Activitatile II.4 si II.5 (Raport stiintific):

II.4 Dezvoltare software de timp real pentru conducerea procesului de epurare biologica.

II.5 Dezvoltare softare de baza pentru sistemul de conducere a procesului de epurare biologica.

1. Introducere

Se face mentiunea ca cele doua activitati sunt tratate unitar, intr-un singur raport stiintific datorita faptului ca sunt legate intre ele din punct de vedere tematic si conceptual. Astfel, pentru conducerea statiei de epurare biologica de laborator a fost proiectat un sistem software compus din doua entitati principale:

1. interfata utilizator (om-masina);
2. sistemul de conducere al procesului de epurare propriuzis.

Cele doua componente ale sistemului de programe sunt entitati software complet independente, comunicarea dintre ele realizandu-se printr-un protocol de comunicatie *UDP* (*User Datagram Protocol* - protocolul datagramelor utilizator). A doua componenta, sistemul de conducere, a fost realizata in doua variante, dupa cum urmeaza: prima varianta, in limbajul *C++*, sub sistemul de operare *Linux*, folosind facilitatile multi-tasking ale acestui sistem de operare; a doua varianta, in *Matlab – Simulink*, sub sistemul de operare *Windows XP*, folosind facilitatile de timp real ale mediului de simulare *Matlab*. Ambele variante realizeaza aceleasi functii ale sistemului de conducere si pot comunica cu interfata om-masina prin acelasi protocol *UDP*. Proiectantul aplicatiei poate selecta oricare dintre cele doua variante de software de conducere cu care sa realizeze un experiment.

In continuare, sunt prezentate cele doua sisteme informatice de conducere:

2. Interfata om-masina

2.1 Introducere

Interfata grafica (*HMI – Human-Machine Interface*) reprezinta prima componenta software a programului de conducere a statiei de epurare, care realizeaza dialogul intre operatorul uman si procesul de control al instalatiei. Ea a fost conceputa pentru a fi utilizata de operatorii tehnologi, care, in cele mai multe cazuri, nu dispun in mod necesar de cunostinte informatice. Astfel, s-a optat pentru o reprezentare schematica a instalatiei de epurare (schema sinoptica), interfata oferind utilizatorului optiuni clare de control si monitorizare a elementelor din sistem.

Pentru a asigura o operare facila, *HMI* a fost impartita in mai multe sectiuni, fiecare oferind o alta perspectiva asupra procesului.

Vizualizarea schemei instalatiei si a starii elementelor din sistem a fost implementata folosind patru moduri de vizualizare: trei dintre ele trateaza subiectele principale de interes (bazinul de alimentare, bazinul aerat si decantorul) impreuna cu elementele adiacente (robinete, traductoare, pompe etc.), al patrulea mod de vizualizare oferind o imagine de ansamblu asupa schemei sinoptice a instalatiei si asupra starii procesului.

Pentru urmarirea evolutiei in timp a valorilor marimilor din proces, interfata grafica ofera doua moduri de vizualizare a marimilor sub forma de grafice: grafice liniare si bargraph-uri.

De asemenea, interfata grafica dispune de optiunea consultarii rezultatelor experimentelor anterioare, operatorul putand afisa grafic sau exporta intr-un fisier, evolutia in timp a marimilor unui experiment anterior.

2.2 Functionalitate

Interfata grafica este impartita in cinci sectiuni:

1. Schema sinoptica
2. Bazinul de alimentare
3. Bazinul aerat
4. Decantorul
5. Evolutii grafice
6. Bargraph-uri
7. Arhiva

2.2.1 Schema sinoptica generala

Prima sectiune a interfetei grafice (*HMI*) ofera o vedere de ansamblu asupra instalatiei de epurare, cuprinzand toate elementele sistemului si marimile din proces (Figura 33); in acest mod de vizualizare nu se pot modifica paramterii instalatiei. Din aceasta perspectiva, se poate observa cu usurinta atat starea globala a sistemului, cat si aceea a diferitelor elemente componente. In dreptul fiecarui obiect activ al instalatiei este afisata starea sa (*ON/OFF*) sau valoarea care il caracterizeaza la un moment dat. Daca elementul este oprit, starea/valoarea sa va fi afisata folosind culoarea rosie, altfel aceasta va fi afisata folosind culoarea verde. Traductoarele au fost grupate in partea din dreapta sus a ecranului, restul componentelor fiind afisate conform pozitiei lor in instalatia fizica de epurare. Figura 34 prezinta in detaliu modul de afisare a starii/valorilor in cadrul schemei sinoptice.

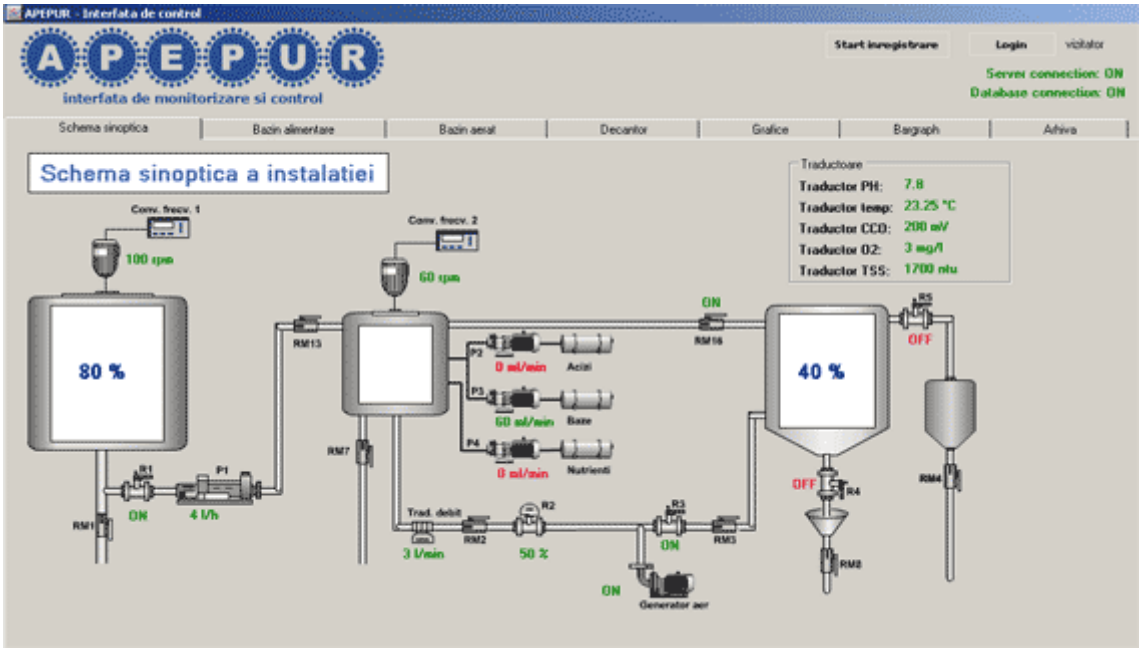


Figura 33: Schema sinoptica a instalatiei

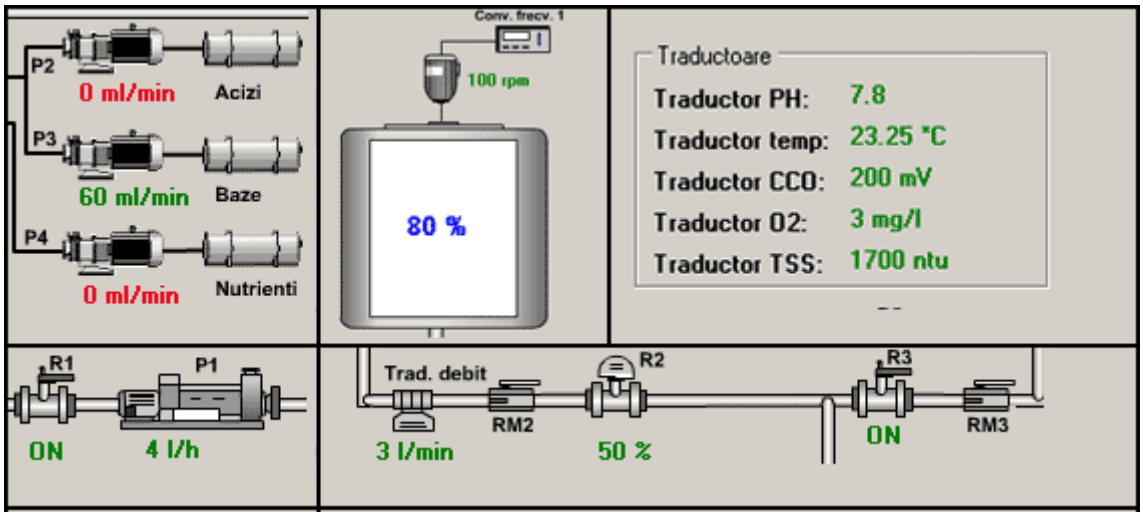


Figura 34: Afisarea starii elementelor in schema sinoptica

Schema sinoptica a instalatiei este divizata in trei parti, care prezinta o detaliere superioara a zonei de interes: bazinul de alimentare, bazinul aerat si decantorul. Pentru a vizualiza o anumita parte a instalatiei, aceasta se selecteaza din meniul principal sau se da click pe zona dorita din schema sinoptica (Figura 33).

2.2.2 Bazinul de alimentare

Aceasta sectiune prezinta in detaliu elementele componente ale primei zone de interes din schema sinoptica, bazinul de alimentare (Figura 35). Elementele active din aceasta portiune a instalatiei sunt: traductorul de nivel pentru bazinul de alimentare, agitatorul, electroventilul bipozitional *R1* si pompa *P1*.

Nivelul apei din bazinul de alimentare este reprezentat grafic in schema si este afisat in procente in partea dreapta a bazinului. In figura 36 se pot observa reprezentarile bazinului de alimentare la umpleri de 20%, 50% si 90 %.

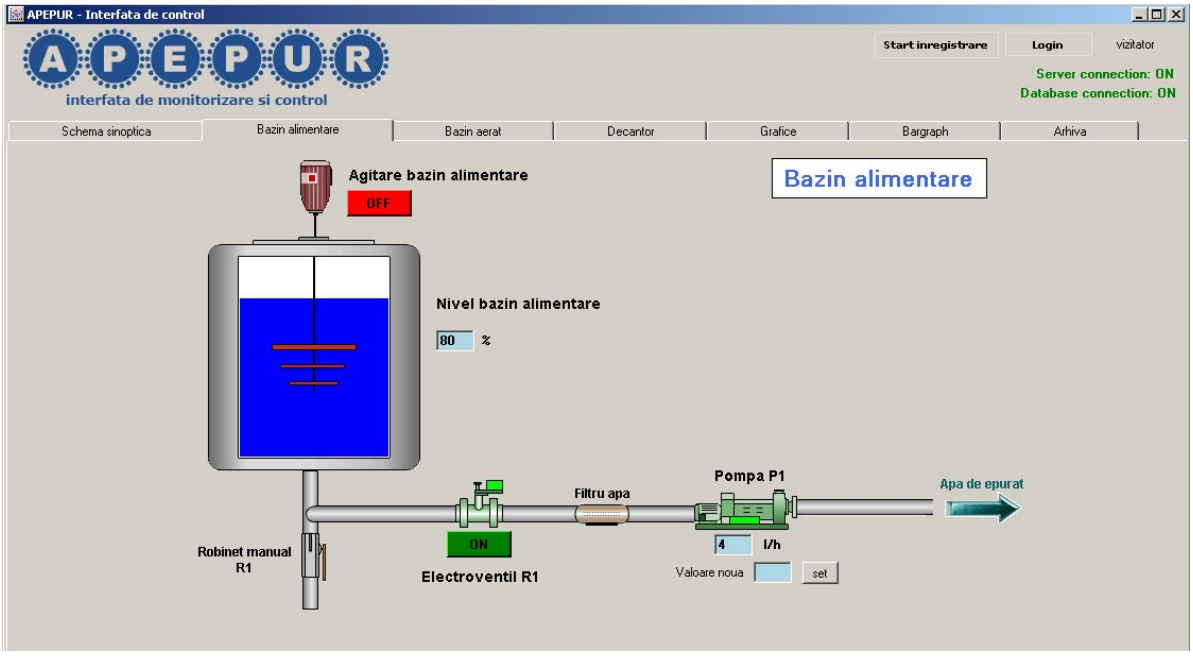


Figura 35: Bazinul de alimentare

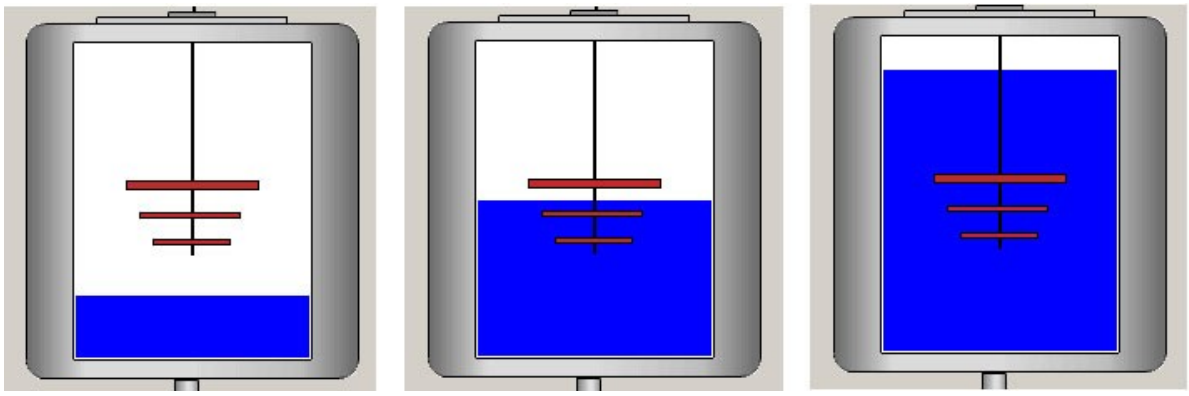


Figura 36: Reprezentarea bazinului de alimentare la diferite nivele de umplere (20%, 50%. 90%)

Toate elementele din figura 35 sunt comandate manual de catre operator. Pentru a putea modifica parametrii sistemului, acesta trebuie sa fie autentificat pe baza numelui de utilizator si a parolei.

Reprezentarea starii componentelor este realizata dupa cum urmeaza: daca elementul este oprit, acesta este afisat in culoare rosie si in dreptul sau exista un buton rosu cu textul *OFF*; daca elementul este pornit, el este afisat in culoare verde, avand asociat un buton cu textul *ON*. Schimbarea starii se realizeaza prin actionarea butonului corespunzator elementului vizat (Figura 36).

Pentru a comanda pompa *P1*, operatorul introduce o valoare in casuta corespunzatoare si actioneaza butonul *Set*. In cazul introducerii unei valori eronate (peste debitul maxim al pompei), acesta va fi avertizat de prezenta erorii si comanda nu se va transmite.

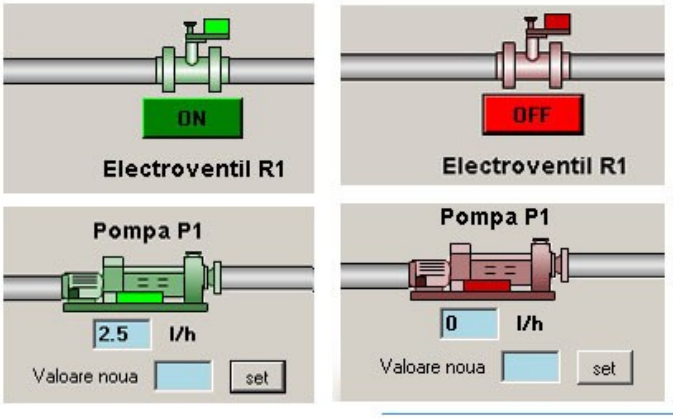


Figura 36: Afisarea starii elementelor (stanga: element deschis, dreapta:element inchis)

2.2.3 Bazinul aerat

In aceasta sectiune se prezinta bazinul aerat impreuna cu componentele adiacente. Aceasta parte contine cele mai multe elemente active, dupa cum se poate observa si in figura 37.

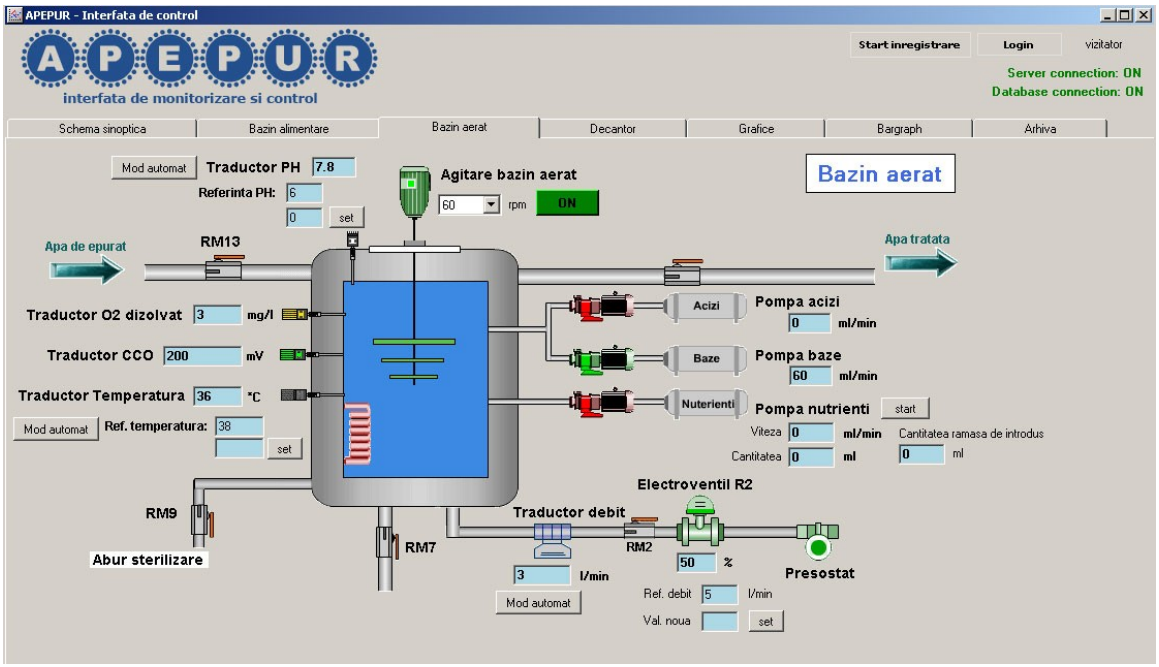


Figura 37: Bazinul aerat

Agitatorul este analog celui de la bazinul de alimentare, avand in plus optiunea modificarii vitezei de rotatie (Figura 38). Reprezentarea starii este similara cu a celorlalte componente: daca este pornit se foloseste culoarea verde, in caz contrar culoarea rosie. Comanda agitatorului se realizeaza astfel: daca este pornit, selectia unei alte viteze de rotatie se alege din lista (Figura 38), comanda fiind automat transmisa software-ului de control al procesului; inchiderea agitatorului se realizeaza prin actionarea butonului de stare. Daca agitatorul este inchis, la selectia unei alte viteze de rotatie din lista, comanda nu se va transmite decat atunci cand se porneste agitatorul prin actionarea butonului de stare.

