MODELAREA ORIENTATA PE OBIECTE A UNEI APLICATII PENTRU ANALIZA UNUI SISTEM DE REGLARE AUTOMATA A NIVELULUI

Proiect realizat de Tarabin Mohammed-Bashar

Profesor coordonator : Sef lucr. dr. mat. Madalina Carbureanu

Grupa 10213

Anul II

Specializarea LCALZ Facultatea de Inginerie Mecanica si Electrica

Universitatea Petrol-Gaze din Ploiesti

CUPRINS

1. Introducere :

Sistemul de reglare a nivelului este foarte des utilizat în instalațiile chimice , Cele mai răspândite sunt bazate pe modificarea debitelor intrare/ieșire.

Sistemele sunt simple, traductoarele de nivel utilizate pentru procesele care operează sub presiune sunt cele cu imersor.

Cerințele de reglare ale proceselor chimice au impus utilizarea fluxului de ieșire asociat unui sistem cu acumulare pentru alt sistem automat (de exemplu sistemul de reglare a concentrației).

In acest caz reglarea nivelului poate utiliza facilitățile unui vaporizator sau condensator.

Astfel a fost elaborat sistemul de reglare a nivelului bazat pe vaporizarea fluidului.

Reglarea concentrației reprezintă o provocare în cadrul sistemelor automate. Cea mai mare problemă o reprezintă traductorul, care este specific tipului de substanță analizată.

În aceste condiții, numărul mare de tipuri de traductoare, specificitatea acestora și prețul de achiziție sunt probleme deosebite în luarea deciziei de implementare a sistemelor de reglare a concentrației.

În cele ce urmează sunt prezentate doua exemple de sisteme de reglare a concentrației, unul destinat coloanelor de fracționare și altul procesului de epurare a apelor reziduale.

Un sistem de reglare automată a nivelului este un sistem mecanic sau electronic utilizat pentru a menține un nivel constant al unui fluid sau a unui material într-un recipient sau într-un sistem de conducte. Acest sistem poate funcționa prin intermediul unui senzor care măsoară nivelul actual și compară cu nivelul dorit, iar dacă există o diferență, acționează asupra unui actuator care adaugă sau scoate fluidul sau materialul din recipient.

Există mai multe tipuri de sisteme de reglare automată a nivelului, inclusiv cele care folosesc senzori cu afișaj electronic sau mecanice pentru a afișa nivelul, sau cele care utilizează controllere programabile pentru a procesa informațiile de la senzori și a controla actuatorii. Unele dintre aceste sisteme pot fi chiar dotate cu funcții de alarmă care sunt activate atunci când nivelul devine prea mare sau prea mic.

Acest sistem poate fi utilizat în diverse industrii, cum ar fi cea petrochimică, a produselor alimentare și a băuturilor, pentru a menține nivelul constant al lichidelor în recipiente sau în conducte, sau in industria generatoarelor de electricitate , unde se folosesc sisteme de reglare automată a nivelului pentru a menține nivelul apei în rezervoarele de acumulare.

In concluzie, un sistem de reglare automată a nivelului este o componentă importantă în multe sisteme industriale, care permite menținerea unui nivel constant al unui fluid sau a unui material într-un recipient sau într-un sistem de conducte, prin intermediul unui

senzor, care măsoară nivelul actual și compară cu nivelul dorit, iar dacă există o diferență, acționează asupra unui actuator care adaugă sau scoate fluidul sau materialul din recipient.

Există mai multe tipuri de senzori utilizați în sistemele de reglare automată a nivelului, cum ar fi senzorii optice, capacitivi, inductivi sau cu flotor. Fiecare dintre acestea are avantaje și dezavantaje specifice, iar alegerea celui potrivit depinde de caracteristicile fluidului sau materialului, precum și de mediul în care este utilizat.

Senzorii optice sunt adesea utilizați în sistemele de reglare automată a nivelului care se ocupă cu lichide transparente sau semi-transparente, cum ar fi apa sau uleiul. Acestea funcționează prin emiterea unei lumini în lichid și detectarea reflexiei acesteia. Senzorii capacitivi funcționează prin detectarea modificării capacității electrice dintre două electrozi în contact cu lichidul sau materialul, în timp ce senzorii inductivi funcționează prin detectarea variațiilor în fluxul magnetic generat de un conductor în contact cu lichidul sau materialul. Senzorii cu flotor funcționează prin detectarea poziției unui element care plutește pe suprafața lichidului sau materialului.

