Iceland Liechtenstein Norway grants

"Working together for a green, competitive and inclusive Europe"

**Project:** *Digitalisation of water sector and water education* - **DIGIWATRO**,
**Contract:** 20-COP-0050

# **Intellectual Output 2:** *Promoting life-long learning in the water sector using hybrid education*



UNIVERSITAS GALATIENSIS

Norwegian University of Life Sciences

**Introducere**

În cadrul acestui Intellectual Output a fost proiectat și realizat un program de formare continuă intitulat „Digitalizare în industria apei”. Programul a fost realizat în cadrul Departamentului de Formare Continuă și Transfer Tehnologic din cadrul Universității „Dunărea de Jos” din Galați, fiind aprobat prin Hotărârea Senatului Universitar nr.331 din 20.10.2022 (print screen din hotărârea Senatului Universitar este dat mai jos).

**ROMÂNIA**
MINISTERUL EDUCAȚIEI
UNIVERSITATEA „DUNĂREA DE JOS” DIN GALAȚI

UNIVERSITAS GALATIENSIS

**Senatul universitar**
**Nr. 1009/20.10.2022/CS**

UNIVERSITATEA "DUNĂREA DE JOS" DIN GALAȚI
Nr. înregistrare 45385
Data înregistrării 20/10/2022

**HOTĂRÂREA SENATULUI UNIVERSITAR**
**nr. 331 din 20 octombrie 2022**

**În baza:**
Legii Educației Naționale nr. 1/2011, cu modificările și completările ulterioare;
Cartei Universității „Dunărea de Jos” din Galați;
*Regulamentului de organizare și de funcționare ale Senatului UDJG*;
Hotărârii Consiliului de administrație nr. 138/11.10.2022 înaintată Senatului, sub semnătura Rectorului universității,

**În urma rezultatului votului electronic cu termen limită 20 octombrie 2022, ora 16.00, la care au participat 75 dintre cei 82 de senatori, fiind întrunit astfel cvorumul și numărul de voturi necesar, Senatul universitar, cu unanimitatea voturilor exprimate,**

**H O T Ă R Ă Ș T E:**

**Art. 1.** Se aprobă, cu avizul Consiliului de administrație coroborat cu avizul pozitiv al Biroului Juridic, propunerea Departamentului de Formare Continuă și Transfer Tehnologic de înființare a cursului de formare continuă *Digitalizare în industria apei*.

**Art. 2.** Se aprobă, cu avizul Consiliului de administrație, coroborat cu avizul pozitiv al Biroului Juridic și al Comisiei didactice și de calitate a Senatului universitar, planul de învățământ pentru cursul de formare continuă *Digitalizare în industria apei*, din cadrul Departamentului de Formare Continuă și Transfer Tehnologic.

Programul conține 4 discipline cu un număr total de 45 ore și 5 credite, 15 ore fiind alocate cursurilor, iar 30 ore fiind alocate activităților aplicative de tip laborator. Disiplinele incluse includ un conținut actualizat, în pas cu trendul actual în digitalizare, într-un efort asumat de către proiect de a îmbunătăți cunoașterea experților în domeniul apei privind digitalizarea. Aceste dicipline sunt (print screen cu planul de învățământ aprobat este dat mai jos):
- Sisteme SCADA
- Senzori software și BIG DATA
- Conducerea automată a proceselor
- Securitate cibernetică.

**Universitatea „Dunărea de Jos" din Galaţi**
**Departamentul de Formare Continuă şi Transfer Tehnologic**
Curs de formare continuă: **Digitalizare în industria apei**
Forma de învăţământ: **cu frecvenţă**
Domeniul de licenţă pe care se fundamentează programul de studii: **Ingineria sistemelor**
Programul de studii universitare de licenţă pe care se fundamentează programul de studii: **Automatică şi Informatică Aplicată**
Competenţe profesionale –Evaluarea prin monitorizare, diagnoză, analiza de date experimentale, în concordanţă cu standarde specifice de performanta a activităţilor de proiectare, implementare, testare, validare, exploatare şi mentenanţă a echipamentelor şi reţelelor de calculatoare folosite pentru conducere automată şi aplicaţii de informatică;
Competenţe transversale - Identificarea rolurilor şi responsabilităţilor într-o echipa plurispecializată, luarea deciziilor şi atribuirea de sarcini, cu aplicarea de tehnici de relaţionare şi muncă eficientă în cadrul echipei; Identificarea oportunităţilor de formare continuă şi valorificarea eficientă a resurselor şi tehnicilor de învăţare pentru propria dezvoltare.