Figura 38: Comanda agitatorului pentru bazinul aerat

Traductoarele de oxigen dizolvat si *CCO* (consum chimic de oxigen) afiseaza valorile primite de la software-ul de control al procesului (Figura 39).

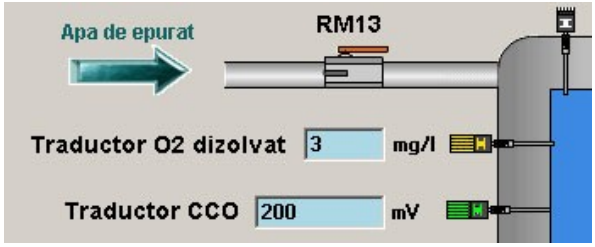


Figura 39: Traductoarele de Oxigen dizolvat si Consum chimic de oxigen

Controlul *pH*-ului se poate realiza in doua moduri: in mod automat si in mod manual. In modul automat, se specifica o referinta pentru *pH*, bucla de reglare din software-ul de control al procesului dozand in mod corespunzator debitul pompelor *P2* (acizi) si *P3* (baze) pentru atingerea referintei impuse. In modul manual, operatorul selecteaza debitul pompelor *P2* si *P3*, acestea nemaiprimind comenzi de la software-ul de control al procesului. Modul curent de operare este afisat intr-un buton ce contine textul „*Mod automat*” sau „*Mod manual*”. Schimbarea modului de functionare se realizeaza prin actionarea acestui buton. Traductorul de *pH* afiseaza valoarea curenta a *pH*-ului.

In figura 40 este prezentata bucla de reglare a *pH*-ului in mod automat. Referinta este impusa prin introducerea valorii in caseta corespunzatore si apasarea butonului *Set*. In acest mod de functionare, nu se pot comanda manual pompele *P2* si *P3*.

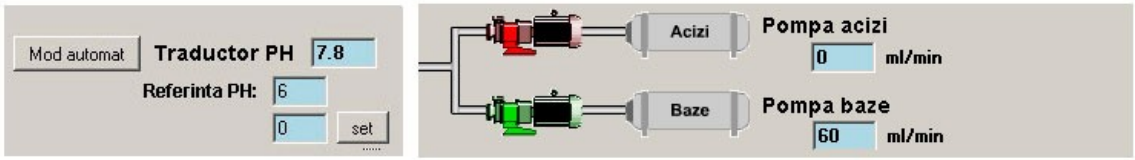


Figura 40: Controlul *pH*-ului in mod automat

Figura 41 prezinta modul manual de control al *pH*-ului. Se observa imposibilitatea setarii unei referinte, controlul pompelor de acizi si baze facandu-se manual, prin introducerea debitului in casuta corespunzatoare si actionarea butonului *Set*.

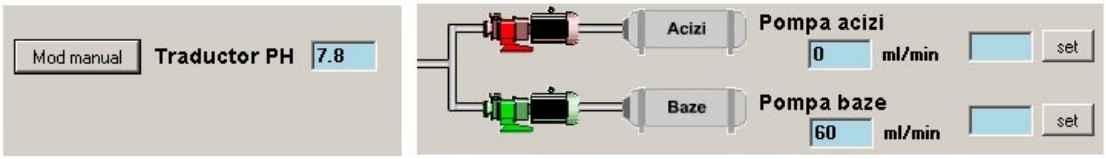


Figura 41: Controlul *pH*-ului in mod manual

Controlul temperaturii in interiorul bazinului aerat are, de asemenea, doua moduri de functionare: automat si manual. Similar, in modul automat se impune o referina pentru temperatura, bucla de reglare din software-ul de control al procesului cupland/decupland rezistenta de incalzirepentru a atinge acea referinta. In modul manual operatorul poate porni sau opri rezistenta de incalzire dupa necesitati. Traductorul de temperatura ofera informatii despre temperatura curenta din interiorul bazinului aerat. Rezistenta de incalzire, daca este cuplata, va fi afisata in culoarea rosie, iar daca este decuplata, in culoarea alba. Schimbarea modurilor de functionare se face la fel cu controlul *pH*-ului, prin actionarea butonului ce afiseaza modul curent de operare. Modul automat de control al temperaturii este prezentat in figura 42. Referinta pentru temperatura se introduce in casuta si se actioneaza abutonul *Set*. In acest exemplu, referinta este mai mare decat valoarea curenta indicata de traductor si deci rezistenta este cuplata (spre deobesire de celelalte elemente active ale sistemului, culoarea rosie, in acest caz, arata faptul ca rezistenta este cuplata si ea incalzeste apa).

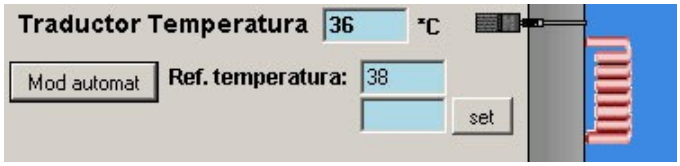


Figura 42: Controlul temperaturii in mod automat

Modul manual este prezentat in figura 43. Se observa posibilitatea cuplarii sau decuplarii explicite a rezistentei de incalzire dupa necesitati.

Figura 43: Controlul temperaturii in mod manual

Debitul de aer ce intra in bazinul aerat poate fi controlat tot in doua moduri de functionare: automat si manual. In modul automat se impune o referinta pentru debitul de aer si bucla de reglare din sistemul de control al procesului comanda deschiderea electroventilului cu actionare continua *R2* pentru a atinge referinta data. In modul manual, operatorul poate comanda deschiderea electroventilului *R2* la o anumita valoare. Presostatul din partea dreapta a schemei indica prezenta/absenta aerului produs de generator, in functie de culoarea de reprezentare (verde – exista aer, rosu – nu exista aer). Traductorul de debit afiseaza debitul de aer ce patrunde in bazinul aerat. Cele doua moduri de functionare sunt prezentate in figurile 44 si 45.

Pompa pentru introducerea nutrientilor in bazinul aerat (*P4*) este comandata manual de catre operator. Acesta introduce debitul pompei si cantitatea de introdus, apoi actioneaza butonul *Start* (Figura 46). Pompa *P4* va introduce cantitatea de nutrienti specificata in bazinul aerat, dupa care se va opri automat.

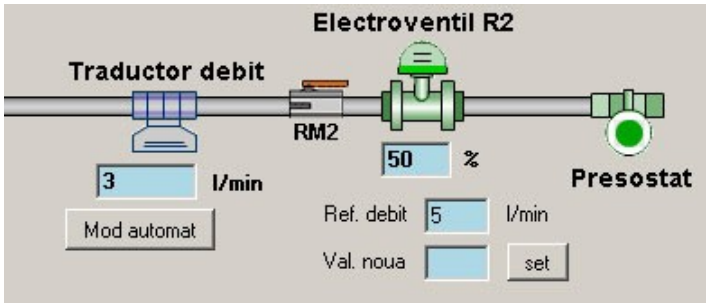


Figura 44: Controlul debitului de aer in mod automat

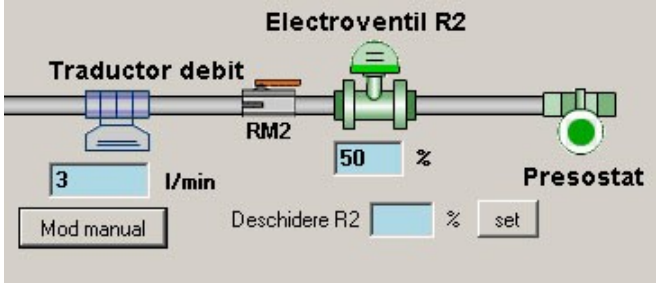


Figura 45: Controlul debitului de aer in mod manual

Figura 46: Comanda pompei de nutrienti

2.2.4. Decantorul

Ultima parte de interes din cadrul schemei sinoptice o reprezinta decantorul. In aceasta zona, elementele active sunt electroventilele bipozitionale *R3*, *R4*, *R5*, *R6*, nivelul apei din decantor si valoarea indicata de traductorul pentru determinarea suspensiilor solide - *TSS* (Figura 47).

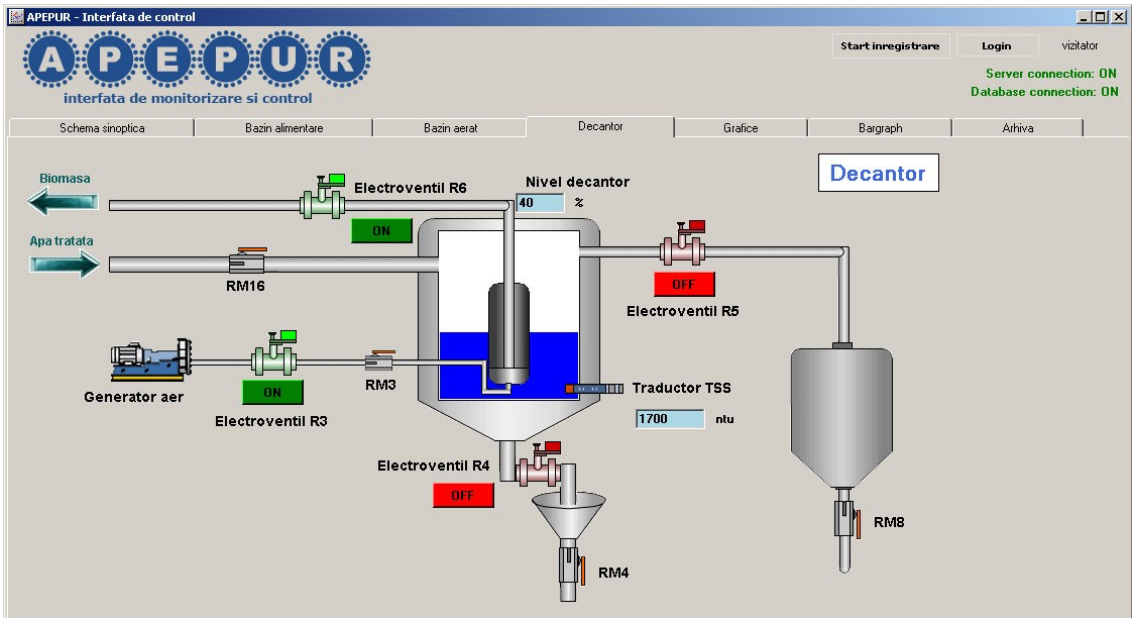


Figura 47: Schema sinoptica a decantorului

Comanda electroventilelor se face prin actionarea butonului corespunzator (din dreptul elementului vizat). Nivelul apei din decantor este reprezentat in mod similar cu nivelul apei din bazinul de alimentare. Valoarea indicata de traductorul de concentratie a suspensiilor solide *TSS* este afisata in casuta de sub reprezentarea grafica a traductorului (Figura 48).

Figura 48: Traductorul *TSS*

2.2.5 Grafice

In aceasta sectiune, operatorul poate vizualiza in timp real evolutia marimilor din proces. Pe un grafic pot fi afisate simultan doua marimi (Figura 49). Operatorul selecteaza marimea dorita din lista si graficul acesteia va apare in caseta grafica corespunzatoare. In partea dreapta a listei de selectie se afiseaza valoarea instantanee a marimii selectate.

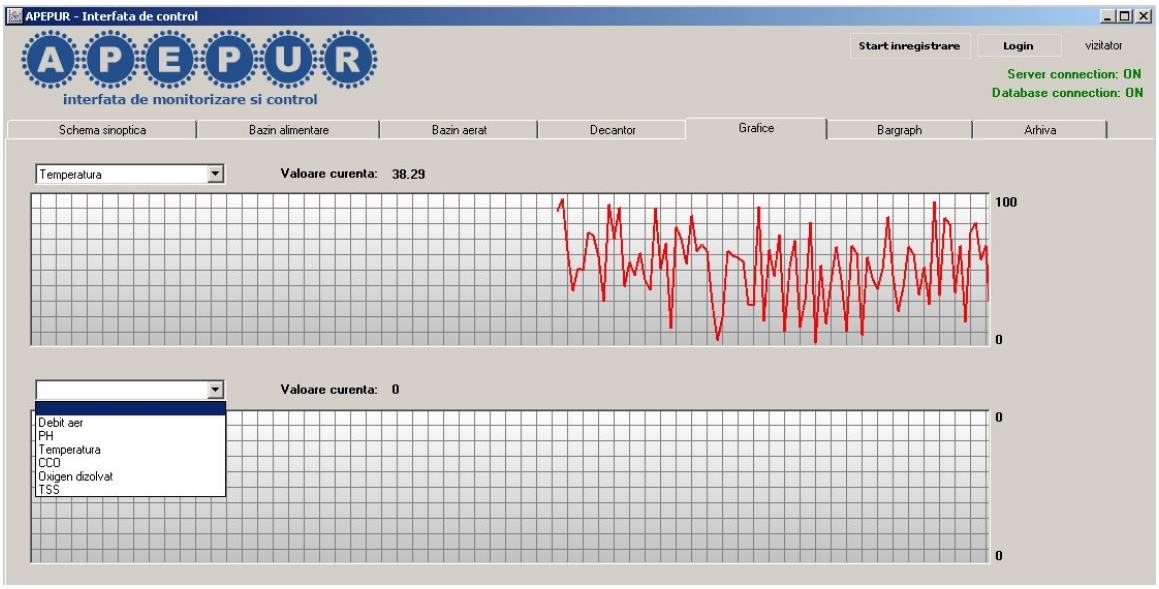


Figura 49: Sectiunea „Grafice”

2.2.6 Reprezentare sub forma de bargraph

Reprezinta al doilea mod de vizualizare grafica a evolutiei marimilor din sistem. Acestea sunt reprezentate printr-o bara verticala ce isi schimba lungimea odata cu variatia valorii marimii respective (Figura 50).

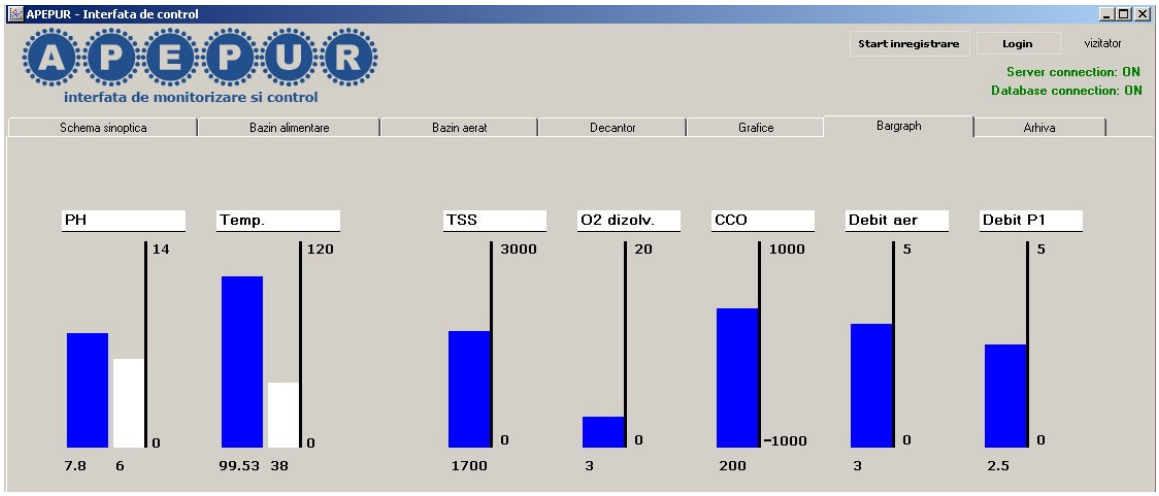


Figura 50: Sectiunea „Bargraph”

Marimile ce au asociate o valoare de referinta sunt reprezentate printr-un bargraph dublu, avand referinta de culoare alba afisata in partea dreapta (Figura 51a). Marimile fara o valoare de referinta asociata sunt reprezenate printr-un bargraph simplu (Figura 51b). La depasirea unei anumite valori limita de alerta, culoarea barei verticale se schimba din albastru in rosu pentru a notifica operatorul de depasirea pragului impus.

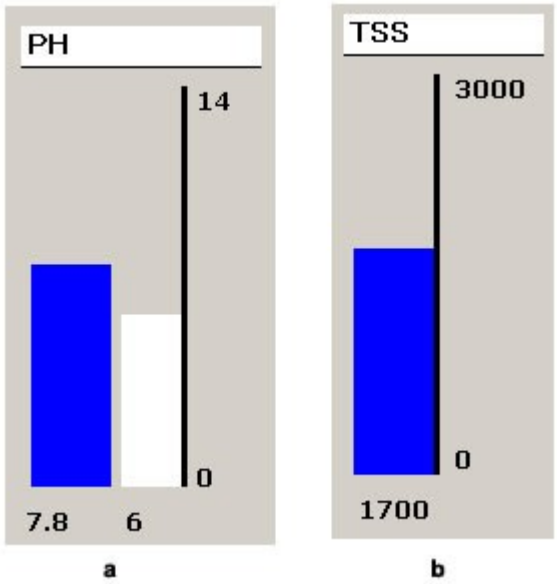


Figura 51: a) Bargraph dublu (cu referinta) b) Bargraph simplu

2.2.7 Arhiva

Aceasta sectiune permite vizualizarea grafica e evolutiei marimilor in cadrul unor experimente anterioare (Figura 52). In partea stanga sunt prezentate experimentele anterioare sub forma „*nume_experiment:data*”. Dupa selectarea experimentului dorit, se poate afisa grafic evolutia valorilor marimilor, selectia acestora facandu-se in partea dreapta a ecranului. Se pot afisa pana la 3 marimi in fiecare grafic din cele doua grafice disponibile.