Actuatorii utilizați în sistemele de reglare automată a nivelului pot fi de mai multe tipuri, cum ar fi electromecanici, electrohidraulici sau pneumatice. Actuatorii electromecanici sunt adesea utilizați pentru controlul valvelor sau robinetelor, în timp ce actuatorii electrohidraulici sau pneumatice sunt utilizați pentru controlul aparatelor de ridicare sau al altor echipamente care implică mișcarea unui element prin intermediul unui cilindru sau alte dispozitive de

acționare.

Sistemele de reglare automată a nivelului pot fi utilizate pentru o varietate de aplicații, cum ar fi controlul nivelului apei în rezervoare, monitorizarea nivelului lichidelor în conducte sau recipiente, menținerea unui nivel constant al unei substanțe într-un proces industrial sau chiar controlul nivelului de materiale solide în containere sau silozuri.

Un sistem de reglare automată a nivelului poate fi integrat cu alte sisteme de control, cum ar fi sisteme de control al procesului sau sisteme de management.

de alarmare și supraveghere, pentru a oferi o monitorizare în timp real și o gestionare eficientă a nivelului. Aceste integrări pot include funcții precum raportarea nivelului în timp real către un sistem de management al procesului sau generarea de alarme în cazul în care nivelul depășește anumite limite prestabilite.

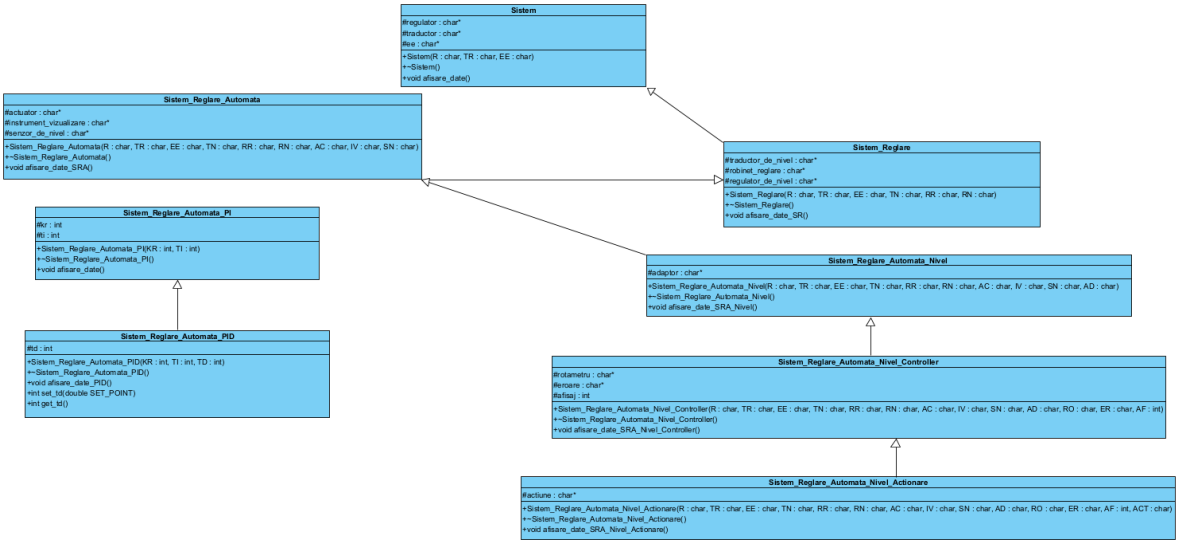
Sistemele de reglare automată a nivelului pot fi, de asemenea, configurate în mod flexibil, astfel încât să poată fi adaptate pentru a se potrivi cu diferitele cerințe de proces și medii de lucru. De exemplu, sistemele pot fi configurate pentru a funcționa în mod automat sau manual, sau pentru a accepta comenzi de la un operator sau de la un alt sistem de control.