## Plan de învăţământ

valabil începând cu anul universitar: 2022-2023

| Nr. crt. | Denumire disciplină | Activităţi didactice | | Nr. de credite | Forma de evaluare |
|---|---|---|---|---|---|
| | | C | L | | |
| 1. | Sisteme SCADA | 3 | 6 | 1 | V |
| 2. | Senzori software şi BIG DATA | 6 | 12 | 2 | V |
| 3. | Conducerea automată a proceselor | 3 | 6 | 1 | V |
| 4. | Securitate cibernetică | 3 | 6 | 1 | V |
| **Total** | | 15 | 30 | **5** | **4V** |
| | | **45** | | | |

În continuare este prezentat conținutul dezvoltat pentru disciplinele din cadrul programului „Digitalizare în industria apei”.

# SCADA

**- note de curs –**

# Cuprins

## 1. Sisteme SCADA, modelul ISA112 (International Society of Automation). Arhitectura sistemelor SCADA.

Sistemele SCADA sunt sisteme de sisteme de control și achiziție de date care in decursul anilor au evoluat de la unele simple, independente pana la sisteme complexe si conectate care comunica prin intermediul rețelelor de comunicație industriale uneori chiar intercontinentale. Denumirea de SCADA vine ca o abreviere a termenilor din limba engleza: „Supervisory , Control and Data Aquisition". Aceste sisteme au in componenta lor atât elemente software cat si elemente hardware.

Dezvoltările sistemelor SCADA au evoluat de la sisteme personalizate implementare pe o anumita configurație hardware si software la sisteme implementate pe platforma hardware si software standard. Acesta evoluție a dus la o optimizare de costuri pe parte de operare, întreținere, precum si la un management eficient datorat multitudinii de informații disponibile in timp real in sistemele moderne. Însă satisfacerea acestor cerințe de volum de informații tot mai mare care se culeg din procesele controlate in scopul eficientizării operării si managementului scot la iveala vulnerabilități ale sistemelor. Sistemele de control industriale, care la începuturi erau sisteme închise si care utilizau hardware si software proprietar acum sunt vulnerabile la intruziunile prin rețele de comunicație externe , inclusiv prin internet precum și intruziuni interne prin personalul care deservește aceste sisteme. Aceste posibile atacuri profita de vulnerabilități ale platformelor standard care se utilizează in cadrul sistemelor SCADA cum ar fi sistemele de operare Windows și PC-uri care au fost adoptate p escara larga in componenta sistemelor.

Importanta sistemelor SCADA sunt date de faptul ca ele sunt componente vitale a infrastructurilor critice ale majorității națiunilor. Ele controlează instalațiile din industria petrolului si gazelor, transporturi, apa si apa uzata, industria energetica, fabrici chimice și multe altele. SCADA oferă date în timp real despre operațiunile de producție către managementul întreprinderii, implementează paradigme de control mai eficiente, îmbunătățește funcționarea instalațiilor și siguranța personalului și reduce costurile de operare. Aceste beneficii sunt posibile prin utilizarea de hardware și software standard în sistemele SCADA combinate cu protocoale de comunicare îmbunătățite și conectivitate sporită către rețele exterioare , inclusiv internetul. Cu toate acestea, aceste beneficii sunt dobândite la prețul vulnerabilității crescute la atacuri sau acțiuni eronate dintr-o varietate de surse externe și interne.