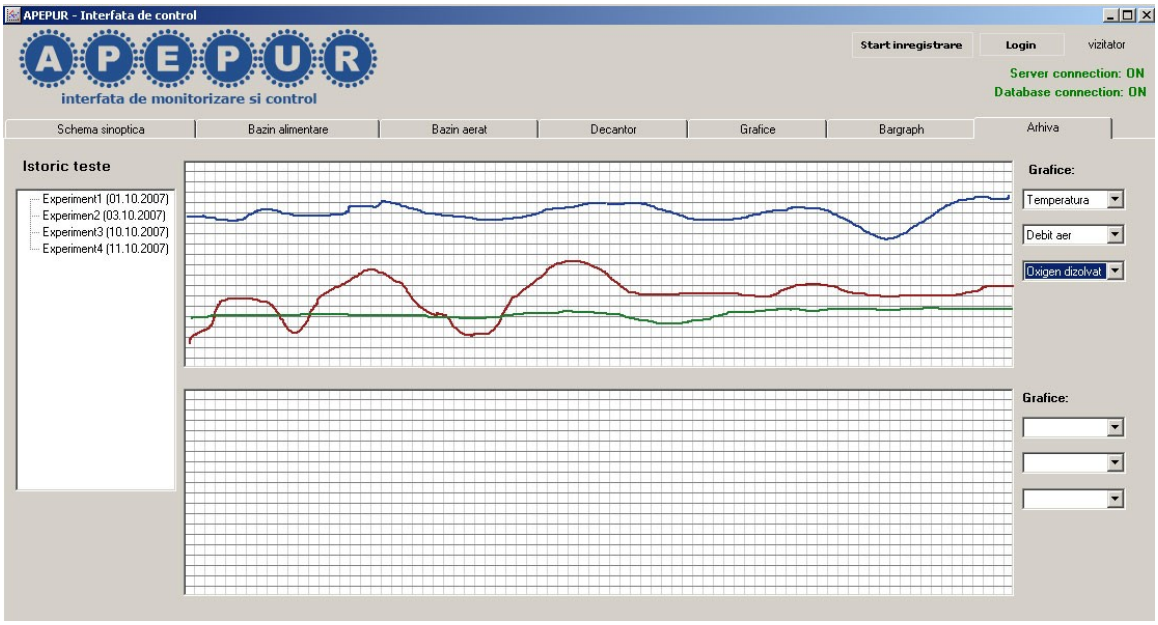


Figura 52: Arhiva

Inregistrarea valorilor in arhiva se face prin actionarea butonului „*Start inregistrare*” si introducerea numeului experimentului (Figura 53). Operatorul nu este conditionat de inregistrarea continua a valorilor unui experiment, memorarea valorilor putandu-se face pe diferite perioade de interes. Implicit, numele experimentului afisat in caseta de dialog este numele ultimului experiment in

contul caruia s-a realizat o monitorizare. Daca se schimba numele experimentului, memorarea valorilor se va realiza pentru noul experiment, atfel se vor inregistra pentru experimentul anterior, a carui finalitate nu a fost specificata. Oprirea memorarii valorilor in baza de date se face actionand acelasi buton ca si la pornirea inregistrarii, care va afisa textul „*Stop inregistrare*”.

Figura 53: Pornirea inregistrarii valorilor in baza de date

2.3Interfata om-masina – structura informatica

2.3.1 Implementare software

Interfata grafica (*HMI* – Human Machine Interface) a fost dezvoltata folosind limbajul de programare *C#*, aceasta alegere fiind sustinuta de capabilitatile foarte bune de care dispune *C#* pentru programarea orientata obiect, cat si de existenta unei suite considerabile de componente grafice predefinite in biblioteca *.NET*. Marea majoritate a componentelor prezente in interfata sunt obiecte simple (*Label-uri*, *Textbox-uri*, *Butoane*) existente in biblioteca *.NET*, pe langa care au fost dezvoltate si componente grafice personalizate, precum bargraph-urile sau graficele in timp real. Interfata grafica comunica bidirectional cu software-ul de conducere a bioreactorului prin intermediul procolului de comunicatie *UDP*. Pentru memorarea datelor referitoare la starea sistemului si analiza ulterioara a datelor, se folosesc facilitatile oferite de sistemul de gestiune al bazelor de date *MS SQL SERVER 2005*. Informatiile schimbate intre *HMI* si software-ul de conducere se impart in doua categorii: *date dinamice*, a caror valoare este achizitionata prin intermediul senzorilor (temperatura, *pH*, debit aer, etc) si *comenzile* trimise de catre operator instalatiei de epurare. Datele dinamice sunt trimise de software-ul de conducere catre interfata grafica la un interval de 250ms, sub forma unui pachet *UDP* continand variabile de tip *double* (64biti). La primirea pachetului de date, interfata grafica se actualizeaza cu noile valori si acestea sunt introduse intr-o baza de date (la un interval de timp specificat) pentru arhivare. Programul ce introduce valorile in baza de date este independent de interfata grafica, nefiind astfel conditionat de rularea acesteia. Tabelul 12 prezinta ordinea datelor in datagrama *UDP*.

Tabelul 12

Pozitie	Nume	Descriere
1	Nivel1	Nivelul apei in bazinul de alimentare (in procente)
2	Avarie	Semnal de avarie (0=ON 1=OFF)
3	PH	Valoare PH (0-14)
4	Temp	Valoare temperatura (0 – 120 ºC)
5	CCO	Valoare consum chimic de oxigen (-1000 +1000 mV)
6	O2	Valoare oxigen dizolvat (0-20 mg/l)
7	Debit_aer	Valoare debit aer (0 – 5 l/min)
8	Nivel3	Nivelul apei in decator (procente)
9	TSS	Valoare TSS (0 -3000 ntu)

10	Presostat	Prezenta presiunii (0=OFF 1=ON)
11	Heater	Rezistenta de incalzire in modul automat (0=OFF 1=ON)
12	P2	Debitul pompei P2 in modul automat (16,8 – 840 ml/min)
13	P3	Debitul pompei P3 in modul automat (16,8 – 840 ml/min)
14	R2	Deschiderea electroventilului R2 in modul automat (procente)
15	Ref_P1	Rezervate pentru folosire ulterioara
16	Ref_DebitAer	

Comenzile emise de operator sunt trimise tot sub forma unui pachet *UDP* catre software-ul de conducere a procesului, care va comanda bioreactorul. Odata cu trimiterea datelor catre aplicatia de control, comenzile emise sunt inregistrate intr-o baza de date pentru a putea reface starea sistemului la un moment de timp ulterior. Detalierea continutului pachetului de comenzi este prezenta in Tabelul 13.

Tabelul 13

Pozitie	Nume	Descriere
1	M1	Starea motorului M1 (0=OFF 1=ON)
2	R1	Starea electroventilului R1 (0=OFF 1=ON)
3	ModBioreactor	Rezervat pentru folosire ulterioara
4	P1	Debit pompa P1 (0 – 5 l/h)
5	Ref_rata_dilutie	Rezervat pentru folosire ulterioara
6	P4	Debit pompa P4 (16,8 – 840 ml/min)
7	Ref_debit	Referinta debit aer
8	R3	Starea electroventilului R3 (0=OFF 1=ON)
9	R4	Starea electroventilului R4 (0=OFF 1=ON)
10	R5	Starea electroventilului R5 (0=OFF 1=ON)
11	R6	Starea electroventilului R6 (0=OFF 1=ON)
12	Ref_Temp	Referinta temperatura
13	Ref_PH	Referinta PH
14	Ref_O2	Referinta O2
15	Ref_Substrat	Rezervat pentru folosire ulterioara
16	M2	Starea motorului M2 (0, 60, 180, 300, 420)
17	P5	Rezervat pentru folosire ulterioara
18	Heater	Starea rezistentei de incalzire R6 (0=OFF 1=ON)
19	P2	Debit pompa P2 (16,8 – 840 ml/min)
20	P3	Debit pompa P3 (16,8 – 840 ml/min)
21	R2	Dechidere electroventil R2 (procente)
22	ModDebit	Mod functionare bucla debit aer (0=Automat 1=Manual)
23	ModTemp	Mod functionare bucla temperatura (0=Automat 1=Manual)
24	Mod PH	Mod functionare bucla PH (0=Automat 1=Manual)

Arhitectura aplicatiei a fost conceputa simplu, pentru a usura mentenanta software-ului si a oferi posibilitati de extensibilitate. Componenta principala a interfetei este fereastra principala (clasa MainWindow, derivata din clasa Form) care are rol de „container” pentru celelalte elemente grafice: titlul aplicatiei, componenta de tip *TabControl* in care se gasesc toate sectiunile interfetei,

informatiile de stare despre conexiunea cu baza de date si server-ul, butonul de autentificare etc.

La pornirea aplicatiei, se instantiaza un obiect al clasei MainWindow care este rulat in firul de executie principal. Constructorul clasei are rolul principal de a initializa componentele grafice (atat cele predefinite, cat si cele personalizate) si de a reface starea anterioara a procesului, prin interogarea unui tabel al bazei de date si extragerea ultimelor valori cunoscute ale marimilor. Tot la instantierea clasei se creeaza un obiect de tip DataReceiver care primeste pachete *UDP* intr-un fir de executie separat si notifica obiectul clasei MainWindow de receptionarea mesajului.

```
public MainWindow() {
    InitializeComponent();
    InitializeCustomComponents();
    SetInitialEnvironment();
    dataReceiver.startThread();
  }
```

Datele primite de interfata sunt memorate intr-o clasa ReceivedData (avand ca date membre variabilele prezentate in Tabelul 12) iar cele trimise catre proces intr-o clasa CommandData (ce contine variabilele prezentate in Tabelul 13).

Functia SetInitialEnvironment() creeaza un obiect de tip SQLmanager pentru a comunica cu baza de date si apleaza metoda sqlManager.GetInitialData() pentru a actualiza obiectul commandData cu ultimele valori cunoscute. De mentionat ca valoarea acestor marimi se poate modifica numai din interfata grafica si orice modificare se face cu actualizarea bazei de date, eliminandu-se astfel posibilitatea reflectarii unei stari incorecte a sistemului. Urmeaza apoi un bloc de structuri decizionale in care se modifica aspectul componentelor grafice in functie de valorile preluate din baza de date.

```
 private void SetInitialEnvironment()
 {
   sqlManager = new SQLmanager();
   if (sqlManager.Connect() != true)
   {
    MessageBox.Show("Nu exista conexiune cu baza de date","Eroare");
    return;
   }
   if (sqlManager.GetInitialData(commandData) == false)
   {
    MessageBox.Show("Eroare la interogarea bazei de date", "Eroare");
    return;
   }
   if (commandData.R1 > 0)
   {
    R1_Button.BackColor = Color.Green;
    R1_Button.Text = "ON";
    R1_Image.Image = HMI_Apepur.Properties.Resources.ElectroventilOn;
    R1_sinLabel.Text = "ON";
    R1_sinLabel.ForeColor = Color.Green;
   }
       ....................
 }
```

UpdateGUI() este functia de actualizare a interfetei grafice, in care fiecare marime din proces este parcursa, modificadu-se componentele grafice ce depind de acea marime.

```
public void UpdateGUI(){
.............................

/********  Traductor PH ******/
textValue = FormatString(processData.PH);
PH_sinLabel.Text = textValue;
PH_sinLabel.ForeColor = (processData.PH == 0) ? Color.Red:Color.Green;
PH_Textbox.Text = textValue;
PH_Textbox.ForeColor = (processData.PH == 0) ? Color.Red : Color.Black;
.............................

/********  Electroventil R2 ******/
if (commandData.ModDebit == MOD_AUTO)
{
  textValue = FormatString(processData.R2)+" %";
  R2_sinLabel.Text = textValue;
  R2_sinLabel.ForeColor = (processData.R2 == 0)?Color.Red :Color.Green;
  R2_Textbox.Text = textValue;
  R2_Textbox.ForeColor=(processData.R2 == 0) ? Color.Red : Color.Black;
  R2_Image.Image= (processData.R2 > 0)? Properties.Resources.R2On :
                                       Properties.Resources.R2Off;
}
.............................
}
```

In cadrul clasei MainWindow se regasesc si functiile ce sunt apelate ca urmare a actiunilor operatorului (ex: apasarea unui buton). Aceste functii verifica starea conexiunilor (comanda nu se poate transmite daca conexiunea cu software-ul de proces sau baza de date este intrerupta) si apoi trimit pachetul de date catre aplicatia de control a procesului, actualizeaza baza de date si aspectul componentelor grafice influentate de comanda data.

```
private void commandR1(object sender, MouseEventArgs e)
{
 if (ValidateConnections() == false )
     return;

 if (sqlManager.UpdateProcessData("R1",1- commandData.R1) == false )
     return;

   int prev = commandData.R1;

   commandData.R1 = 1- commandData.R1;
   dataSender.sendUDPpacket(commandData);

 if(prev==OFF)
 {
    R1_Button.BackColor = Color.Green;
    R1_Button.Text = "ON";
    R1_Image.Image = Properties.Resources.ElectroventilOn;
```

```
    R1_sinLabel.Text = "ON";
    R1_sinLabel.ForeColor = Color.Green;
  }
  else
  {
    R1_Button.BackColor = Color.Red;
    R1_Button.Text = "OFF";
    R1_Image.Image = Properties.Resources.ElectroventilOff;
    R1_sinLabel.Text = "OFF";
    R1_sinLabel.ForeColor = Color.Red;

     //la inchiderea lui R1 se inchide automat si P1
    if (commandData.P1> 0)
    {
      valoareNouaP1_Textbox.Text = "0";
      commandP1((object) valoareNouaP1_Textbox,null);
    }
  }

}
```

Clasa SQLmanager mediaza dialogul aplicatiei cu baza de date. Cele mai importante metode ale clasei sunt Connect(), GetInitialData() si UpdateDatabase().

Functia Connect() realizeaza conexiunea la baza de date

```
public bool Connect (){
  try
  {
   SQLconnection = new System.Data.SqlClient.SqlConnection();
   SQLconnection.ConnectionString = "*****";
   SQLconnection.Open();
   return true;
  }
  catch (Exception ex)
  {
   return false;
  }
}
```

Metoda GetInitialData() interogheaza baza de date si extrage valorile initiale la pornirea aplicatiei.

```
public bool GetInitialData(CommandData commandData)
{
 try
 {
    SqlDataReader myReader = null;
    SqlCommand myCommand = new SqlCommand("SELECT * FROM
ProcessData",
                                          SQLconnection);
    myReader = myCommand.ExecuteReader();
    myReader.Read();

    commandData.M1=Convert.ToDouble(myReader["M1"]);}
```

```
    ..............
    commandData.ModPH = Convert.ToDouble(myReader["ModFuncPH"]);
    myReader.Close();
  }
  catch (Exception ex)
  {
   return false;
  }
return true;
```

Metoda UpdateDatabase() actualizeaza tabelul de stare al prcesului.

```
public bool UpdataDatabase(string component, double value) {
 try {
    SqlCommand myCommand = new SqlCommand("UPDATE ProcessData SET "
+
    component + "=" + value.ToString(), SQLconnection);
    if (myCommand.ExecuteNonQuery() > 0)
              return true;
    else
    MessageBox.Show("Eroare la inserarea in baza de date.", "Eroare");
                  return false;
}
catch(Exception e){
   MessageBox.Show("Eroare la inserarea in baza de date.", "Eroare");
   return false;
}

}
```

Clasa DataReceiver are rolul de a prelua din retea (intr-un fir de executie separat) pachetele *UDP* si de a apela metoda UpdateGUI() a clasei MainWindow la receptionarea unui pachet. Cea mai importanta metoda a clasei este communicationThread(), firul de executie in care se receptioneaza pachetele.

```
void communicationThread()
{
  Socket receiveSocket = new Socket(AddressFamily.InterNetwork,
                            SocketType.Dgram, ProtocolType.IP);
  receiveSocket.Bind(ipReceiveData);
  receiveSocket.ReceiveTimeout = 1000;

  byte[] data = new byte[buffer.count*8];

  while (true)
  {
   int bytesReceived = 0;
   try
   {
    bytesReceived = receiveSocket.Receive(data);
   }
   catch (Exception e)
   {
    try
```

```
      {
       mainWindow.Invoke(new ServerErrorDelegate(ServerError));
      }
      catch { }
    }

   if (bytesReceived > 0)
   {
    try
    {
     Monitor.Enter(buffer);

     buffer.Nivel1 = BitConverter.ToDouble(data, 0);
     ..........................................................
     buffer.Ref_DebitAer=BitConverter.ToDouble(data, 120);

     mainWindow.Invoke(new UpdateGUIdelegate(MessageReceived), null);
     }
     catch(Exception e){}
     finally{
       Monitor.Exit(buffer);
     }
  }
 }
}
```

Compoentele grafice personalizate au fost realizate prin derivarea clasei UserControl si implementarea functionalitatii specifice. Clasele reprezentative sunt RTChart (afisarea graficelor in timp real) si Bargraph (afisarea graficelor sub forma de bargraph).

Unui obiect de tip RTChart ii trebuie specificata un vector cu valorile ce se doresc afisate pe ecran. La un interval de timp prestabilit, se introduc in vector valorile curente primite de la software-ul de conducere a procesului si se apeleaza metoda AddedValue() pentru desenarea efectiva a graficului.

Metodele principale ale clasei sunt DrawChart() care deseneaza graficul efectiv, DrawBackgroundAndGrid() ce deseneaza fundalul si grila si CalcVerticalPosition() care calculeaza pozitia pe verticala in cadrul componentei, in functie de valoarea maxima a marimii respective si de inaltimea graficului.

```
private void DrawChart(Graphics g)
{
 visibleValues = Math.Min(chartWidth / valueSpacing, drawValues.Count);

 PointF previousPoint = new PointF(Width + valueSpacing, Height);
 PointF currentPoint = new PointF();

 for (int i = 0; i < visibleValues; i++)
 {
  currentPoint.X = previousPoint.X - valueSpacing;
  currentPoint.Y = CalcVerticalPosition(drawValues[i]);

  g.DrawLine(new Pen(lineColor,2), previousPoint, currentPoint);
  previousPoint = currentPoint;
 }
```

```
}

private void DrawBackgroundAndGrid(Graphics g)
{
 Rectangle baseRectangle = new Rectangle(0, 0,Width, Height);
 g.FillRectangle(new LinearGradientBrush(baseRectangle, Color.White,
        Color.Silver, LinearGradientMode.Vertical), baseRectangle);

 // Liniile verticale
 for (int i = Width - gridScrollOffset; i >= 0; i -= GRID_SPACING)
   g.DrawLine(Pens.Gray, i, 0, i, Height);

 // Liniile orizontale
  for (int i = 0; i < Height; i += GRID_SPACING)
   g.DrawLine(Pens.Gray, 0, i, Width, i);

}

private float CalcVerticalPosition(float value)
{
   float result = 0.0f;

   result = value * this.chartHeight / maxValue;

   result = this.chartHeight - result + graphOffset;

   return (float)Convert.ToDouble(Math.Round(result));
}
```

Clasa Bargraph are rolul de a afisa valoarea curenta a unei marimi si referinta impusa de operator sub forma de bare verticale. Metoda de interes in aceasta clasa o reprezinta functia de desenare a componentei, OnPaint(). Daca valoarea curenta a marimii a depasit limitele unui prag impus [*minThreshold*, *maxThreshold*], bagraph-ul va fi desenat folosind culoarea rosie, pentru a alerta operatorul.

```
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    Graphics g = e.Graphics;

    Brush referenceBrush = Brushes.White;
    Brush valueBrush = Brushes.Blue;
    Brush textBrush = Brushes.Black;

    double t = currentValue;
    if (currentValue > maxValue) t = maxValue;
    int valueHeight = Convert.ToInt16(t * HEIGHT / maxValue);

    if (currentValue > maxThreshold || currentValue < minThreshold)
    {
      valueBrush = Brushes.Red;
      textBrush = Brushes.Red;
    }

    Rectangle ValueBar = new Rectangle(5, 230 - valueHeight, 40,
```

```
                                        valueHeight);
    g.FillRectangle(valueBrush, ValueBar);

    Rectangle ReferenceBar = new Rectangle(45,30,3,200);
    g.FillRectangle(Brushes.Black, ReferenceBar);

    Rectangle TitleBar = new Rectangle(0, 0, 100, 20);
    g.FillRectangle(Brushes.White, TitleBar);

    //valoarea curenta
     g.DrawString(currentValue.ToString(), new Font("Verdana", 9,
                            FontStyle.Bold), textBrush1, 5, 240);

     g.DrawLine(Pens.Black, new Point(0, 20), new Point(100, 20));


     g.DrawString(minValue, new Font("Verdana", 9,FontStyle.Bold),
                                    Brushes.Black, 53, 215);
     g.DrawString(maxValue.ToString(), new Font("Verdana",
                        9,FontStyle.Bold), Brushes.Black, 53, 30);
}
```

3. Sistem informatic de conducere – varianta dezvoltata in Matlab-Simulink

3.1Blocul de timp real

Mediul de dezvoltare *Matlab – Simulink* ofera posibilitatea de a utiliza blocuri din biblioteca proprie, precum si de a construi blocuri dedicate aplicatiilor de simulare si control in timp real a proceselor. In felul acesta, utilizand placi de achizitie multifunctionale, achizitia de date si controlul proceselor reale se simplifica considerabil. Un alt avantaj al dezvoltarii sistemelor de conducere in timp real in Matlab – Simulink il constituie faptul ca acesta apare ca un sistem informational structurat, usor de inteles si deschis la extinderi ulterioare. De asemenea, aceste sisteme informationale pot fi verificate si depanate rapid.