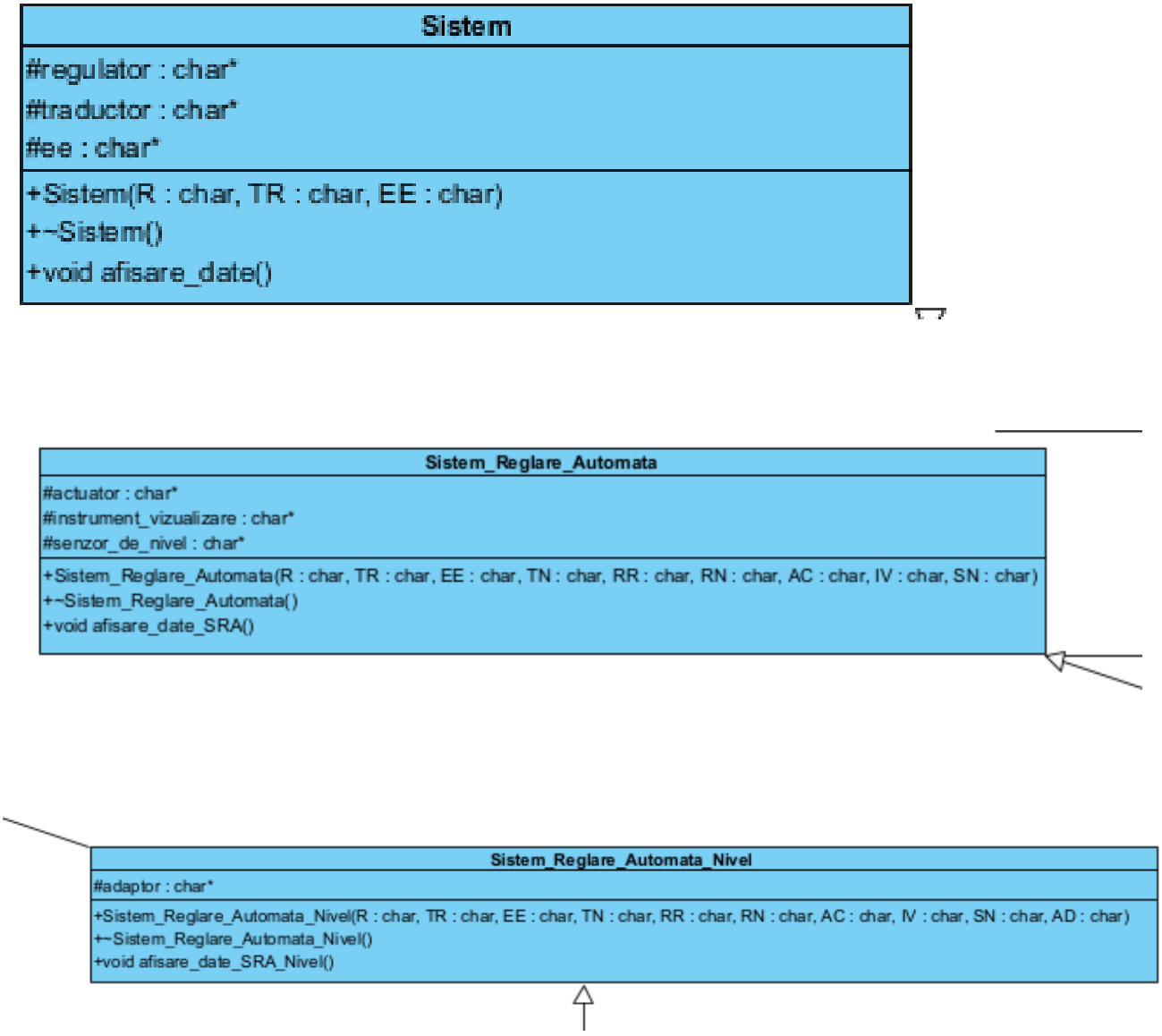
Pentru a asigura o funcționare fiabilă și sigură, sistemele de reglare automată a nivelului ar trebui să fie întreținute și verificate în mod regulat. Aceasta poate include verificarea senzorilor și a actuatorilor