Un sistem SCADA este format din următoarele trei elemente:

- Unul sau mai multe dispozitive de interfață cu elementele de câmp care sunt de cele mai multe ori echipamente RTU (Remote Telemetry Unit) sau PLC-uri (Programable Logic Controler), care aduna datele din câmp – semnale de stare de la elementele de execuție si de la elementele de instrumentație, executa programele software implementate, transmit comenzi către elementele de execuție , transmit toate aceste informații achiziționate si procesate către sistemele superioare de vizualizare, monitorizare.
- Un sistem de comunicații format din una sau mai multe tipuri de rețele de comunicație industriale care este utilizat pentru a transfera date între dispozitivele de interfață de date de camp și unitățile de control și elementul central al SCADA. Sistemul poate fi radio, cablu, satelit etc. sau orice combinație a acestora.
- Un server sau servere sau un PLC care formează elementul central (uneori numite Centru SCADA, master stație sau unitate terminală principală .
- O colecție de software standard și/sau personalizat care este utilizat in centrul SCADA sau pe interfețele de operare din câmp sau pe calculatoarele de operare si care se numesc software HMI (Human Machine Interface ).

Componenta sistemelor SCADA este detaliata de asociații de specialiști in domeniu, care au drept scop standardizarea acestui tip de sisteme si una din aceste asociațiile care dezvolta standarde in domeniul SCADA este ISA112 – International Society of Automation. Comitetul asociației are acum peste 200 de experți SCADA din întreaga lume, reprezentând o gamă largă de industrii

Activitatea acestei asociații abordează standardizarea activităților de proiectarea, implementarea, operarea și întreținerea sistemelor a SCADA pentru diverse industrii esențial pentru a sprijini integritatea și fiabilitatea generală a acestor sisteme. Standardele și rapoartele tehnice ce se dezvolta oferă îndrumări cu privire la modul de proiectare, implementare, operare și întreținere a sistemelor SCADA prin documentarea celor mai bune practici într-o serie de industrii. Planul anticipat este de a dezvolta unul sau mai multe standarde care să fie completate cu rapoarte tehnice care extind detaliile de implementare și orientări specifice industriei.

In figura 1 este prezenta arhitectura modelului unui sistem SCADA in viziunea ISA112.