In figura 54 este prezentat modul in care blocul de timp real interactioneaza cu restul mediului Matlab.

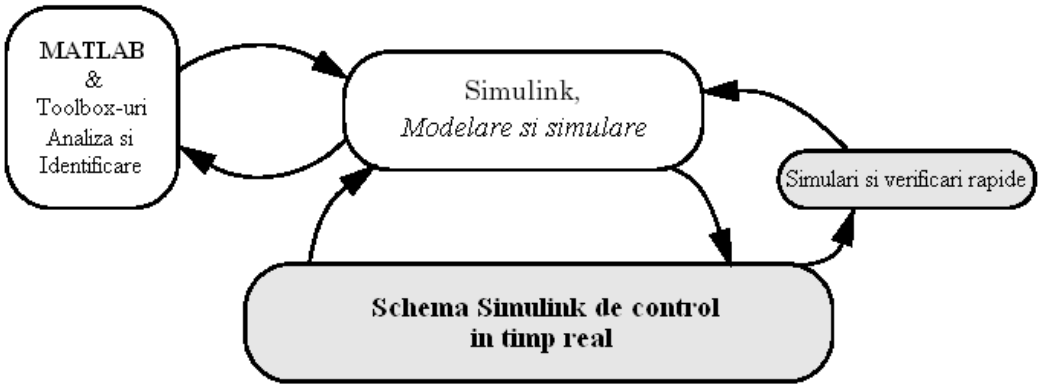


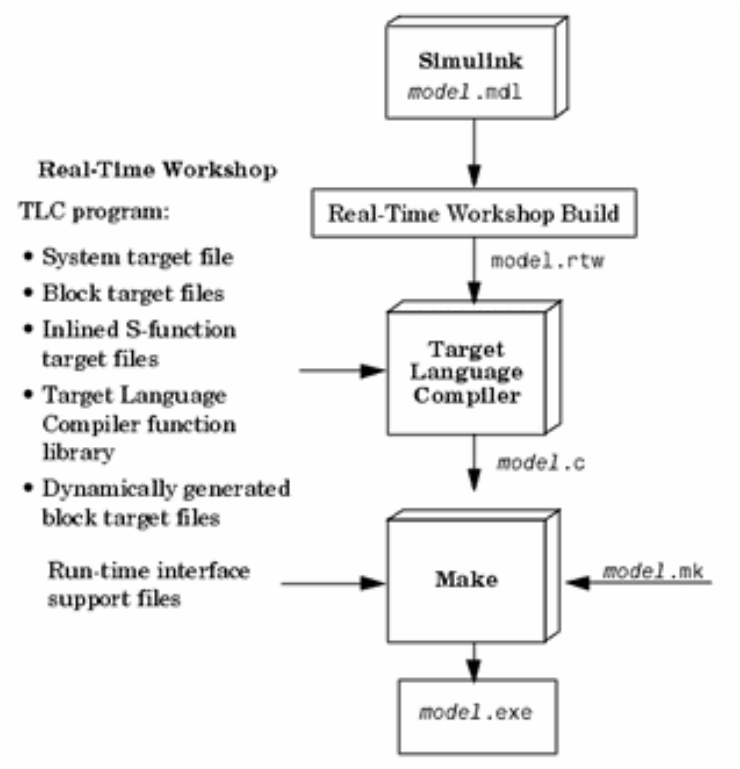
Figura 54: Mediul *Matlab-Simulink* si controlul proceselor in timp real

Pentru executarea in timp real a schemelor *Simulink* pot fi utilizate urmatoarele variante de software:

Real-Time Workshop

Matlab Simulink Real-Time Workshop extrage, genereaza si compileaza cod sursa din modelele *Simulink* pentru a crea aplicatii software de timp real pentru diverse sisteme. *Real-Time Workshop* permite:

- Generarea automata a codului sursa pentru diverse platforme
- O cale rapida si directa de la proiectare la implementare
- Interfata grafica simpla



Real-Time Workshop permite accelerarea procesului de dezvotarea al aplicatiilor software. Procesul de generare a codului sursa din modelele *Simulink* folosind *Real-Time Workshop* este prezentat in urmatoarea diagrama:

Principalele componente si trasaturi ale *Real-Time Workshop* sunt:

- *Simulink Code Generator* – transforma automat modelul *Simulink* in cod *C*
- *Make process* – permite configurarea compilarii codului sursa generat pentru o rapida prototipizare
- *Modul Simulink extern* – permite rularea modelului *Simulink* in timp real
- *Targeting support*
- Simulari rapide
- *Large-scale modeling*

Real-Time Windows Target

Real-Time Windows Target este o solutie pentru testarea si dezvoltarea sistemelor de timp real. Este un mediu de dezvoltare ce foloseste un desktop *PC* cu *MATLAB*® si *Simulink*® pentru a crea modele utilizand blocuri *Simulink*. Dupa crearea unui model si testarea acestuia cu ajutorul *Simulink*-ului se poate genera

cod executabil cu ajutorul programului *Real-Time Workshop*® si a compilatorului *Open Watcom C/C++*, dupa care se poate rula aplicatia in timp real utilizand optiunea "external mode" din *Simulink*. Imbinarea modului extern din *Simulink* si a *Real-Time Windows Target* permite folosirea modelelor *Simulink* pentru controlul proceselor, vizualizarea semnalelor folosind aceleasi blocuri de tip osciloscop din *Simulink* ce sunt folosite pentru simulari “non-real-time”, ajustarea on-line a parametrilor - folosind parametrii din casutele de dialog de ale blocurilor *Simulink* se pot modifica parametrii aplicatiei in timp ce aceasta ruleaza in timp real.

Aplicatii tipice ce folosesc *Real-Time Windows Target* sunt:

- Control de timp real: crearea de prototipuri pentru diferite instalatii, periferice pentru calculatoare, sisteme de control.
- Crearea de controlere tip prototip conectate la procese reale.
- Activitati didactice: proceduri pentru modelare, simulare si testarea sistemelor de timp real.

Mediul de dezvoltare *Real-Time Windows Target* prezinta diferite avantaje pentru testarea si dezvoltarea aplicatiilor de timp real:

- Kernel (nucleu) de timp real - *Real-Time Windows Target* foloseste un kernel de timp real ce utilizeaza ceasul intern al *PC*-ului ca sursa primara:
  - *Timer interrupt* – kernelul primeste intreruperile de la ceasul intern al *PC*-ului inainte ca sistemul de operare *Windows* sa le primeasca. Aceast lucru blocheaza orice apel catre sistemul de operare ceea ce face imposibila utilizarea functiilor *Win32* in codul *C* al functiilor *Simulink S*. Pentru a obtine o perioada de esantionare precisa, kernelul reprogrameaza ceasul *PC*-ului la o frecventa mai inalta.
  - Comunicarea cu hardware-ul – kernelul comunica cu echipamentul *I/O* folosind driverele *I/O*. Blocurile de intrari si iesiri analogice, intrari si iesiri digitale, apeleaza driverele placilor de achizitie. Toate aceste blocuri pot fi configurate separat.
  - *Simulink external mode* – comunicatia intre *Simulink* si aplicatia de timp real se face prin optiunea de simulare externa.
- Analiza si achizitia semnalelor direct cu ajutorul blocurilor *Simulink*
- Ajustarea facila a parametrilor

Bloc de Timp Real pentru Matlab Simulink

O alta solutie pentru a utiliza mediul *Matlab Simulink* in timp real este folosirea unui bloc de timp real. Acest bloc a fost realizat utilizand o *S*-functie *Simulink* scrisa in limbaj *C++*. Blocul este realizat pe simplul concept ca pentru a face o schema *Simulink* sa ruleze cu o temporizarea de timp real, timpul de ciclu (timpul in care *Simulink*-ul are nevoie sa calculeze un pas de simulare, ce depinde de tipul de hardware si de sistemul de operare) sa fie mai scurt decat timpul de simulare dorit. Daca aceasta conditie nu este valabila, simularea in timp real nu este posibila. Acest bloc de timp real nu foloseste un sistem de operare diferit si nu ruleaza un kernel de timp real pentru a furniza o simulare in timp real - executia schemei simulink este oprita daca timpul de ciclu este mai mic decat timpul de simulare, conceptul fiind unul simplu dar foarte eficient, care se preteaza utilizarii in cazul aplicatiei de conducere a statiei de epurare biologica de laborator, datorita faptului ca procesul de epurare este foarte lent.

3.2 **Sistemul de conducere**

Sistemul informatic de conducere a statiei pilot de epurare bilogica este prezentat schematic in figura 55 si este alcatuit din urmatoarele blocuri principale:

- blocul “Bioreactor” ce contine driverele *I/O* pentru placile de achizitie;
- blocul “Comunicatie” contine un bloc dedicat ce trimite si primeste un set de variabile ce constituie comunicatia cu interfata grafica (protocol *UDP*);
- controlere pentru buclele de reactie existente (nivelul de conducere de baza): regulatorul de *pH*, regulatorul de debit de aer si regulatorul de temperatura.
- bucla de generala de reglare prin care se va asigura eficienta procesului de epurare; controlerul general va contine diverse legi de reglare care vor genera comenzi (viteza de dilutie – pompa P_2 si viteza de aerare – electrovalva continua R_2) pentru a mari randamentul procesului de epurare biologica.

Trebuie facuta mentiunea ca aceasta bucla de reglare face obiectul etapei III a proiectului, etapa in care vor fi studiate o serie de legi de conducere avansata in scopul de a imbunatati indicatorii procesului de epurare biologica.

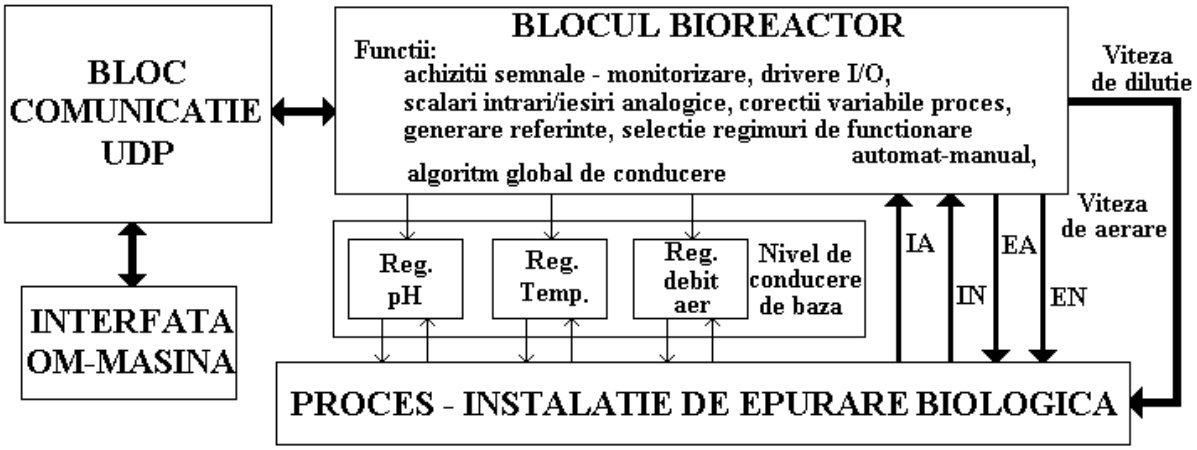


Figura 55: Sistemul informatic de conducere a statiei pilot de epurare

Aceste blocuri se regasesc in schema *Simulink* (Figura 56) care se executa pe calculatorul de proces pentru conducerea statiei pilot in timp real.

In figura 56 se pot observa componentele sub forma blocurilor *Simulink*: blocul „Bioreactor”, blocul „Comunicatie”, precum si cele trei regulatoare (regulatorul pentru debitul de aer, regulatorul de temperatura si cel de reglare a *pH*-ului). In continuare sunt detaliate fiecare din blocurile mentionate anterior.

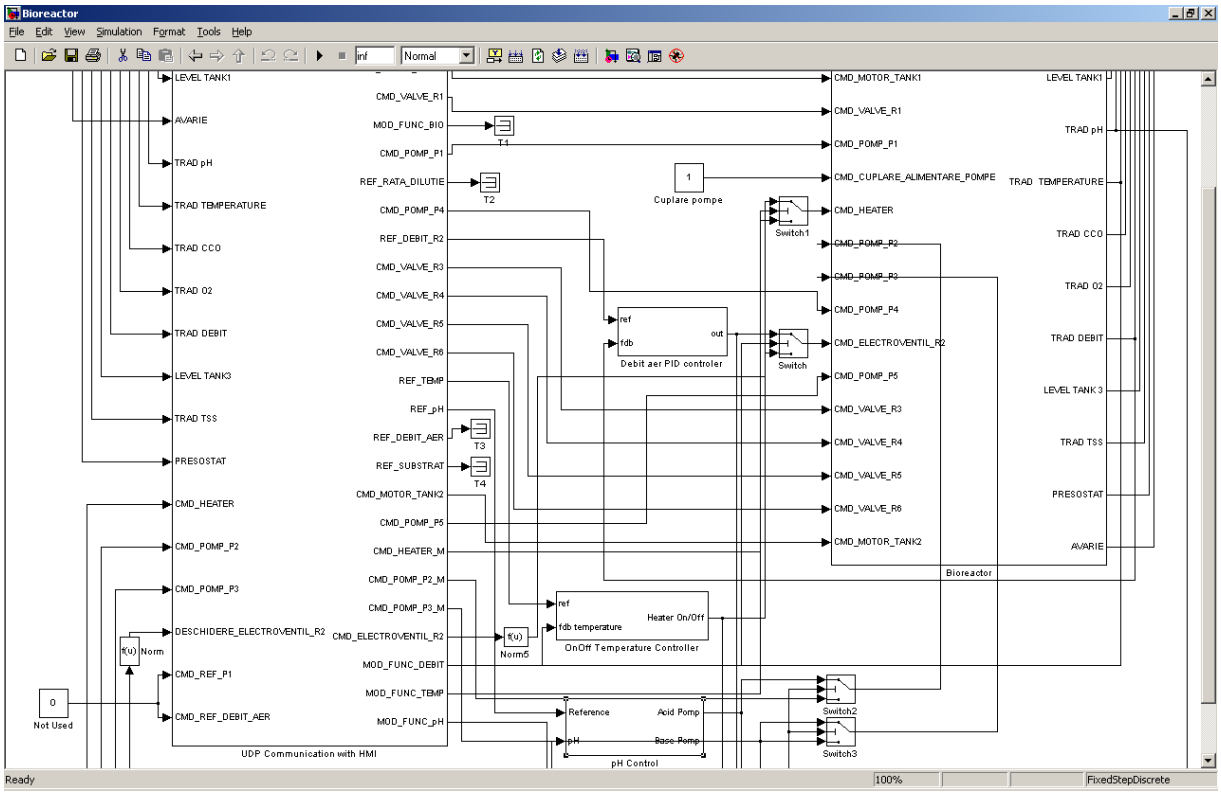


Figura 56: Schema generala *Simulink* a statiei pilot

a). Blocul „**Bioreactor**” (figura 57)

Se prezinta la inceput lista de variabilelor cu care lucreaza sistemul informational de conducere a statiei pilot de epurare (tabelul 14):

Alocare variabile placi de achizitie PC - Bioreactor

Nr. Crt.	MNEMONICA	Denumire parametru	OBSERVATII
	INTRARI ANALOGICE		
1	IA00	Traductor de nivel pentru bazinul de alimentare, **0,15m-1m/4-20mA**	4-20mA => 1-5V la IA PC
2	IA01	Traductor de temperatura pt. masurarea temperaturii in bazinul aerat, **0-120grC/4-20mA**	4-20mA => 1-5V la IA PC
3	IA02	Traductor de pH, **0-14pH/4-20mA**	4-20mA => 1-5V la IA PC
4	IA03	Traductor pentru determinarea consumului chimic de oxigen **-1000mV-+1000mV/4-20mA**	4-20mA => 1-5V la IA PC
5	IA04	Traductor de concentratie de oxigen dizolvat, **0mV-33mV/4-20mA**	4-20mA => 1-5V la IA PC
6	IA05	Traductor debit aer, **0-5l/min/4-20mA**	4-20mA => 1-5V la IA PC
7	IA06	Traductor de nivel pentru decantor, **0,15m-1m/4-20mA**	4-20mA => 1-5V la IA PC

8	IA07	Traductor pentru determinarea suspensiilor solide in decantor, **0-3000ntu/4-20mA**	4-20mA => 1-5V la IA PC
	IESIRI ANALOGICE		
1	EA00	Cda. Pompa **P1** (peristaltica) - alimentarea cu apa de tratat (influent)	Cda U = 0 - 4Vcc la pompe
2	EA01	Cda. Pompa dozatoare **P2** (peristaltica) pentru corectia pH-ului cu acid	Cda U = 0 - 4Vcc la pompe
3	EA02	Cda. Pompa dozatoare **P3** (peristaltica) pentru corectia pH-ului cu baza	Cda U = 0 - 4Vcc la pompe
4	EA03	Cda. Pompa **P4** (peristaltica) pentru alimentarea bazinului aerat	Cda U = 0 - 4Vcc la pompe
5	EA04	Cda. Pompa dozatoare **P5** (peristaltica) pentru recircularea interna bioreactor	Cda U = 0 - 4Vcc la pompe
6	EA05	Cda. **R2**: electroventil cu actionare continua pentru reglarea debitului de aer in bazinul aerat	Cda U = 0 - 5Vcc la ventil
	IESIRI NUMERICE		
1	EN00	Cuplare agitator bazin de alimentare	
2	EN01	Cuplare alimentare pompe dozatoare	
3	EN02	Cuplare rezistenta incalzire bazin aerat	
4	EN03	Cuplare agitator bazin aerat (in treapta I ~60rot/min)	
5	EN04	Cda electroventil alimentare instalatie cu apa de tratat **R1**	
6	EN05	Cda. electroventil on-off pentru alimentarea sistemului de recirculare namol **R3**	
7	EN06	Cda. electroventil on-off pentru pentru evacuarea namolului in exces **R4**	
8	EN07	Cda. electroventil on-off pentru pentru evacuarea apei tratate **R5**	
9	EN08	Cda. electroventil on-off pentru recircularea namolului in bazinul aerat **R6**	
9	EN09	Cuplare agitator bazin aerat (in treapta II ~180rot/min)	
10	EN10	Cuplare agitator bazin aerat (in treapta III ~300rot/min)	
11	EN11	Cuplare agitator bazin aerat (in treapta IV ~420rot/min)	
	INTRARI NUMERICE		
1	IN00	Prezenta presiune aer comprimat	
2	IN01	Oprire de avarie	

Tabel 14 – Alocarea intrarilor/iesirilor analogice/numerice pentru controlul statiei de epurare

In partea stanga a figurii 57 se gasesc iesirile analogice iar in partea dreapta iesirile numerice. Pentru iesirile analogice (6 la numar) au fost realizate blocuri de normare, blocuri de limitare valori, casete de vizualizare a valorilor, la dispozitia proiectantului, pentru depanare programe si eventuale verificari, blocuri de oprire a pompelor peristaltice dupa o rampa descrescatoare (a se vedea specificatia de

functionare a pompelor, prezentata in cadrul activitatii II.2) si driverele de iesire. Toate semnalele analogice de iesire sunt conditionate prin blocuri *SI* de semnalul de *AVARIE* (la apasarea butonului de avarie toate semnale trec in starea zero. Iesirile numerice sunt si ele conditionate de semnalul *AVARIE* si sunt toate aplicate driver-ului iesirilor numerice.