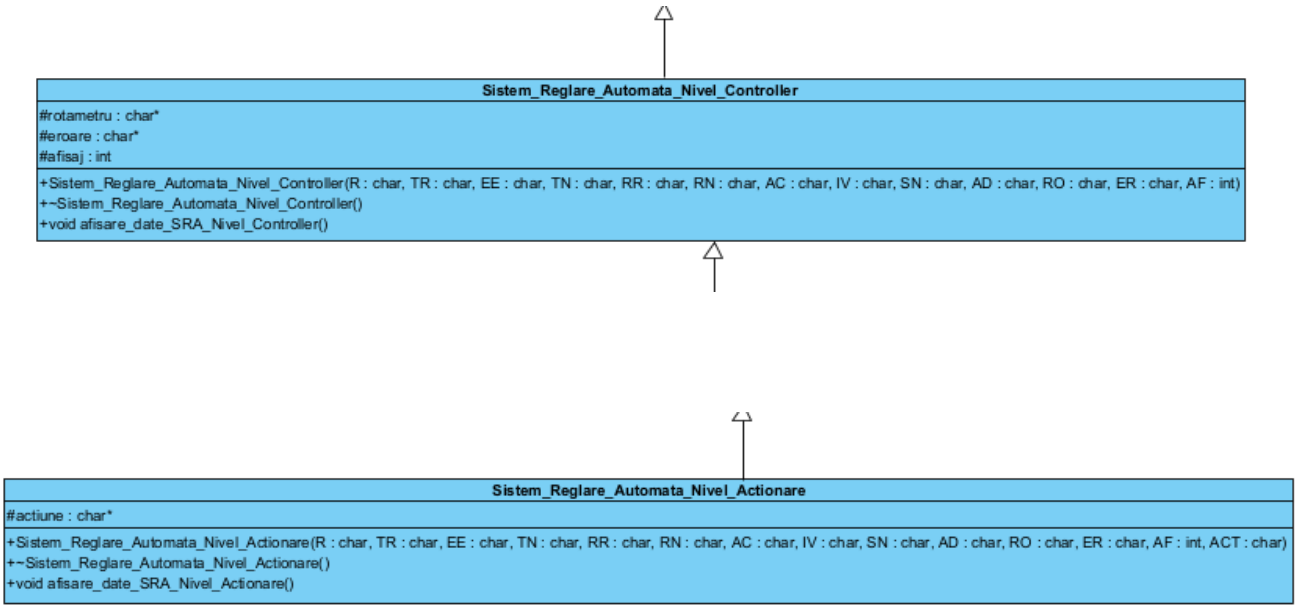
pentru a se asigura că funcționează corect, sau calibrarea acestora dacă este necesar. De asemenea, ar trebui efectuate teste de funcționare pentru a se asigura că sistemul răspunde corect la diferențele de nivel și că alarmele funcționează corect.

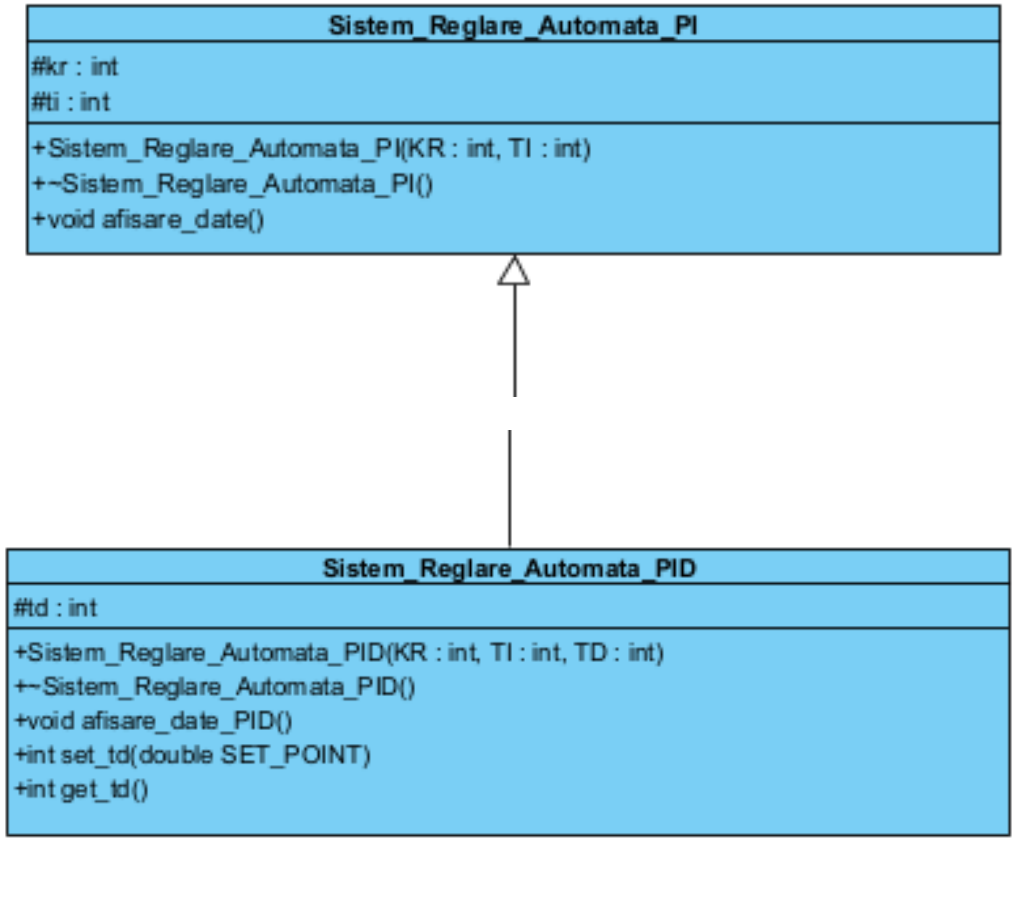
2. Modelul OMT :

