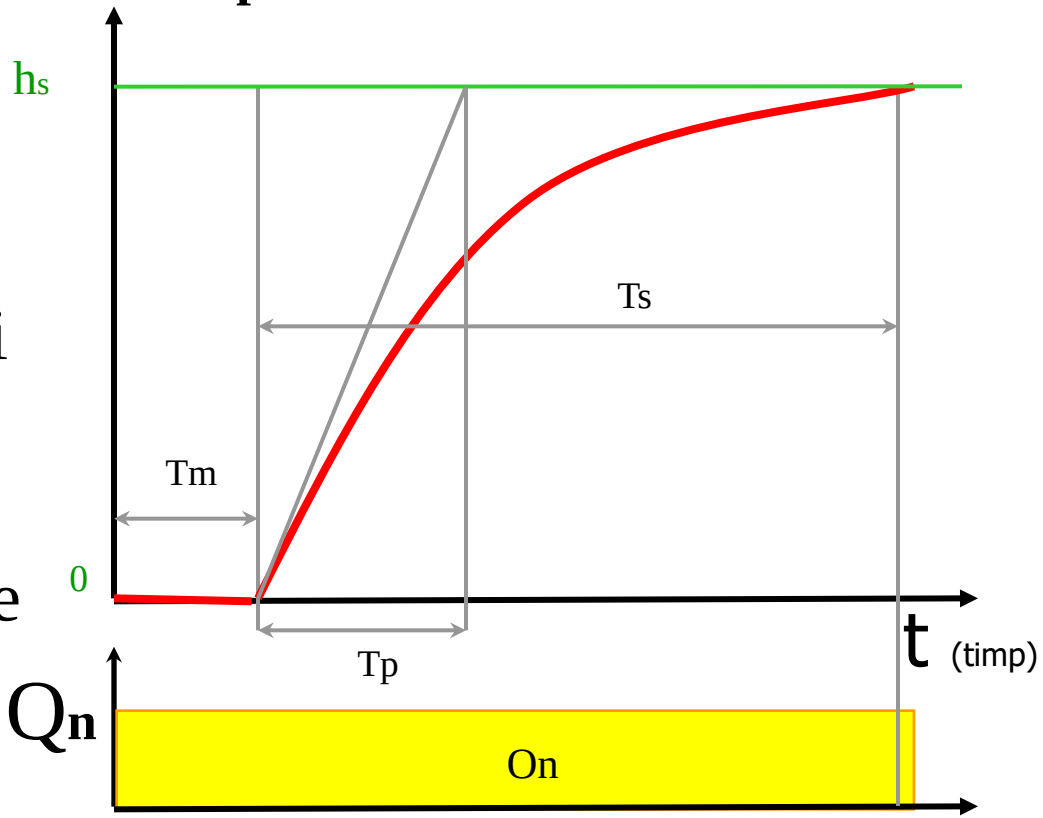
Aplicarea in proces a unei intrari de tip treapta si determinarea aproximativa a parametrilor:

T_p – ct. de timp procesului

T_m – valoare timp mort

T_s – valoare timp stabilizare

h_s – valoarea de stabilizare



Raspuns experimental la intrare treapta:

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

PID – metode experimentale:

Metode bazate pe raspunsul indicial al procesului (Ziegler-Nichols).

Calculul parametrilor auxiliati:

K_p – amplificarea procesului $\quad K_p = \dfrac{h_s}{val.\Delta u}$

a – param. auxiliar $\quad a = K_p \dfrac{T_m}{T_f}$

Calculul parametrilor regulatorului:

P: $K_r = 1/a$

PI: $K_r = 0.9a$ $\quad T_i = 3T_m$

PID: $K_r = 1.2a$ $\quad T_i = 2T_m$ $\quad T_d = \dfrac{T_m}{2}$

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

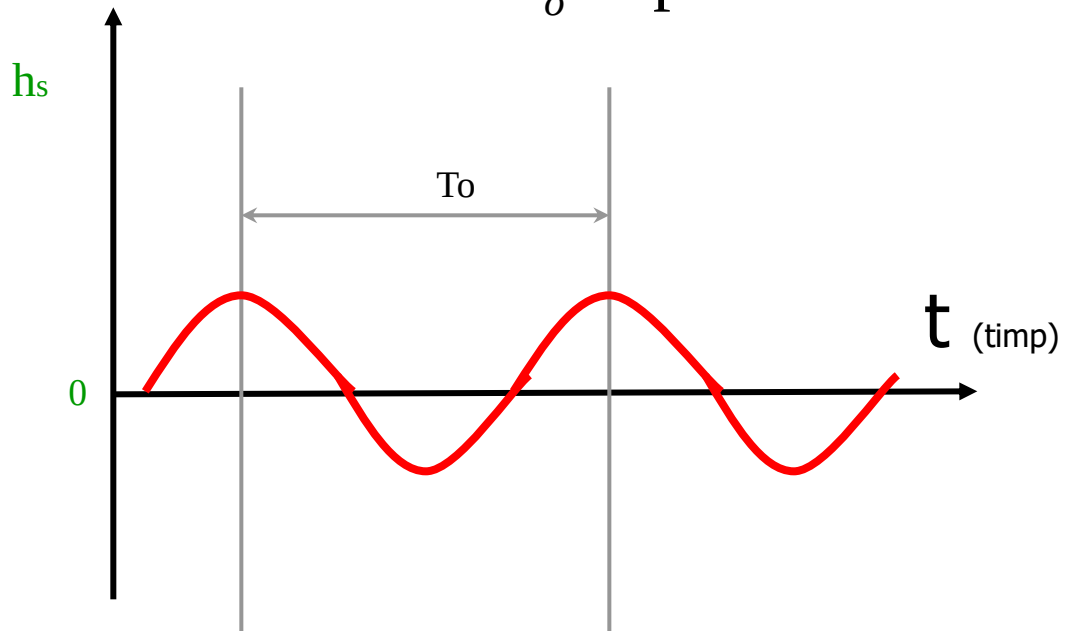
PID – metode experimentale:

Metode bazate pe limita de stabilitate.

Regulatorul PID doar cu componenta P activa: crescut Kr treptat pana cand procesul ajunge la o oscilatie neamortizata.

$K_o = Kr$ – amp. de oscilatie

T_o – perioada oscilatiei



C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

PID – metode experimentale:

Metode bazate pe limita de stabilitate (Ziegler-Nichols).

Calculul parametrilor regulatorului:

P:

$$K_r = 0.5K_o$$

PI:

$$K_r = 0.4K_o$$
$$T_i = 0.8T_o$$

PID:

$$K_r = 0.6K_o$$
$$T_i = 0.5T_o$$
$$T_d = 0.12T_o$$

C8: Implementarea algoritmilor de reglare.

7. (Auto)acordarea algoritmilor de reglare

RST (si PID) – metode exacte:

Metode bazate pe identificarea modelului procesului.

- Obtinerea datelor din proces;
- Identificarea modelului procesului;
- Validarea modelului;
- Calculul parametrilor regulatorului (alocare de poli etc.):
- … implementare.

C8: Implementarea algoritmilor de reglare.

8. Arhitecturi hardware/software de TR

Implementare in TR:

- Leader;
- FBD;
- ST;
- C (TR);
- ASM;
- LabView;
- Simulink;