Intrarile analogice si cele numerice sunt tratate in figura 58.

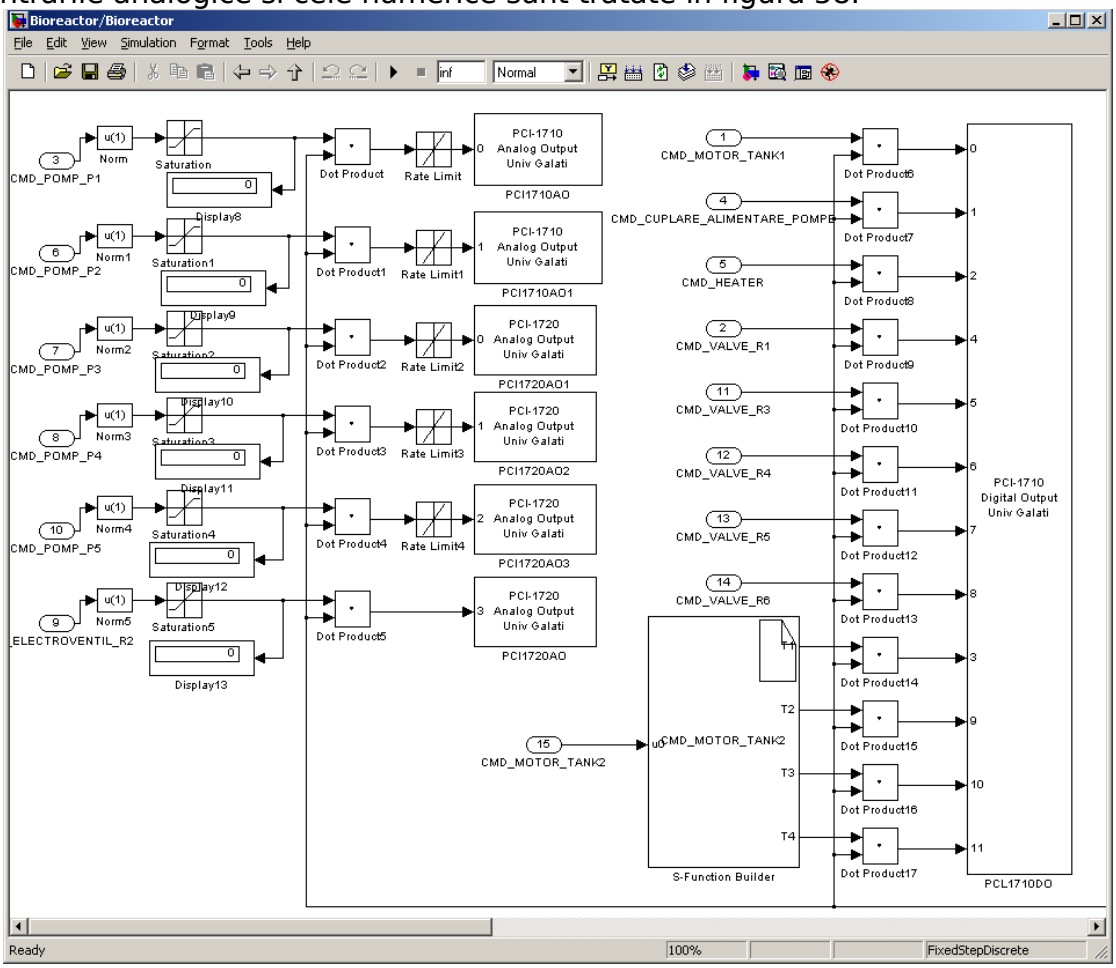


Figura 57: Blocul *Simulink Bioreactor* – iesiri analogice si iesiri digitale

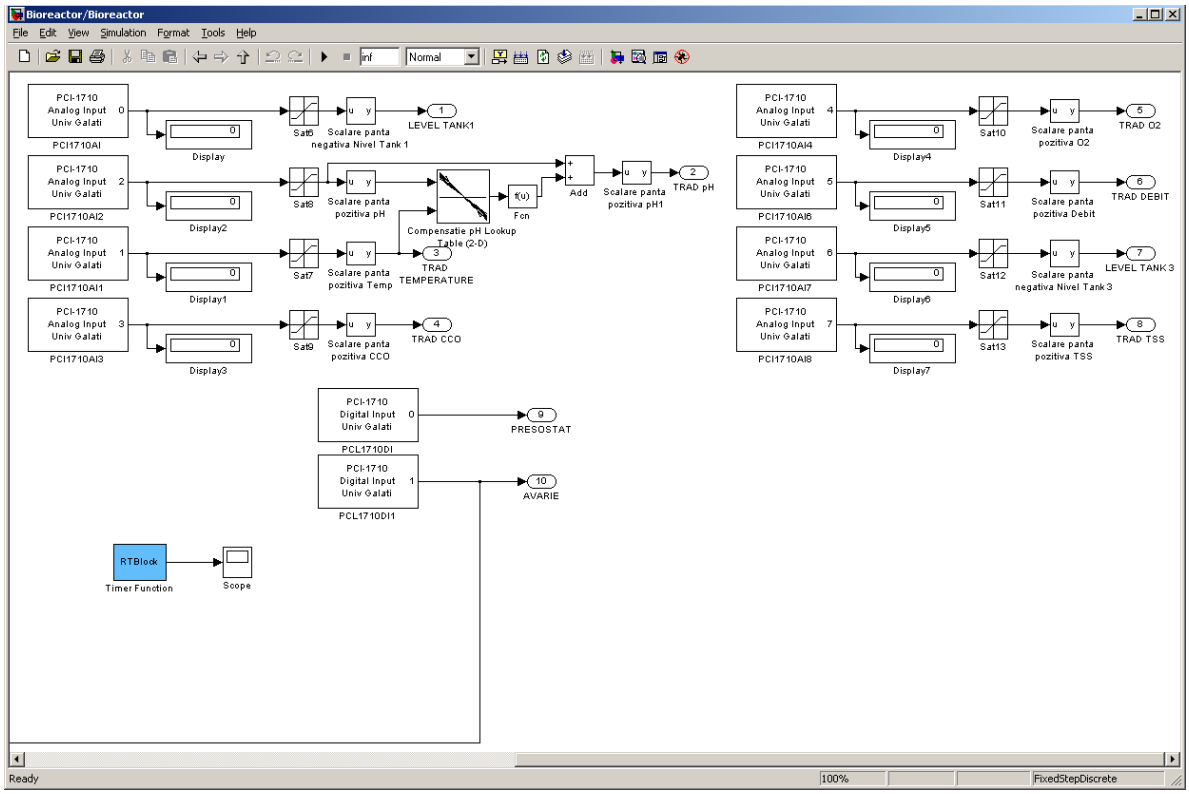


Figura 58: Blocul *Simulink Bioreactor* – intrari analogice si intrari digitale

Sunt in total 8 intrari analogice si doua iesiri numerice. Pentru iesirile analogice au fost create driverele de iesiri analogice, blocuri de limitare a valorilor, blocuri de scalare si casetele cu valorile numerice pentru verificare-depanare programe. O mentiune speciala trebuie facuta pentru achizitia *pH*-ului, in sensul ca aici exista in plus un bloc de corectie a *pH*-ului cu temperatura (Bloc de compensatie *pH*). Pentru intrarile numerice au fost create drivere speciale.

Drivere *I/O* pentru placile de achizitie *PCI-1710* si *PCI-1720*

Driverele *I/O* pentru placile de achizitie *PCI-1710* si *PCI-1720* au fost create utilizand *S-functii* scrise in *C++* si au fost incorporate intr-o biblioteca numita *PCI1710* care contine urmatoarele blocuri (figura 59):

- *PCI-1720 Analog Output*
- *PCI-1710 Analog Output*
- *PCI-1710 Analog Input*
- *PCI-1710 Digital Input*
- *PCI-1710 Digital Output*

Fiecare intrare/iesire analogica/digitala se poate regasi in unul din aceste blocuri. Configurarea intrarii/iesirii respective se face prin dublu-click si prin schimbarea parametrilor din fereastra corespunzatoare (figurile 60 si 61).

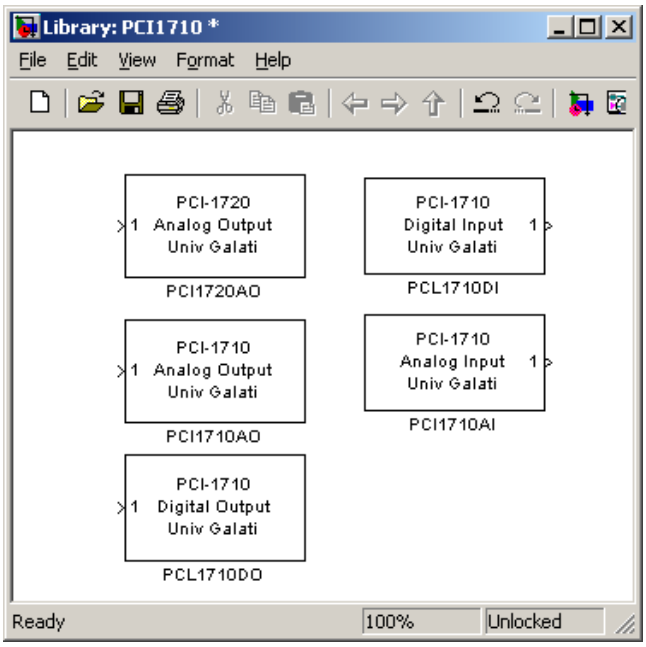


Figura 59: Drivere placi de achizitie *PCI-1720* si *PCI-1710*

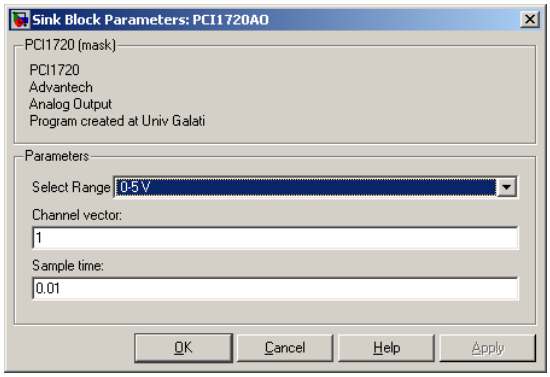


Figura 60: Configurarea unei *EA*

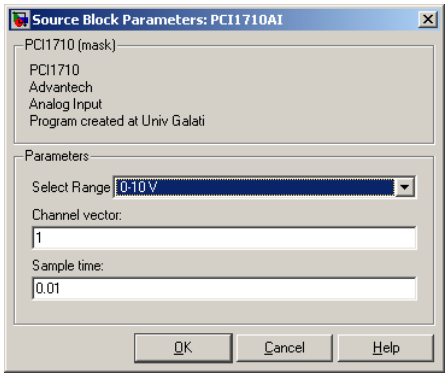


Figura 61: Configurarea unei *IA*

Cu ajutorul acestor blocuri se pot conecta diferite procese la aplicatia de timp real in *Simulink*:

- senzorii, traductoarele si elementele de executie sunt conectate la placile *I/O* de achizitie;
- placile *I/O* de achizitie fac conversia tensiune – unitati ingineresti si invers;
- valorile numerice sunt citite sau scrise la placile *I/O* de catre driverele *I/O*.

b). Blocul „**Comunicatie**” cu interfata om-masina

Acest bloc realizeaza transmiterea unui bloc de cuvinte intre sistemul de conducere si interfata om-masina, continand toate semnalele masurate sau comenzile catre elementele de executie. La fiecare 250 msec., indiferent daca s-a schimbat ceva in starea sistemului, sistemul de conducere transmite pachetul de informatii sistemului de interfata om-masina, dupa care primeste un pachet de raspuns. Acest protocol este un protocol destinat aplicatiilor care doresc sa utilizeze propria secventiere si control al fluxului informational si nu mecanismele asigurate de *TCP* (*Transmission Control Protocol*). Spre deosebire de *TCP*, *UDP* asigură un nivel mai redus de control al erorilor fiind protocolul preferat pentru transmiterea mesajelor de tip „broadcast”. In general este un protocol folosit in aplicatii pentru care comunicarea rapida este foarte importanta.

Continutul celor doua mesaje de comunicatie, *HMI* (*Human Man Interface*) - *Simulink* si cel *Simulink - HMI*, sunt descrise in tabelele 15 si 16.

CONTINUTUL MESAJULUI UDP HMI-SIMULINK						
Pos	Tip	Nume	Descriere	Valoare fizica minima	Valoare fizica maxima	Descriere valori
1	DOUBLE	CMD_MOTOR_TANK1	Comanda agitare bazin alimentare	0	1	1 = ON (motor pornit) 0 = OFF (motor oprit)
2	DOUBLE	CMD_VALVE_R1	Comanda electrovalva R1	0	1	1 = ON (valva deschisa) 0 = OFF (valva inchisa)
3	DOUBLE	MOD_FUNC_BIO	Mod functionare Bioreactor	0	1	Not used
4	DOUBLE	CMD_POMP_P1	Comanda pompa P1	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
5	DOUBLE	REF_RATA_DILUTIE	Referinta rata dilutie	0	100	Not used
6	DOUBLE	CMD_POMP_P4	Comanda pompa P4	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
7	DOUBLE	REF_DEBIT_R2	Referinta debitului de aer prin R2	0	5	Referinta controler debit aer – litri
8	DOUBLE	CMD_VALVE_R3	Comanda electrovalva R3	0	1	1 = ON (valva deschisa) 0 = OFF (valva inchisa)
9	DOUBLE	CMD_VALVE_R4	Comanda electrovalva R4	0	1	1 = ON (valva deschisa) 0 = OFF (valva inchisa)
10	DOUBLE	CMD_VALVE_R5	Comanda electrovalva R5	0	1	1 = ON (valva deschisa) 0 = OFF (valva inchisa)
11	DOUBLE	CMD_VALVE_R6	Comanda electrovalva R6	0	1	1 = ON (valva deschisa) 0 = OFF (valva inchisa)
12	DOUBLE	REF_TEMP	Referinta controlerului de temperatura	0	120	Referinta pentru controlerul de temperatura a apei - grade Celsius
13	DOUBLE	REF_pH	Referinta controlerului de pH	0	14	Referinta pentru controlerul de pH
14	DOUBLE	REF_DEBIT_AER	Referinta controlerului de debit aer	0	100	Not used
15	DOUBLE	REF_SUBSTRAT	Referinta substrat	0	100	Not used
16	DOUBLE	CMD_MOTOR_TANK2	Comanda agitator bioreactor	0	420	0 = Agitator oprit 60 = Motor agitator treapta I 180 = Motor agitator treapta II 300 = Motor agitator treapta III 420 = Motor agitator treapta IV
17	DOUBLE	CMD_POMP_P5	Comanda pompa P5	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
18	DOUBLE	CMD_HEATER_M	Comanda a rezistentei de incalzire a apei	0	1	0 = Rezistena incalzire bioreactor ON 1 = Rezistenta incalzire bioreactor OFF
19	DOUBLE	CMD_POMP_P2_M	Comanda pompa P2	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
20	DOUBLE	CMD_POMP_P3_M	Comanda pompa P3	0	4	0 = Pompa oprita 1 = Pompa pornita 1%

						4 = Pompa pornita 100%
21	DOUBLE	CMD_ELECTROVENTIL_R2	Comanda electroventil R2	0	100	0 – Electroventil inchis 100 – Electroventil deschis la maxim
22	DOUBLE	MOD_FUNC_DEBIT	Mod functionare control debit aer	0	1	0 = Control debit de aer in mod manual 1 = Control debit de aer automat
23	DOUBLE	MOD_FUNC_TEMP	Mod functionare control temperatura	0	1	0 = Control temperatura in mod manual 1 = Control temperatura automat
24	DOUBLE	MOD_FUNC_pH	Mod functionare control pH	0	1	0 = Control pH in mod manual 1 = Control pH automat

Tabelul 15: Mesajul *UDP HMI - Simulink*

CONTINUTUL MESAJULUI UDP SIMULINK-HMI						
Pos	Tip	Nume	Descriere	Valoare fizica minima	Valoare fizica maxima	Descriere valori
1	DOUBLE	LEVEL_TANK1	Nivelul apei in tancul 1	0	100	Nivel apa – cm
2	DOUBLE	AVARIE	Semnal de avarie	0	1	0 = Buton avarie ON 1= Buton avarie OFF
3	DOUBLE	TRAD_pH	Valoare pH	0	14	pHul in bioreactor - unitati pH
4	DOUBLE	TRAD_TEMPERATURE	Valoare temperatura	0	120	Temperatura apei - grade Celsius
5	DOUBLE	TRAD_CCO	Valoare CCO	-1000	1000	Consum chimic oxigen - mV
6	DOUBLE	TRAD_O2	Valoare O2	0	33	Concentratie oxigen dizolvat – mV
7	DOUBLE	TRAD_DEBIT	Valoare debit aer	0	5	Debit aer – litri
8	DOUBLE	LEVEL_TANK3	Nivelul apei in tancul 3	0	100	Nivel apa – cm
9	DOUBLE	TRAD_TSS	Valoare TSS	0	3000	Suspensii solide in decantor - ntu
10	DOUBLE	PRESOSTAT	Presostat	0	1	0 = lipsa presiune 1= prezenta presiune
11	DOUBLE	CMD_HEATER	Valoare comanda rezistenta incalzire data de controlerul de temperatura	0	1	0 = Rezistena incalzire bioreactor ON 1 = Rezistenta incalzire bioreactor OFF
12	DOUBLE	CMD_POMP_P2	Valoarea pompei P2 data de controlerul de pH	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
13	DOUBLE	CMD_POMP_P3	Valoarea pompei P3 data de controlerul de pH	0	4	0 = Pompa oprita 1 = Pompa pornita 1% 4 = Pompa pornita 100%
14	DOUBLE	DESCHIDERE_ELECTROVENTIL_R2	Valoare deschidere electroventil R2	0	100	0 – Electroventil inchis 100 – Electroventil deschis la maxim
15	DOUBLE	CMD_REF_P1	Valoarea comenzii referintei P1	0	100	Not Used
16	DOUBLE	CMD_REF_DEBIT_AER	Valoare debit aer	0	100	Not Used

Tabelul 16: Mesajul *UDP Simulink – HMI*

c). **Blocurile de reglare**

1. Regulatorul de *pH* – figura 62

Este prezentat in figura 62. Regulatorul comanda atat pompa de acid, cat si pe cea de baza. El are o constructie speciala, datorita faptului ca este prevazut cu o zona de insensibilitate a erorii, astfel incat sa nu lucreze continuu, provocand astfel compromiterea experimentului. In esenta, regulatorul este de tip *PI-discret*, atat pentru componenta *acid*, cat si pentru cea de *baza* (figura 63). Cele doua functii de transfer in discret (transformata *z*), pentru acid, respectiv baza, sunt urmatoarele:

$$H_a(z) = K_{p\_a} \frac{1 + \frac{T_e}{2T_{i\_a}} + \left(\frac{T_e}{2T_{i\_a}} - 1\right) z^{-1}}{1 - z^{-1}}$$

$$H_b(z) = K_{p\_b} \frac{1 + \frac{T_e}{2T_{i\_b}} + \left(\frac{T_e}{2T_{i\_b}} - 1\right) z^{-1}}{1 - z^{-1}}$$

Valorile numerice ale parametrilor sunt urmatoarele: K_{p_a}=-1, T_{i_a}=100, K_{p_b}=1, T_{i_b}=100 , T_e=250msec.