Clasa si Diagrama Obiectelor




Sistem
#regulator : char*
#traductor : char*
#ee : char*
+Sistem(R : char, TR : char, EE : char)
+~Sistem()
+void afisare_date()
Sistem_Reglare_Automata
#actuator : char*
#instrument_vizualizare : char*
#senzor_de_nivel : char*
+Sistem_Reglare_Automata(R : char, TR : char, EE : char, TN : char, RR : char, RN : char, AC : char, IV : char, SN : char)
+~Sistem_Reglare_Automata()
+void afisare_date_SRA()
Sistem_Reglare_Automata_Nivel
#adaptor : char*
+Sistem_Reglare_Automata_Nivel(R : char, TR : char, EE : char, TN : char, RR : char, RN : char, AC : char, IV : char, SN : char, AD : char)
+~Sistem_Reglare_Automata_Nivel()
+void afisare_date_SRA_Nivel()


Sistem_Reglare_Automata_Nivel_Controller
#rotametru : char*
#eroare : char*
#afisaj : int
+Sistem_Reglare_Automata_Nivel_Controller(R : char, TR : char, EE : char, TN : char, RR : char, RN : char, AC : char, IV : char, SN : char, AD : char, RO : char, ER : char, AF : int)
+~Sistem_Reglare_Automata_Nivel_Controller()
+void afisare_date_SRA_Nivel_Controller()
Sistem_Reglare_Automata_Nivel_Actionare
#actiune : char*
+Sistem_Reglare_Automata_Nivel_Actionare(R : char, TR : char, EE : char, TN : char, RR : char, RN : char, AC : char, IV : char, SN : char, AD : char, RO : char, ER : char, AF : int, ACT : char)
+~Sistem_Reglare_Automata_Nivel_Actionare()
+void afisare_date_SRA_Nivel_Actionare()


Sistem_Reglare_Automata_PI
#kr : int
#ti : int
+Sistem_Reglare_Automata_PI(KR : int, TI : int)
+~Sistem_Reglare_Automata_PI()
+void afisare_date()
Sistem_Reglare_Automata_PID
#td : int
+Sistem_Reglare_Automata_PID(KR : int, TI : int, TD : int)
+~Sistem_Reglare_Automata_PID()
+void afisare_date_PID()
+int set_td(double SET_POINT)
+int get_td()