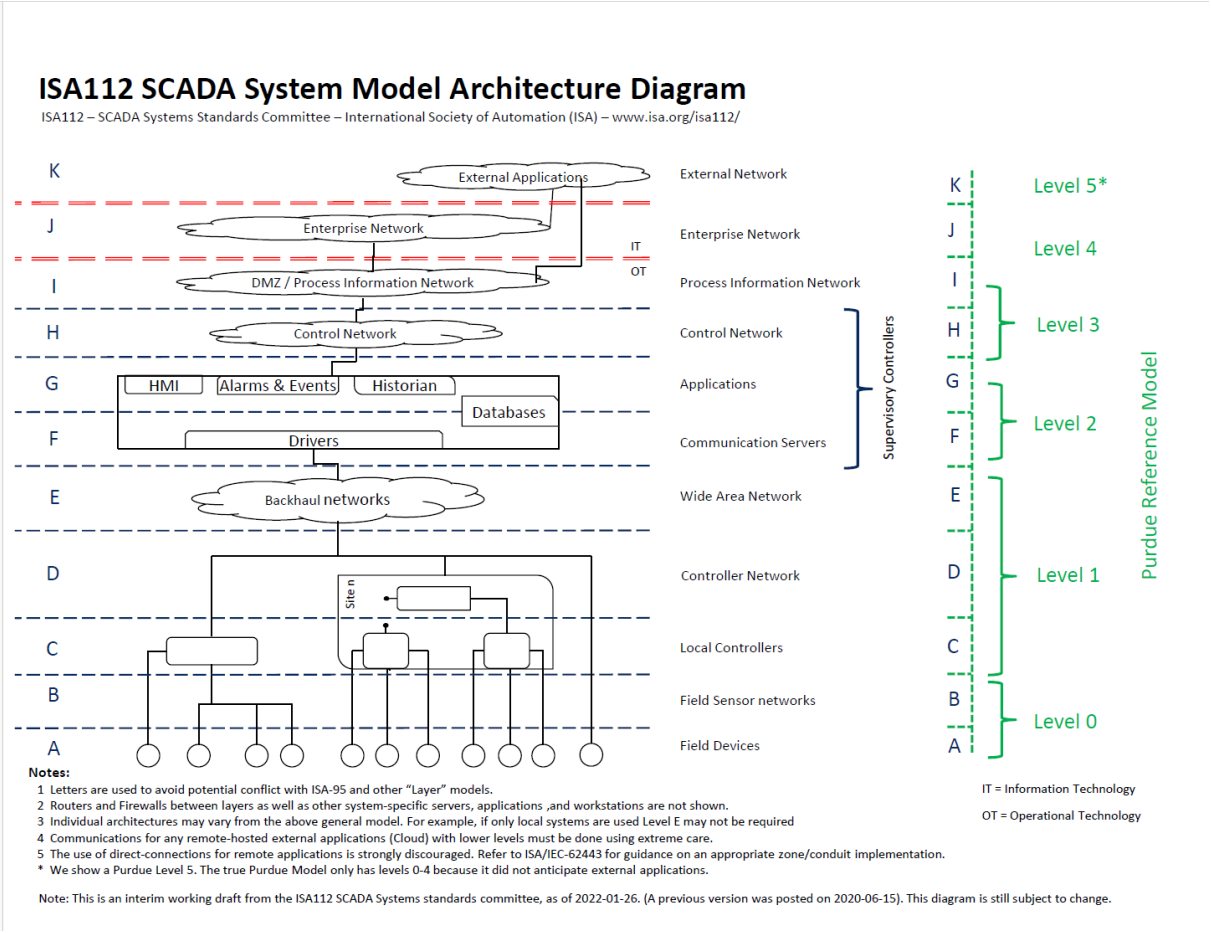


Figura 1 – Arhitectura modelului sistemului SCADA.

Modelul este unul de detaliu care exprima nivelele sistemului si acestea sunt corelate cu un alt model recunoscut „Purdue Reference Model". Cele doua modele sunt similare, modelul ISA detaliază intr-o mai mare măsura componentele unui sistem SCADA.

Primul nivel ISA1112 este nivelul A, nivelul elementelor de execuție si de instrumentație (senzori)– „Field Devices". Aici regăsim actuatoare, vane, motoare, pompe, coboți, brațe robotice, etc.

Acestea reprezintă toate acele elementele care primesc comenzi de la elementele de control si interacționează cu procesul condus.

Al doilea nivel notat cu B este nivelul rețelelor pentru senzori. Unii senzori transmit semnale unificate către sistemele de control însă exista si senzori inteligenți care transmit informațiile pe rețele de comunicație pe fir sau fără fir pe rețele IO link, DeviceNet, Profibus.

Cele doua nivele A si B formează Level 0 - „Phisical Process" in modelul Purdue .

Al treilea nivel notat cu C este nivelul echipamentelor de control local – „Local controllers". In aceasta zona regăsim PLC-urile , RTU-urile care executa programele ce controlează local procesul.

Al patrulea nivel notat cu D este nivelul rețelelor de comunicație intre PLC-uri – „Controller Networks". In acesta zona pot exista rețele de tip ControlNet, EthernetIP, Profinet.

Al cincilea nivel notat cu E este nivelul rețelelor de comunicație de arie larga care pot conecta sistem e de control din locații diferite. - „Wide Area Networks". In acesta zona pot exista rețele pe protocol de Modbus, etc.

Cele trei nivele C, D si E formează Level 1 - „Basic Control" in modelul Purdue .

Al șaselea nivel notat cu F este nivelul serverelor comunicație – sunt mașinile pe care rulează serverele cu aplicațiile de HMI, baze de date, alarme, historian, drivere de comunicație cu echipamentele din nivele de control etc. – „ Communication Servers". Anumite servere pot fi in configurație redundanta pentru siguranța in funcționare a procesului condus.

Al șaptelea nivel notat cu G este nivelul aplicațiilor – aplicațiile SCADA dezvoltate conțin ecrane de monitorizare si control a instalației, alarme, rețele, ecrane pentru parametri si toate configurarile de funcționalitate si securitate, etc. – „ Applications".