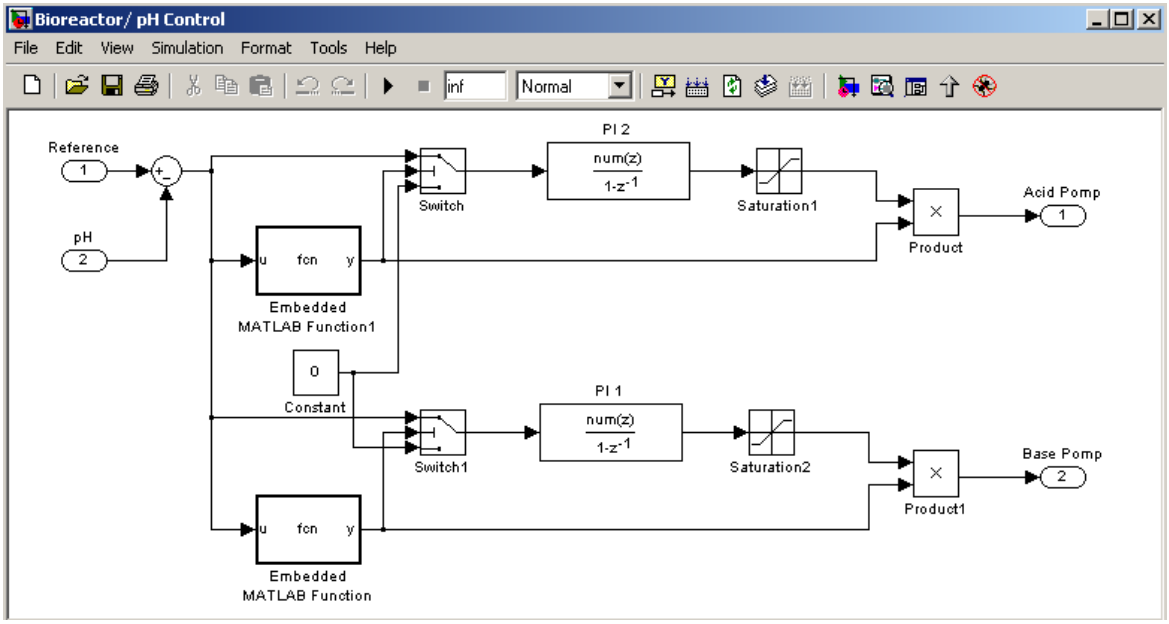


Figura 62: Regulatorul de *pH*

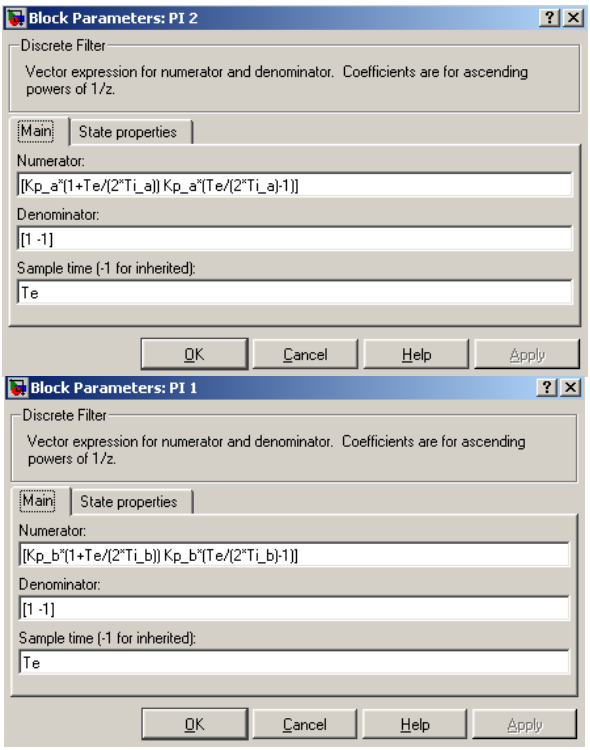


Figura 63: Functia de transfer a regulatorului de *pH* (indicele a – acid, b – baza)

In figura 64 sunt prezentate cele doua functii care realizeaza zona de insensibilitate pentru comanda *pH*-ului (daca eroarea de *pH* este cuprinsa intre valorile -0.15 si 0.15, comanda pompelor de acid sau baza are valoarea 0)

```
function y = fcn(u)
%y = u;
if u < -0.15
    y = 1;
else
    y = 0;
end
```

```
function y = fcn(u)
%y = u;
if u > 0.15
    y = 1;
else
    y = 0;
end
```

Figura 64: Functii pentru a realiza o zona de insensibilitate a regulatorului de *pH* atunci cand eroarea de *pH* este cuprinsa intre valorile -0.15 si 0.15

2. Regulatorul de temperatura – figura 65

Reglarea temperaturii se face printr-un regulator simplu, de tip bipozitional, deoarece procesul de epurare nu necesita conditii dure de temperatura, tehnologiile de epurare admitand variatii de temperatura de +/- 5 gradeC in bazinul aerat, lucru realizabil cu o reglare bipozitionala („on-off”). Trebuie mentionat faptul ca in bazinul aerat se face o corectie a temperaturii continutului acestuia, in prealabil apa de epurat, care vine din bazinul de alimentare (cu o temperatura mentinuta intre 2-4 gradeC) fiind preincalzita la temperatura de 30 gradeC.

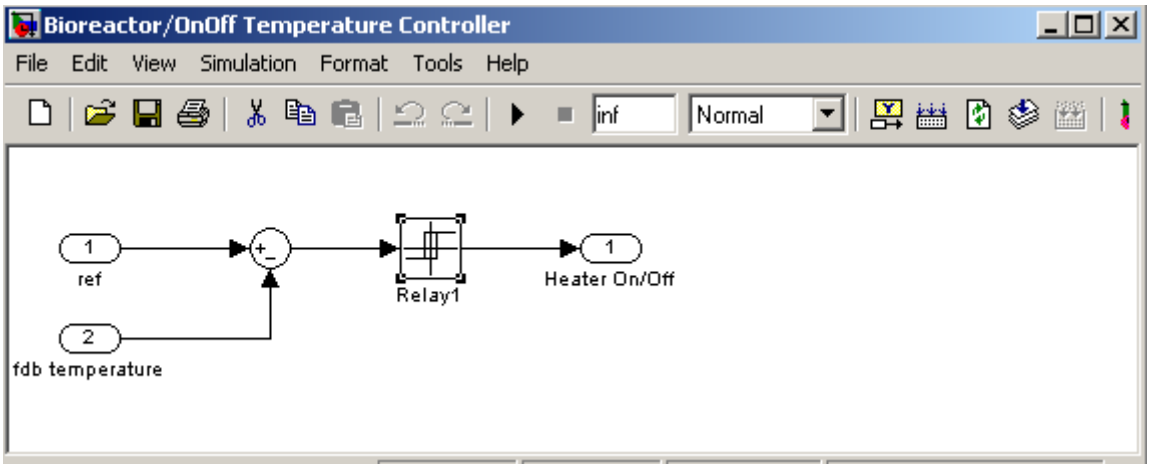


Figura 65: Schema *Simulink* a regulatorului de temperatura

3. Regulatorul de debit aer – figura 66

Reglarea debitului de aer este necesara asigurarii oxigenarii continutului bazinului de epurare, procesul fiind aerob. S-a implementat un regulator *PID-numeric*, cu o componenta derivativa foarte mica, cu rol de a face mai rapida comanda catre electrovalva cu deschidere continua, R_2.

Functia de transfer a regulatorului este urmatoarea:

$$H(z)=\frac{K_p+K_i\frac{t}{2}+2\frac{K_dT_d}{t+2T_d}+\left[-K_p\frac{2T_d}{t+2T_d}+K_i\frac{t^2}{t+2T_d}-\frac{4K_dT_d}{t+2T_d}\right]z^{-1}+\left[\frac{t-T_d}{t+T_d}\left(-K_p+\frac{K_it}{2}\right)+\frac{2K_dT_d}{t+2T_d}\right.}{1-\frac{4T_d}{t+2T_d}z^{-1}-\frac{t-2T_d}{t+2T_d}z^{-2}}$$

cu valorile numerice ale parametrilor: K_p=0.2, K_i=0.15, K_d=0.75, T_d=0.006, t=250msec.

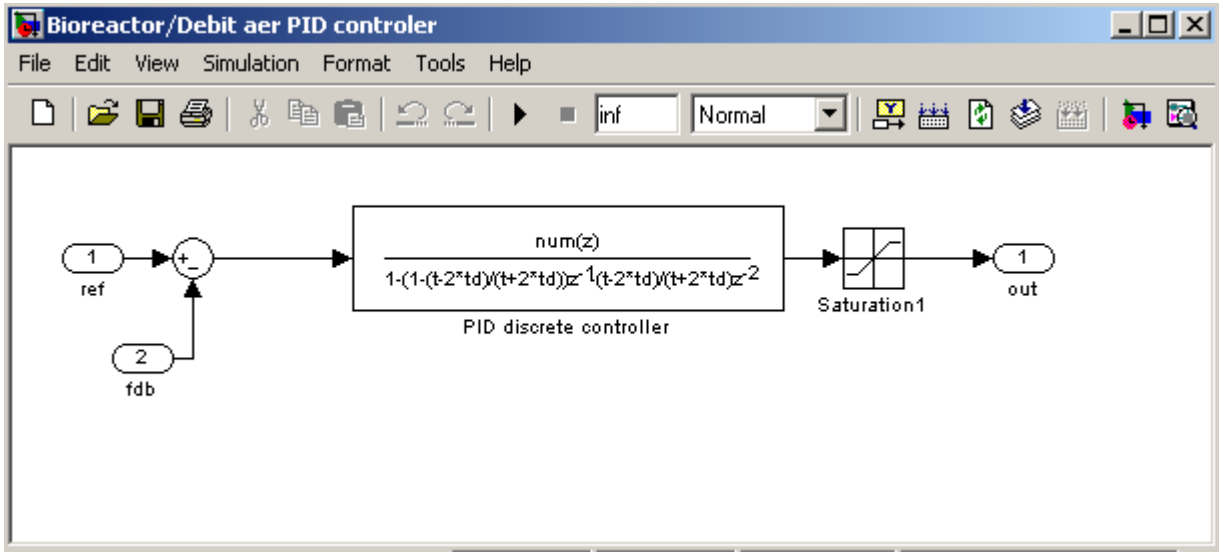


Figura 66: Schema *Simulink* a regulatorului de debit de aer

4. Sistem informatic de conducere – varianta dezvoltata sub sistemul de operare Linux

4.1Architectura aplicatiei de timp real

Prezentare generala

Aplicaţia de conducere este împărţită în două componente: aplicaţia de timp real şi interfaţa grafică. Componentele funcţionează pe echipamente de calcul separate fiind legate intre ele în reţeaua locală.

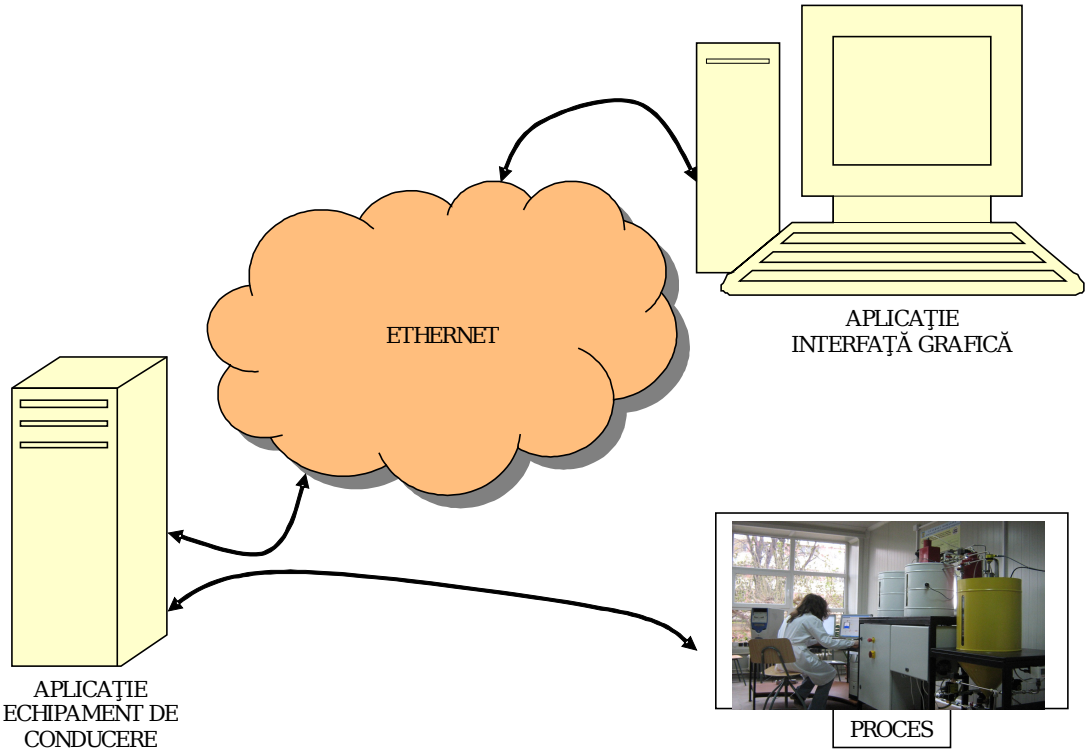


Figura 67: Aplicaţia distribuită de conducere

In continuare se vor prezenta aspectele de implementare a structurii informatice ale acestei variante de software, facandu-se referire la aspectele specifice conducerii instalatiei de epurare biologica, aratate pe larg in sectiunea 3. Ele sunt absolut similare, functiile sistemului de conducere fiind aceleasi in ambele variante de realizare din punct de vedere informatic.

Aplicaţia de timp real are rolul de a comanda modulele staţiei pilot conform cu cerinţele de timp şi robusteţe ale sistemului de conducere. Aplicaţia ce implementează interfaţa grafică are rolul de a oferi posibilitatea operatorului de a interveni in operaţiile de conducere.

Aplicaţia de timp real are ca baza nucleul (kernelul) de *Linux*, exploatând funcţionalităţile de multi-tasking şi partajare a resurselor proceselor.

Arhitectura aplicaţiei urmăreşte următoarele aspecte:

- **Timpul de execuţie** – toate operaţiile de reglare automată/conducere se efectuează într-un timp maxim definit de tehnolog **(250ms**).
- **Robusteţea arhitecturii** – în condiţiile în care un modul al aplicaţie nu funcţionează conform specificaţiilor, acesta nu trebuie să afecteze restul sistemului de control şi conducere.
- **Scalabilitatea** – aplicaţia sa poată fi portată cu uşurinţă pe orice arhitectură *UNIX*. Modulele componente ale aplicaţiei de timp real să poate fi dezvoltate în orice limbaj de programare.