3. Codul sursa al aplicatiei :

Clasa de baza si alte caracteristici ale codului

```
#include <iostream>
#include <iomanip>
#include <string.h>
#include <conio.h>
#include <stdio.h>

using namespace std;



class Sistem
{
protected:
    char *regulator, *traductor, *ee;

public:
    Sistem(char*, char*, char*);
    ~Sistem();
    void afisare_date();
};

Sistem::Sistem(char*R, char*T, char *EE)
{
    regulator = new char[strlen(R)+1];
    strcpy(regulator, R);
    traductor = new char[strlen(T)+1];
    strcpy(traductor, T);
    ee = new char[strlen(EE)+1];
    strcpy(ee, EE);
}

Sistem::~Sistem()
{
    cout<<"Distruge obiect :" << endl;
    cout<<"\nObiect distrus :"<< endl << endl;
}

void Sistem::afisare_date()
{
        cout<<"\n Sistem";
        cout<<"\n Regulator:" << regulator;
```

```
        cout<<"\n Traductor:" << traductor;
        cout<<"\n Element executie:" <<ee;
}

class Sistem_Reglare : public Sistem
{
protected:
    char *traductor_de_nivel, *robinet_reglare, *regulator_de_nivel;

public:
    Sistem_Reglare(char*, char*, char*, char*, char*, char*);
    ~Sistem_Reglare();
    void afisare_date_SR();
};

Sistem_Reglare::Sistem_Reglare(char *R, char *T, char *EE, char *TN, char
*RReg, char *RN) : Sistem(R, T, EE)
{
    traductor_de_nivel = new char[strlen(TN)+1];
    strcpy(traductor_de_nivel, TN);
    robinet_reglare = new char[strlen(RReg)+1];
    strcpy(robinet_reglare, RReg);
    regulator_de_nivel = new char[strlen(RN)+1];
    strcpy(regulator_de_nivel, RN);
}

Sistem_Reglare::~Sistem_Reglare()
{
    cout<<"\nObiect Distrus " ;
}

void Sistem_Reglare::afisare_date_SR()
{
    Sistem::afisare_date();
    cout << "\n \n \t Sistem Reglare";
        cout << "\n Traductor-ul de Nivel :" << traductor_de_nivel;
        cout << "\n Regulator-ul de Nivel :" << regulator_de_nivel;
}

class Sistem_Reglare_Automata : public Sistem_Reglare
{
protected:
    char *actuator, *instrument_vizualizare, *senzor_de_nivel;

public:
    Sistem_Reglare_Automata(char*, char*, char*, char*, char*, char*, char*,
char*, char*);
    ~Sistem_Reglare_Automata();
    void afisare_date_SRA();
};
```

```
Sistem_Reglare_Automata::Sistem_Reglare_Automata(char *R, char *T, char *EE,
char *TN, char *RReg, char *RN, char *AC, char *IV, char *SN) :
Sistem_Reglare(R, T, EE, TN, RReg, RN)
{
    actuator = new char[strlen(AC)+1];
    strcpy(actuator , AC );
    instrument_vizualizare = new char[strlen(IV)+1];
    strcpy(instrument_vizualizare , IV);
    senzor_de_nivel = new char[strlen(SN)+1];
    strcpy(senzor_de_nivel , SN);
}

Sistem_Reglare_Automata::~Sistem_Reglare_Automata()
{
    cout<<"\nObiect distrus !"<< endl;
}

void Sistem_Reglare_Automata::afisare_date_SRA()
{
    Sistem::afisare_date();
    cout << "\n \n \n \t Sistem Reglare Automata";
        cout << "\n Actuatorul de nivel:" << actuator;
        cout << "\n Instrument Vizualizare:" << instrument_vizualizare;
        cout << "\n Senzori de Nivel:" << senzor_de_nivel;
}

class Sistem_Reglare_Automata_Nivel : public Sistem_Reglare_Automata
{
protected:
    char *adaptor;

public:
    Sistem_Reglare_Automata_Nivel(char*, char*, char*, char*, char*, char*,
char*, char*, char*, char*);
    ~Sistem_Reglare_Automata_Nivel();
    void afisare_date_SRA_Nivel();
};

Sistem_Reglare_Automata_Nivel::Sistem_Reglare_Automata_Nivel(char *R, char
*T, char *EE, char *TN, char *RReg, char *RN, char *AC, char *IV, char *SN,
char *A) : Sistem_Reglare_Automata(R, T, EE, TN, RReg, RN, AC, IV, SN)
{
    adaptor = new char[strlen(A)+1];
    strcpy(adaptor, A);
}

Sistem_Reglare_Automata_Nivel::~Sistem_Reglare_Automata_Nivel()
{
    cout<<"\n Obiect Distrus!" << endl;
}

void Sistem_Reglare_Automata_Nivel::afisare_date_SRA_Nivel()
{
```

```
    Sistem::afisare_date();
    cout << "\n \n Sistem Reglare Automata Temperatura:";
        cout << "\n Adaptor:" << adaptor;
}

class Sistem_Reglare_Automata_Nivel_controller :
Sistem_Reglare_Automata_Nivel
{
protected:
    char *rotametru;
    double eroare;
    int afisaj;

public:
    Sistem_Reglare_Automata_Nivel_controller(char*, char*, char*, char*,
char*, char*, char*, char*, char*, char*, char*, double, int);
    ~Sistem_Reglare_Automata_Nivel_controller();
    void afisare_date_SRA_Nivel_controller();
};

Sistem_Reglare_Automata_Nivel_controller::Sistem_Reglare_Automata_Nivel_cont
roller(char *R, char *T, char *EE, char *TN, char *RReg, char *RN, char *AC,
char *IV, char *SN, char *A, char *RO, double ER , int AF) :
Sistem_Reglare_Automata_Nivel(R, T, EE, TN, RReg, RN, AC, IV, SN, A)
{
    rotametru = new char[strlen(RO)+1];
    strcpy(rotametru, RO);
    eroare = ER;
    afisaj = AF;
}

Sistem_Reglare_Automata_Nivel_controller::~Sistem_Reglare_Automata_Nivel_con
troller()
{
    cout<<"\n Obiect Distrus!"<< endl;
}

void
Sistem_Reglare_Automata_Nivel_controller::afisare_date_SRA_Nivel_controller(
)
{
    Sistem::afisare_date();
    cout << "\n Sistem Reglare Automata a Nivelului Controller:";
        cout << "\n Rotametru de Nivel:" << rotametru;
        cout<<"\n Eroarea:"<<eroare;
        cout<<"\n Afisaj:"<<afisaj;
}

class Sistem_Reglare_Automata_Nivel_Actionare : public
Sistem_Reglare_Automata_Nivel_controller
{
protected :
    char *actiune;
```

```
public:
    Sistem_Reglare_Automata_Nivel_Actionare(char*, char*, char*, char*,
char*, char*, char*, char*, char*, char*, char*, double, int, char*);
    ~Sistem_Reglare_Automata_Nivel_Actionare();
    void afisare_date_SRA_Nivel_Actionare();
};

Sistem_Reglare_Automata_Nivel_Actionare::Sistem_Reglare_Automata_Nivel_Actio
nare(char *R, char *T, char *EE, char *TN, char *RReg, char *RN, char *AC,
char *IV, char *SN, char *A, char *RO, double ER , int AF, char*ACT) :
Sistem_Reglare_Automata_Nivel_controller(R, T, EE, TN, RReg, RN, AC, IV, SN,
A, RO, ER, AF)
{

    actiune = new char[strlen(ACT)+1];
    strcpy(actiune, ACT);

}

void
Sistem_Reglare_Automata_Nivel_Actionare::afisare_date_SRA_Nivel_Actionare()
{

Sistem_Reglare_Automata_Nivel_controller::afisare_date_SRA_Nivel_controller(
);
    cout<<"\n Actiunea:"<<actiune;
}

class Sistem_Reglare_Automata_PI
{
    protected: int kr,ti;
    public: Sistem_Reglare_Automata_PI(int,int);
             ~Sistem_Reglare_Automata_PI();
             void afisare_date();
};

Sistem_Reglare_Automata_PI::Sistem_Reglare_Automata_PI(int KR,int TI)
{
    kr=KR;
    ti=TI;
}

Sistem_Reglare_Automata_PI::~Sistem_Reglare_Automata_PI()
{
    cout<<"\n Obiect distrus!"<<endl;
}

void Sistem_Reglare_Automata_PI::afisare_date()
{
    cout<<"\n Kr:"<<kr;
    cout<<"\n Ti:"<<ti;
}
```

```
class Sistem_Reglare_Automata_PID:public Sistem_Reglare_Automata_PI
{
    protected: int td;
    public: Sistem_Reglare_Automata_PID(int,int,int);
             ~Sistem_Reglare_Automata_PID();
             void afisare_date_PID();
             int set_td(double SET_POINT){td=SET_POINT;}
             int get_td();
};

Sistem_Reglare_Automata_PID::Sistem_Reglare_Automata_PID(int KR,int TI,int
TD):Sistem_Reglare_Automata_PI(KR,TI)
{
    kr=KR;
    ti=TI;
    td=TD;
}

Sistem_Reglare_Automata_PID::~Sistem_Reglare_Automata_PID()
{
    cout<<"\n Obiect distrus!"<<endl;
}

void Sistem_Reglare_Automata_PID::afisare_date_PID()
{
    Sistem_Reglare_Automata_PI::afisare_date();
    cout<<"\n Td:"<<td;
}

int Sistem_Reglare_Automata_PID::get_td()
{
    return td;
}

int main()
{
    Sistem_Reglare_Automata_Nivel_controller s1("De Nivel", "De Nivel",
"Automat", "Diferentiala", "Diferentiala", "AC", "Tip cadran", "Tablou",
"Superior", "De Nivel", "Nivelului Curent", 0.03, 16);
    Sistem_Reglare_Automata_Nivel_controller s2(" Nivel", " Nivel",
"Automat", "Exponentiala", "Diferentiala", "AC", "Tip cadran", "Tablou",
"Superior", "De Nivel", "Nivelului Recent", 0.3, 12);
    Sistem_Reglare_Automata_Nivel_controller s3("De Nivel", "De Nivel",
"Automat", "Diferentiala", "Diferentiala", "AC", "Tip cadran", "Tablou",
"Exterior", "De Nivel", "Diferenta de nivele", 0.16, 9);
        s1.afisare_date_SRA_Nivel_controller();
        cout<<"\n_____
_____"<< endl;
        s2.afisare_date_SRA_Nivel_controller();
        cout<<"\n_____
_____"<< endl;
        s3.afisare_date_SRA_Nivel_controller();
```

```
        cout<<"\n_____
______"<<endl;
        Sistem_Reglare_Automata_PID s4(4,7,9);
        s4.afisare_date_PID();
        s4.set_td(6);
        cout<<"\n Noua valoare de referinta TD:"<<s4.get_td();

cout<<"\n_____"<
<endl;



    return 0;
}
```