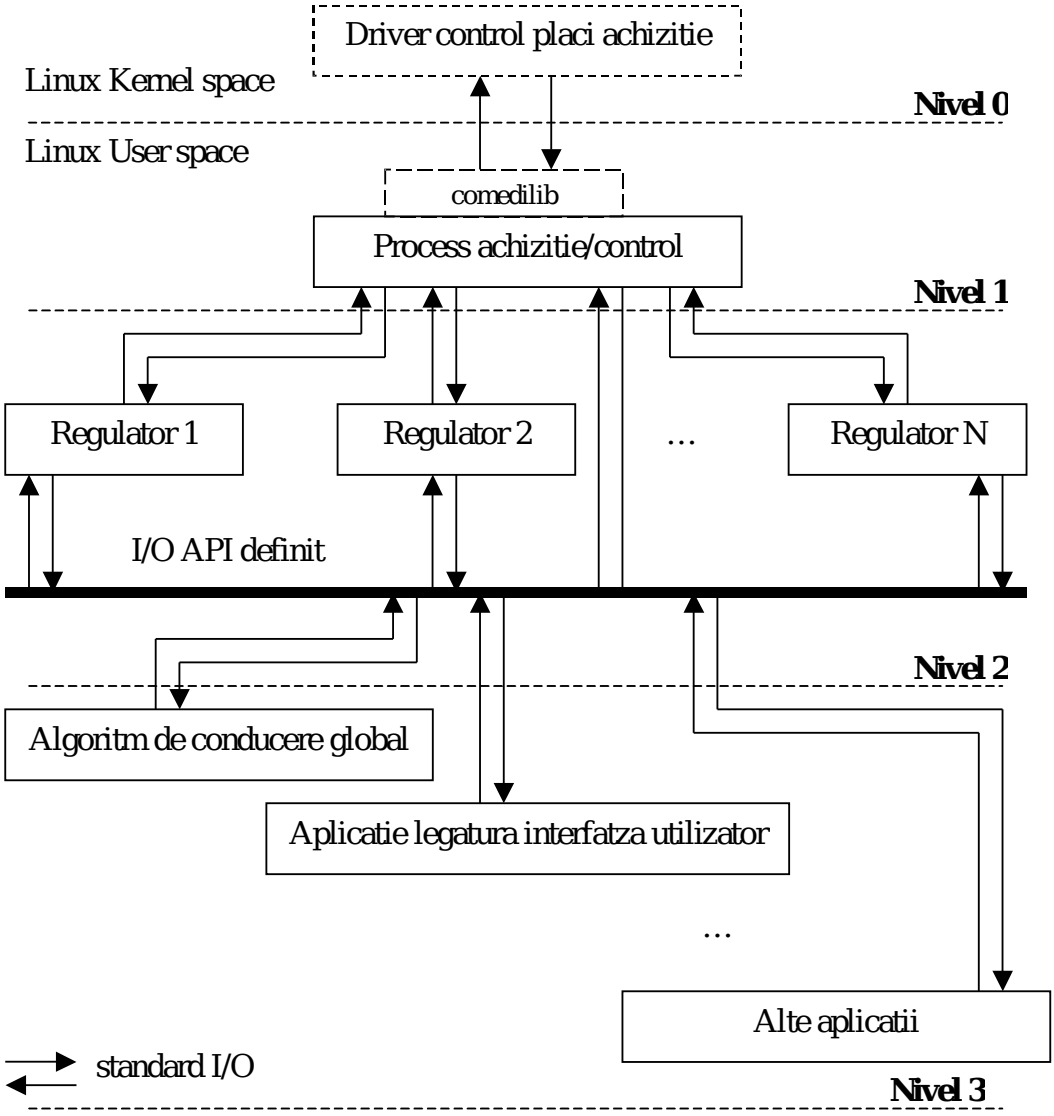


Figura 68: Arhitectura aplicatiei

Pentru a atinge obiectivele sus menţionate, vor fi luate în considerare următoarele constrângeri:

- Modulele ce implementează funcţionalităţi ale aplicaţiei vor rula ca procese separate. Acestea vor avea spaţiul de date separat astfel încât erorile de funcţionalitate sau coruperile de date sa nu afecteze decât modulul defect;
- Comunicaţia între modulele aplicaţiei se face prin interfaţa standard *I/O*;
- Ordinea intrării în execuţie a proceselor va fi gestionată de sistemul de operare prin specificarea de priorităţi ale proceselor.

Aplicaţia de timp real este împărţită în 4 nivele de funcţionalitate după cum urmează:

- Nivelul 0 corespunde driverelor plăcilor de achiziţie. Acestea rulează în spaţiul protejat de kernel, accesul către ele făcându-se prin biblioteca *comedilib*.
- Nivelul 1 corespunde procesului ce realizează achiziţia şi setarea datelor pe plăcile de achiziţie. Acesta are scopul de a dirija fluxul de date intre procesele aplicaţiei şi driverele plăcilor de achiziţie **CONFORM PRINCIPIILOR DE REGLARE**.
- Nivelul 2 corespunde regulatoarelor. Acestea comunica atât cu procesul de achiziţie, în scopul achiziţionării de date şi setării de mărimi de comanda, cat şi cu procesele de pe nivelul imediat superior, în scopul setării referinţelor, a

pornirii, respectiv opririi regulatorului. Accesul se face prin interfaţa standard *I/O*.

- Nivelul 3 reprezintă nivelului de aplicaţie. Pe acest nivel exista procesul ce implementează atât algoritmul de comanda şi control cat şi aplicaţia ce realizează legătura cu interfaţa utilizator.

Fiecare modul prezentat în (Figura 68) este reprezentat de un proces *Linux*. Priorităţile de execuţie a proceselor sunt stabilite astfel încât un proces de pe un nivel superior să nu ruleze cu o prioritate mai mică decât un proces de pe nivel inferior şi cea mai mare prioritate din arhitectura sa fie mai mica decât orice alta aplicaţie ce rulează pe sistem.

Procesul de achiziţie şi control, definit în continuare *PAC*, de pe nivelul 1, este singurul care comunica cu driverele plăcilor de achiziţie prin utilizarea bibliotecii *comedilib*. Acest modul rulează prioritar fată de celelalte procese, având scopul de gestiona fluxul de date din sistem.

Procesele de pe nivelul doi au scopul de a implementa regulatoarele sistemului. Ele sunt conectate la procesul de achiziţie prin canale *FIFO* şi comunica prin protocolul descris in sectiunea urmatoare. Fiecare asemenea proces are deschis un canal pe care primeşte date conform protocolului prin care se pot opera schimbări asupra stării regulatorului.

Procesele de pe nivelul trei au scopul de a asigura legătura cu interfaţa utilizator şi de conducere a procesului.

4.2Comunicaţia interproces

Procesele comunica intre ele prin operatii standard *I/O* (*open*, *read*, *write*, *close*) peste canale *FIFO*, pe baza unui *API* predefinit. Fiecare proces deschide un canal de comunicatie prin care poate primi comenzi sau raspunsuri la comenzi în coformitate cu protocolul de comunicaţie. Acesta poate fi împărţit in două componente:

- nivelul de comunicaţie – descris de biblioteca *ipcomp*;
- nivelul de date – specific aplicaţiei

4.2.1 Biblioteca *ipcomp*

Biblioteca *ipcomp* implementează cadrul generic de funcţionare a comunicaţiei între procesele aplicaţiei. Fiecare proces foloseşte biblioteca *ipcomp* pentru a deschide un canal de comunicaţie prin care va primi mesaje de la alte procese sau pentru a comunica mesaje către alte aplicaţii. La iniţializarea bibliotecii, in cadrul procesului este pornit un task ce asculta mesaje venite pe un canal de comunicaţie predefinit. La recepţia unui mesaj valid taskul apelează o funcţie a procesului ataşată comenzii din mesaj.

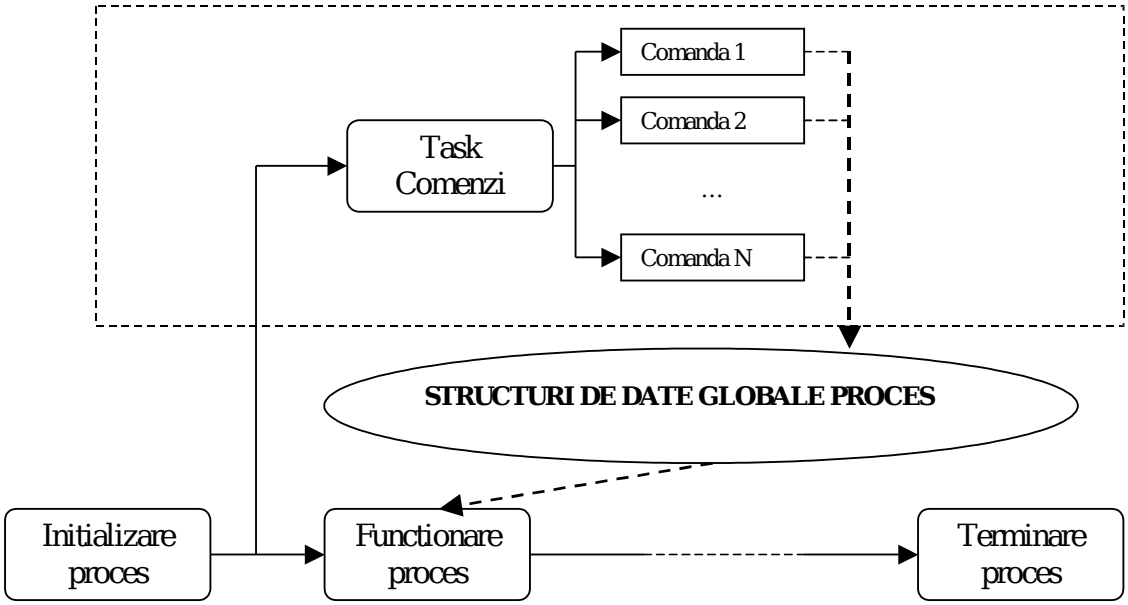


Figura 69: Functionarea comunicatiei în cadrul procesului

Comunicarea se realizează între procese prin operaţii read/write peste fişiere *FIFO*. Datele comunicate se numesc mesaje şi sunt formate din header (tabelul 17) (nivelul de comunicaţie) şi un bloc de date (nivelul de date) prin care pot fi comunicaţi parametrii comenzii.

Crt.	Denumire	Tip	Dimenisune	Functionalitate
1.	uiMagicNumber	UINT32	1	ID. mesaj *ipcomp*
2.	uiMessageSize	UINT32	1	Dimensiune mesaj
3.	uiCrc32	UINT32	1	CRC mesaj
4.	uiCommand	UINT32	1	comanda
5.	szReturnPipeName	char	128	Canalul procesului ce a iniţiat mesaj

Tabelul 17: Header mesaj *FIFO*

Mesajul conţine informaţii privind procesul sursa, prin numele canalului de comunicaţie (*szReturnPipeName*), dimensiunea pachetului (*uiMessageSize*), suma de control a blocului de date (*uiCrc32*) şi identificatorul comenzii (*uiCommand*). Aceasta va fi numita în continuare *comanda inter-proces*. Headerul este urmat de blocul ce conţine date legate de comanda (denumite în continuare *parametri comanda*), dimensiunea acestuia fiind cea din câmpul *uiMessageSize* al structurii.

Deschiderea unui canal de comunicaţie de către un proces se face prin apelul funcţiei *ipcompStart* având ca parametru numele fişierului (*FIFO*) ce va reprezenta canalul de comunicaţie peste care sunt efectuate operaţiile recepţie de mesaje.

Funcţia *ipcompRegister* este folosita pentru a adaugă funcţionalitate la recepţia unei comenzi. Funcţia primeşte ca parametrii identificatorul conexiunii generat de *ipcompStart*, comanda care trebuie tratata şi numele funcţiei care va trata evenimentul.

Funcţia este definita de tipul *IPCOMP_FUNC*, descris în headerul bibliotecii şi primeşte ca parametrii identificatorul conexiunii, pointerul către sursa comenzii şi pointerul către *parametrii de comandă*.

4.2.2 **Comunicaţia cu nivelul 1**

Pe acest nivel rulează procesul de achiziţie şi control care implementează principiul de achiziţie şi comanda, asigurând restricţiile de timp şi fluxul de date între procese. Comunicaţia între nivelul 1 şi canalele de nivel superior este definita ca standard şi este realizat conform cu diagrama din figura 70.

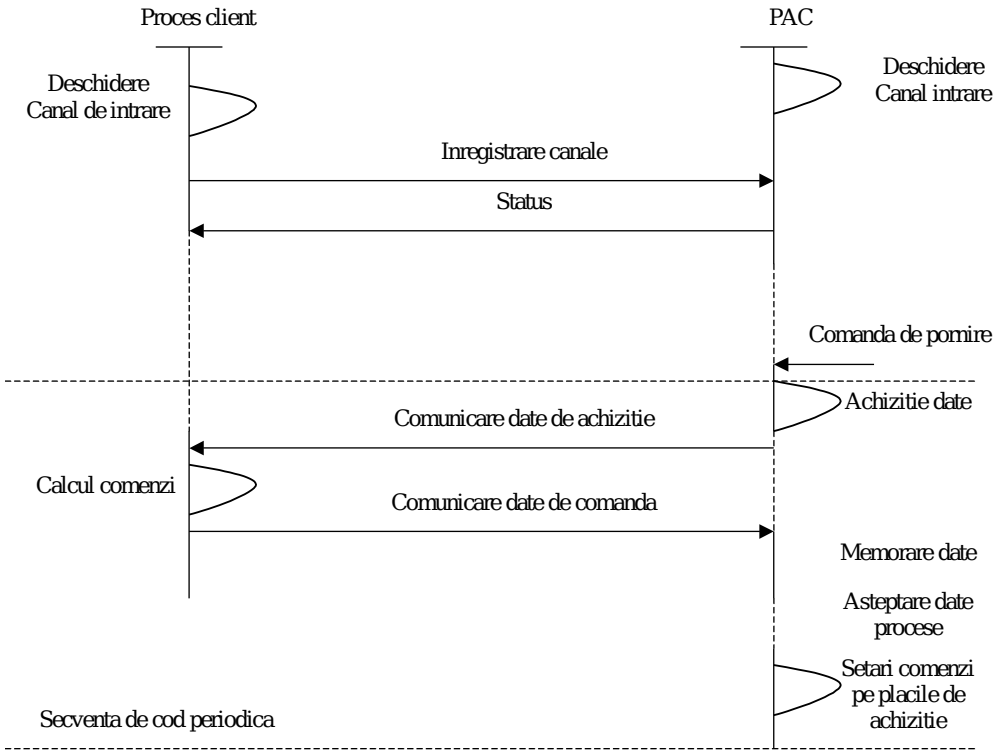


Figura 70: Comunicaţia cu procesul de achiziţie şi control

Comunicaţia cu *PAC* se realizează în doua etape:

1. Configurarea – etapa în care procesele client, definite în continuare *PC,* comunică informaţii privind datele de achiziţie şi comandă.
2. Comunicarea datelor de achiziţie – *PAC* primeşte comanda inter-proces de pornire. Acesta comunica datele de achiziţie proceselor client conform configuraţiei primite la pasul anterior şi primeşte comenzile provenind de la regulatoare.

PAC conţine două taskuri:

1. Taskul ce realizează comunicaţia cu celelalte procese - implementat de biblioteca *ipcomp*;
2. Taskul ce realizează achiziţia, comunicarea şi setarea comenzilor pe / de pe plăcile de achiziţie.

Cele doua taskuri comunica între ele printr-un set de structuri globale şi semafoare conform diagramei de mai jos.

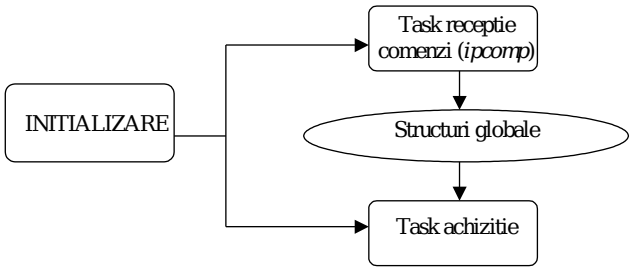


Figura 71: Taskurile *PAC*

Acest proces trebuie sa se execute prioritar raportat la arhitectura sistemului de conducere. Motivul este generat de faptul ca realizarea comunicaţiei cu procesele client se face în ordinea prioritarii la execuţie a acestora. Funcţionalitatea dorită este următoarea:

- *PAC* primeşte comanda inter-proces de pornire;
- *PAC* realizează achiziţia datelor;
- *PAC* realizează comunicarea datelor achiziţionate către procesele client prin intermediul canalelor *FIFO* deschise de acestea;
- *PAC* aşteaptă rezultatele comenzilor regulatoarelor de la procesele client;
- *PC* citesc datele de intrare de pe canalele *FIFO*, realizează calculele necesare generării comenzilor şi le comunica către *PAC*. Ordinea intrării în execuţie a *PC* este condiţionata de prioritatea de execuţie a proceselor.
- *PAC* setează valorile de ieşire pe plăcile de achiziţie;
- *PAC* reia operaţiile de mai sus atât timp cit nu este suspendat prin comanda inter-proces de oprire.

Procesul de achiziţie şi control implementează următorul set de comenzi inter-proces:

Tabelul 18: Comenzi *PAC*

Nume comanda	ID	Descriere comanda
COM_REGISTER	0x00	Setare configuraţia *PAC*
COM_UNREGSITER	0x01	Resetare configuraţia *PAC*
COM_START	0x02	Pornire achiziţie *PAC*
COM_STOP	0x03	Oprire achiziţie *PAC*
COM_EXIT	0x04	Închide aplicaţia de comanda şi control
COM_DATA_SET	0x05	Comunica date de comanda.
COM_DATA_GET	0x06	Comunica date de achiziţie.
COM_ACK_OK	0x07	Răspunsul ultimei operaţii.
COM_ACK_ERROR	0x08	Răspunsul ultimei operaţii.
COM_INVALID	0x09	Răspunsul ultimei operaţii.

- **Comanda *COM_REGISTER***

Este folosită de *PC* pentru a comunica către *PAC* canalele de pe placa de achiziţie care prezintă interes. *PAC* va comunica către procesul client doar datele configurate în aceasta structură.

Tabelul 19: Comanda *COM_REGISTER*

Crt.	Denumire	Dimensiune	Funcţionalitate
1.	uiItems	UINT32	Numărul de canale de configurat
2.	uiChanne	UINT32	Identificatorul canalului

	l		
3.	uiCardId	UINT32	Identificatorul plăcii de achiziţie
4.	uiIsInput	UINT32	Nul daca este canal de ieşire

Câmpurile 2,3 şi 4 se repetă în cadrul mesajului de numărul de ori exprimat în câmpul *uiItems*.

- **Comanda *COM_UNREGISTER***

Este folosita de către *PC* pentru a comunica către *PAC* canalele de pe placa de achiziţie care urmează a fi resetate. *PAC* întrerupe comunicarea către procesul client a datelor configurate în această structură.

Tabelul 20: Comanda *COM_UNREGISTER*

Crt.	Denumire	Dimensiune	Funcţionalitate
1.	uiItems	UINT32	Numărul de canale de configurat
2.	uiChannel	UINT32	Identificatorul canalului
3.	uiCardId	UINT32	Identificatorul plăcii de achiziţie
4.	uiIsInput	UINT32	Nul daca este canal de ieşire

Câmpurile 2,3 şi 4 se repetă în cadrul mesajului de numărul de ori exprimat în câmpul *uiItems*.

- **Comanda *COM_START***

Porneşte procesul de achiziţie şi comunicare datelor în cadrul *PAC*. Comanda nu are parametri.

- **Comanda *COM_STOP***

Opreşte procesul de achiziţie şi comunicare datelor în cadrul *PAC*. Comanda nu are parametri.

- **Comanda *COM_EXIT***

Întrerupe funcţionarea aplicaţiei. Comanda nu are parametri.

- **Comanda *COM_DATA_SET***

Setează valorile de comanda pe plăcile de achiziţie. Valorile sunt reprezentate în unităţi inginereşti.

Tabel 21: Comanda *COM_DATA_SET*

Crt.	Denumire	Dimensiune	Funcţionalitate
1.	uiItems	UINT32	Numărul de valori de setat
2.	uiChannel	UINT32	Identificatorul canalului
3.	uiCardId	UINT32	Identificatorul plăcii de achiziţie
4.	uiIsInput	UINT32	Nul daca este canal de ieşire
5.	fValue	DOUBLE	Valoarea în unităţi inginereşti

Câmpurile 2, 3, 4 şi 5 se repetă în cadrul mesajului de numărul de ori exprimat în câmpul *uiItems*.

- **Comanda *COM_DATA_GET***

Comunica valorile achiziţionate către *PC* în unităţi inginereşti.