4. Testarea :

Exemple pentru 3 seturi de date :

***toate componentele sunt sub cerintele minime**

***toate componentele indeplinesc cerintele recomandate**

***componentele au date aleatorii ce indeplinesc cerinte minime pentru diferite sisteme de operare**

```
Sistem
Regulator:De Nivel
Traductor:De Nivel
Element executie:Automat
Sistem Reglare Automata a Nivelului Controller:
Rotametru de Nivel:Nivelului Curent
Eroarea:0.03
Afisaj:16
```

```
Sistem
Regulator: Nivel
Traductor: Nivel
Element executie:Automat
Sistem Reglare Automata a Nivelului Controller:
Rotametru de Nivel:Nivelului Recent
Eroarea:0.3
Afisaj:12
```

```
Sistem
Regulator:De Nivel
Traductor:De Nivel
Element executie:Automat
Sistem Reglare Automata a Nivelului Controller:
Rotametru de Nivel:Diferenta de nivele
Eroarea:0.16
Afisaj:9
```

```
Kr:4
Ti:7
Td:9
Noua valoare de referinta TD:6
```

```
Obiect distrus!

Obiect distrus!

Obiect Distrus!

Obiect Distrus!
```

5. Concluzii :

Sistemul poate menține un nivel constant al lichidului, indiferent de variațiile de debit sau de alți parametri perturbatori

Sistemul poate reduce erorile de nivel și variațiile nivelului, îmbunătățind precizia și stabilitatea sistemului

Sistemul poate permite o mai bună monitorizare și control al nivelului, făcând posibilă detectarea și corectarea erorilor sau a problemelor înainte ca acestea să devină critice

Sistemul poate îmbunătăți eficiența și reduce costurile prin automatizarea procesului de reglare a nivelului și eliminarea intervențiilor manuale

Automatizarea poate face posibilă înregistrarea și analiza datelor, permitând identificarea tendințelor și îmbunătățirea performanței în timp

Sistemul poate îmbunătăți siguranța prin monitorizarea continuă a nivelului și detectarea rapidă a depășirii limitelor permise, prevenind astfel accidentele sau daunele materiale.

Sistemul poate optimiza performanța și eficiența procesului prin reglarea precisă a nivelului, asigurând că componentele critice ale procesului sunt alimentate în mod continuu și evitând suprasarcina sau alte probleme.

Sistemul poate face posibilă integrarea cu alte sisteme de control și automatizare, creând un sistem de control centralizat pentru mai multe procese sau echipamente.

Sistemul poate permite o mai bună scalabilitate și flexibilitate, adaptându-se la schimbările în parametrii procesului sau la noi cerințe de producție.

Sistemul poate reduce costurile de întreținere prin automatizarea procesului de reglare a nivelului și prevenirea problemele mecanice sau de alte tip.

6. Bibliografie :

Programare orientata pe obiecte. exemple in limbajul C++ , Mihaela Oprea

https://ro.wikipedia.org/wiki/Automatizare