Tabel 22: Comanda *COM_DATA_GET*

Crt.	Denumire	Dimensiune	Funcţionalitate
1.	uiItems	UINT32	Numărul de valori de setat
2.	uiChannel	UINT32	Identificatorul canalului
3.	uiCardId	UINT32	Identificatorul plăcii de achiziţie
4.	uiIsInput	UINT32	Nul daca este canal de ieşire
5.	fValue	DOUBLE	Valoarea în unitari inginereşti

Câmpurile 2, 3, 4 şi 5 se repetă în cadrul mesajului de numărul de ori exprimat în câmpul *uiItems*.

- **Comanda *COM_ACK_OK***

Indica faptul că ultima operaţiei s-a efectuat cu succes. Comanda nu are parametri.

- **Comanda *COM_ACK_ERROR***

Indica faptul că ultima operaţie s-a efectuat cu eşec. Comanda nu are parametri.

- **Comanda *COM_INVALID***

Indica faptul că ultima comanda efectuată nu a fost recunoscută. Comanda nu are parametri.

4.2.3 Comunicaţia intre nivele

Procesele client comunică între ele prin intermediul aceloraşi canale *FIFO* utilizând *API*-ul descris la [4.2.2] sau o extensie a acestuia. Ea este folosita pentru a realiza comunicarea cu regulatoarele:

- setarea valorilor de referinţa
- extragerea de valori de stare
- oprirea/pornirea/repornirea

4.3Arhitectura unui proces client

Procesul client (*PC*) este orice proces ce comunică cu *procesul de achiziţie* (*PAC*). Comunicaţia are loc prin intermediul bibliotecii *ipcomp*. Principiul de baza al aplicaţiei este descris în figura de mai jos.

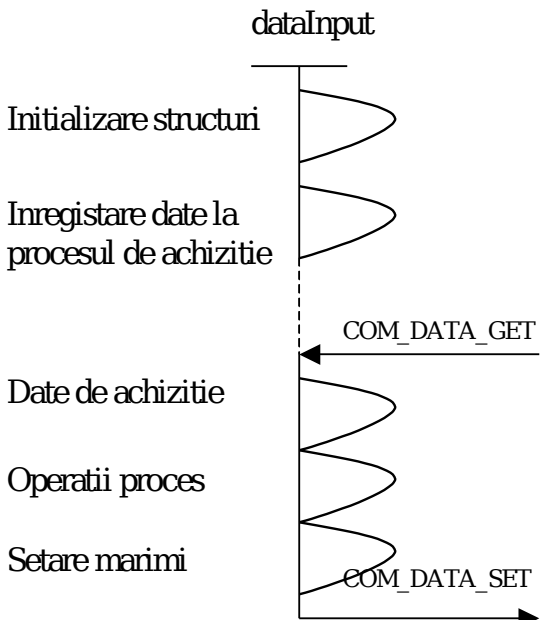


Figura 72: Funcţionalitate proces client

Aplicaţia înregistrează datele utilizate către procesul de achiziţie conform cu [4.2.2], după care aşteaptă comenzi pe canal de la celelalte procese participante la proces. Interfaţa de intrare este o extindere a interfeţei de la [4.2.2], comenzi particulare putând fi adăugate pentru a putea obţine variate funcţionalităţi.

4.4 **Structura aplicatiei**

Aplicaţia este bazata pe principiile descrise în paragrafele [4.2.1] şi [4.2.2]. Elementele de pe nivelul doi, prezentate în termeni generici în figura 73 vor fi instanţiate de următoarele procese:

- Regulatorul de temperatură
- Regulatorul de *pH*
- Regulatorul debitului de aer

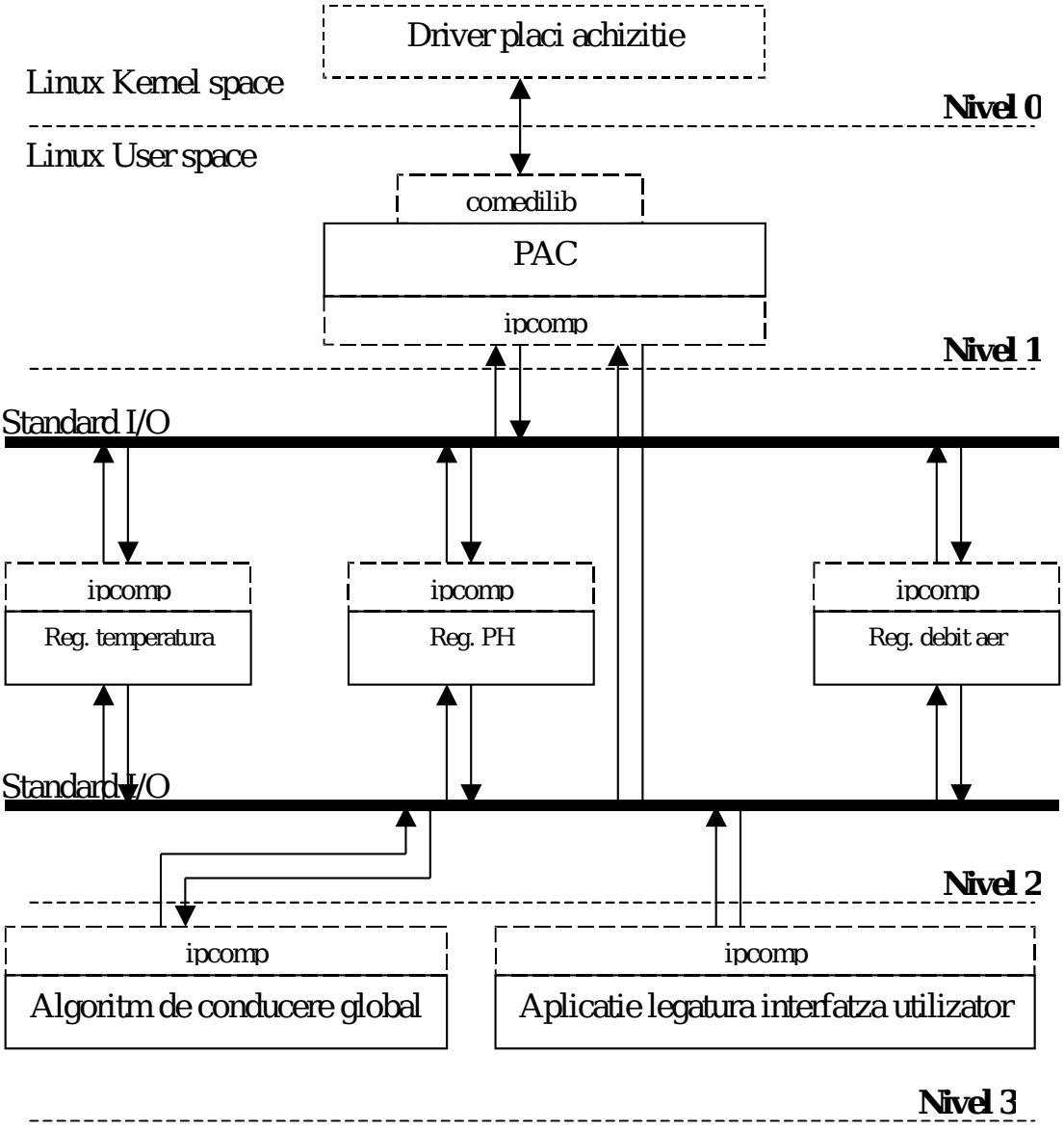


Figura 73: Arhitectura aplicaţie de conducere şi comanda

4.4.1 **Proces – achiziţie şi control date**

Procesul de achiziţie şi control date conţine doua taskuri:

- primul va realiza achiziţia şi setarea datelor pe plăcile de achiziţie (referit ca *task achiziţie*)
- al doilea, taskul de mesaje, este cel implementat de biblioteca *ipcomp* şi va asista comunicaţia cu procesele client

Cele doua procese comunică între ele următoarele informaţii prin intermediul unor structuri globale:

- Informaţii privitoare la datele ce trebuiesc achiziţionate respectiv setate pe canalele plăcilor de achiziţie. Aceste date sunt configurate prin intermediul comenzilor *COM_REGISTER* şi *COM_UNREGISTER*.
- Pornirea respectiv oprirea taskului de achiziţie prin comenzile *COM_START*, *COM_STOP*

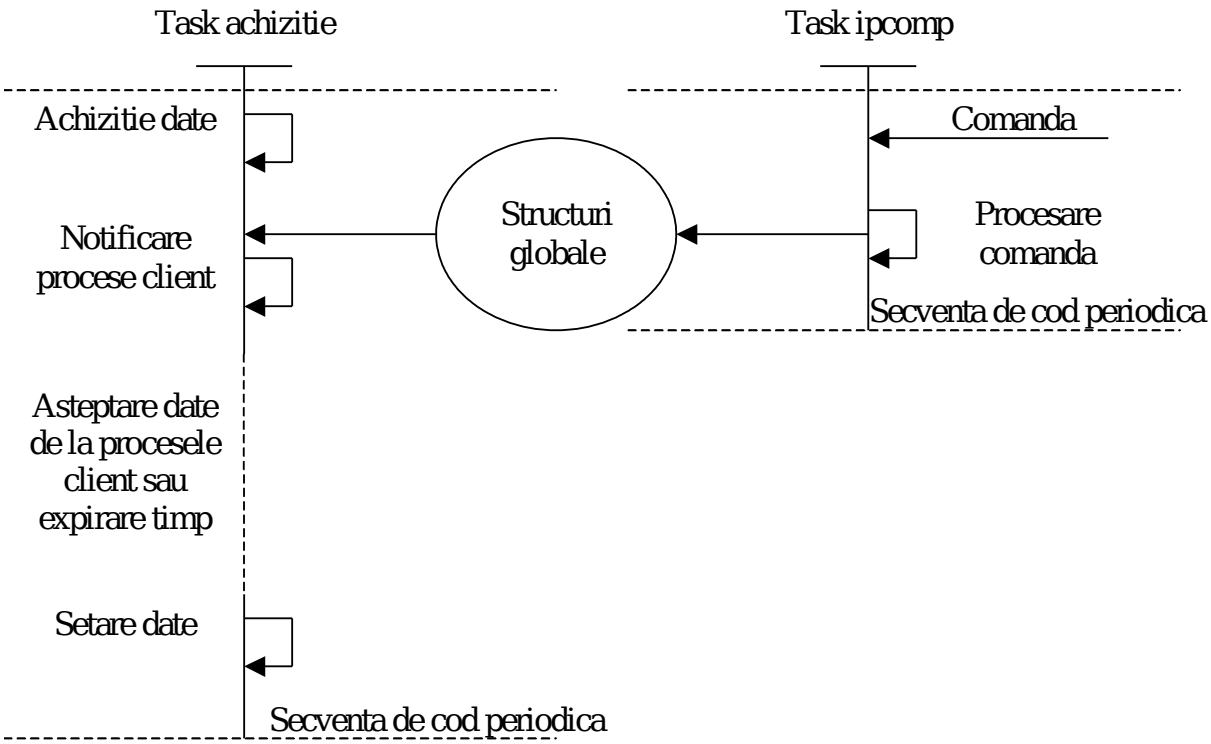


Figura 73: Comunicaţie taskuri *PAC*

Regulatorul de temperatură

Este descris in sectiunea 3. Caracteristicile acestuia sunt prezentate in tabelul 23.

Tabelul 23: Regulator temperatură

Regulator temperatură	
Mărimi intre	Temperatura masurata
Canal	*AI01 – PCI* 1710
Interval unităţi inginereşti	0-120grC
Interval	4-20mA
Mărimi ieşire	Comanda pornire / oprire rezistenta încălzire
Canal	*EN02 – PCI* 1710
Interval unităţi inginereşti	0/1
Interval	0/1
Mărimi referinţă	Temperatura
Interval unităţi inginereşti	0-120 gr C

Regulatorul de *pH*

Este descris in sectiunea 3. Caracteristicile acestuia sunt prezentate in tabelul 24.

Tabelul 24: Regulatorul de *pH*

Regulator PH	
Mărimi intrare	*pH*-ul masurat
Canal	*AI02 – PC*I 1710
Interval unităţi inginereşti	0-14
Interval	4-20mA
Mărimi ieşire	Comanda pompe dozatoare acid, respectiv baza
Canal	*AO1 – PCI* 1710, *Vout0 – PCI* 1720
Interval unităţi inginereşti	0-100ml/h
Interval	0-4V
Mărimi referinţă	*pH*-ul
Interval unităţi inginereşti	0-14

Regulatorul de debit aer in bazinul aerat

Este descris in sectiunea 3. Caracteristicile acestuia sunt prezentate in tabelul 25.

Tabelul 25: Regulatorul de debit de aer

Regulator debit aer bazin aerat	
Mărimi intrare	Debitul de aer masurat
Canal	*AI05 – PCI* 1710
Interval unităţi inginereşti	0-5l/h
Interval	4-20mA
Mărimi ieşire	Comanda catre electrovalva cu actiune continua *R2*
Canal	*Vout3 – PCI* 1720
Interval unităţi inginereşti	Procente
Interval	0-5V
Mărimi referinţă	Debitul de aer
Interval unităţi inginereşti	0-5l/h

4.4.2 Legătura cu interfaţa grafică

Aplicaţia de timp real comunică cu interfaţa grafică prin pachete *Ethernet UDP/IP*. Pachetele recepţionate de aplicaţie (*pachete HMI de comanda*) si cele către interfaţa grafica (*pachete HMI de achiziţie*) au fost descrise in sectiunea 3. Se respecta protocolul *UDP*, intocmai, interfata grafica putand lucra, fie cu versiunea de sistem de conducere din *Matlab-Simulink*, fie cu aceasta, prezentata in sectiunea de fata, dezvoltata sub sistemul de operare *Linux*.

Din punct de vedere software, aplicaţia conţine un task ce implementează aplicaţia server ce asculta mesajele *UDP/IP* venite de la interfaţa grafica şi un task ce implementează protocolul de comunicaţie *ipcomp*. Sincronizarea taskurilor se realizează după următorul algoritm:

- pachetul de date *HMI* recepţionat de server este convertit în structura de comanda *COM_DATA_SET* şi memorat intr-o variabila globala;

- funcţia ce implementează evenimentul rezultat în urma recepţionării comenzii *COM_DATA_GET* la nivelul *ipcomp*, memorează datele de achiziţie recepţionate şi le converteşte în pachet *HMI* de achiziţie, comunică datele recepţionate de la server proceselor de timp real şi efectuează comunicarea la nivel *Ethernet* a pachetului de achiziţie către interfaţa grafică.

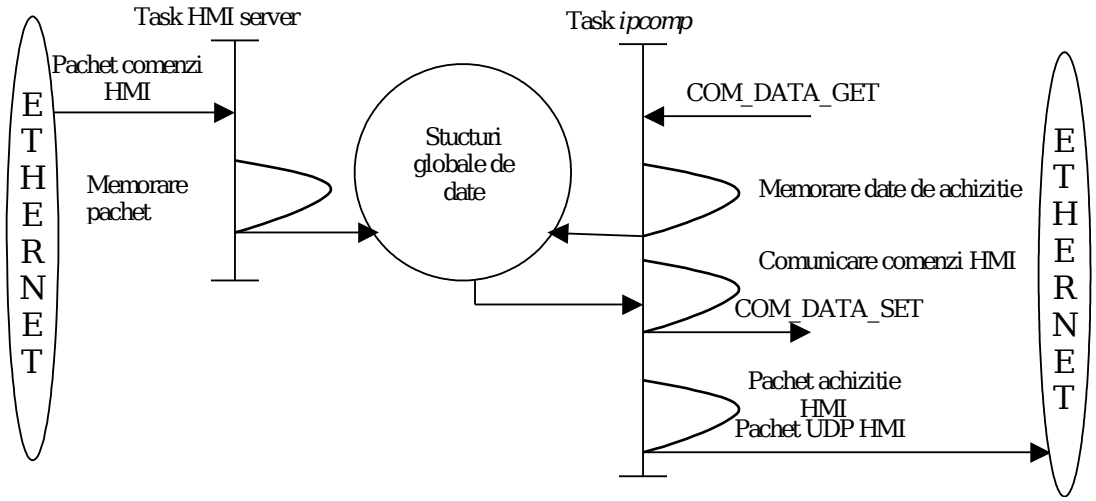


Figura 14: Comunicaţia între interfaţa grafică şi aplicaţia de timp real

Întrucât acest proces are prioritatea cea mai mica în cadrul sistemului, pentru fiecare operaţie de scriere în canal *FIFO*, planificatorul de taskuri va ceda accesul procesului care realizează citirea datelor.

4.5Algoritmul global de conducere

Acesta este realizat sub forma unui task care actualmente este vid, urmand ca in etapa III acesta sa contina algoritmul de conducere la nivelul superior (peste buclele simple) prin care se urmareste eficientizarea procesului de epurare, adica imbunatatirea indicatorilor de calitate a apei epurate. In principiu, legile de comanda avute in vedere vor genera viteza de dilutie si rata de aerare (debitul de apa de epurat asigurat de pompa *P1* si debitul de aer asigurat de electrovalva cu actiune continua *R2*), in functie de parametrii de intrare ai procesului.

In aceasta varianta s-a luat in considerare functionarea instalatiei intr-un regim automat general, care realizeaza secventierea operatiilor de initializare a unei sarje, precum si oprirea ei. Regimul mentionat este prezentat in diagrama de stari din figura75.

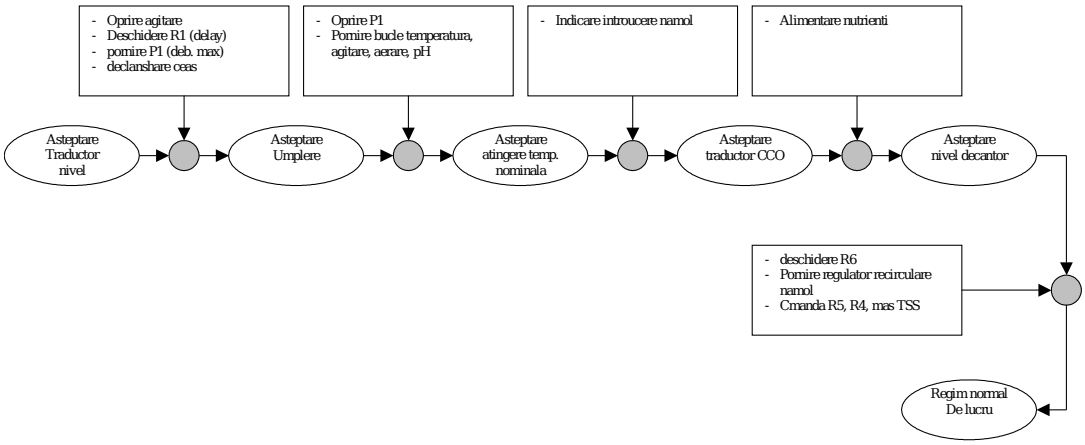


Figura 75: Starile sistemului la initializarea unei sarje