Cele trei nivele F si G formează Level 2 - „Supervisory Control" in modelul Purdue .

Al optulea nivel notat cu H este nivelul rețelei de control – aplicațiile de control pentru sistemul condus, adică planificarea producției, managementul rețelelor, controlul calității, MES - „ Control Network ".

Al noualea nivel notat cu I este nivelul zonei DMZ si control de proces –- „ Process Information Network ".

Cele doua nivele H si I formează Level 3 - „Operation systems (Manufactoring)" in modelul Purdue .

Nivelele de la A la I sunt nivele care sunt in zona de OT – Operational technology

Al zecelea nivel notat cu J este nivelul zonei de rețea de întreprindere –- „ Enterprise Network ". ERP application.

Nivelul J corespunde nivelului Level 3.5 - „DMZ in modelul Purdue .

Al zecelea nivel notat cu K este nivelul aplicațiilor externe –- „ External application ".

Nivelul k corespunde nivelului Level 4 - „Enterprise (IT)” in modelul Purdue .

Nivelele J si K sunt nivele care sunt in zona de IT – Information technology

## 2. Rețele industriale de comunicație utilizate in aplicațiile SCADA.

Pentru ca două sau mai multe sisteme să comunice, ele trebuie să vorbească un același fel de limbaj care poarta numele de protocol de comunicație. Fiecare protocol este format din anumite reguli de comunicație, cum se inițiază aceasta, cu se comunica, cum se finalizează comunicația.

Protocoalele SCADA au evoluat din necesitatea de a trimite și primi date și informații de control atât la nivel local pe o distanta scurta cat și pe distanțe lungi iar timpul sa fie determinist. Determinist în acest context se referă la capacitatea de a prezice timpul necesar pentru a tranzacția să aibă loc atunci când toți parametri de comunicație sunt cunoscuți. Pentru a realiza comunicarea în timp determinist pentru aplicații industriile esențiale cum ar fi zona de rafinare, utilități electrice, și alți utilizatori de sisteme SCADA, producători de dispozitive de control, cum ar fi PLC-urile, și-au dezvoltat propriile protocoale și structuri de comunicații. Producătorii importanți de echipamente de automatizare si control au dezvoltat protocoale si exemple in acest sens putem da :

- Allen Bradley (Rockwell Automation) – o mare companie americana care a dezvoltat protocoalele: DeviceNet, ControlNet, Data Highway+, Data Highway 485, Ethernet IP, si altele.
- Siemens – o mare companie europeana care a dezvoltat Profibus, Profinet, S7 protocol, etc.
- Modicon – preluata de Scheider – o firma europeana acum dar infiintata in america a dezvoltat protocoalele Modbus RTU. Modbus TCP.

Multe dintre aceste protocoale la început au fost proprietare, apoi datorita preocupării specialiștilor pentru standardizare începuta după anii 90 s-a trecut dezvoltarea de protocoale deschise pentru comunicația sistemelor de control, protocoale standard non – proprietare.

Apoi, pe măsură ce Internetul a câștigat popularitate, companiile au căutat să ia avantajul protocoalelor și instrumentelor dezvoltate pentru Internet, cum ar fi familia de protocoale care au la baza modelul ISO OSI - TCP/IP , prezentata in figura 2. În plus, producătorii iar organizațiile cu standarde deschise au modificat protocoalele cele foarte populare și eficiente astfel încât sa poată fi accesibile fiecărui producător de echipamente capabile de comunicație.

Un protocol definește formatul mesajelor și reguli pentru schimbul de mesaje. Modelele de nivel înalt sunt utilizate pentru a defini unde sunt aplicate protocoalele și către compartimentează funcțiile necesare trimiterii și primirii mesajelor . Modelul de arhitectură stratificat a fost adoptat pe scară largă și este foarte eficient. În acest model, elementele necesare comunicării sunt împărțite in nivele cu interfețe definite între fiecare nivel. Două dintre cele mai utilizate pe scară largă modelele de referință de comunicare sunt interconectarea sistemelor deschise (OSI) și protocolul de control al transmisiei/protocolul Internet (TCP/IP) model.

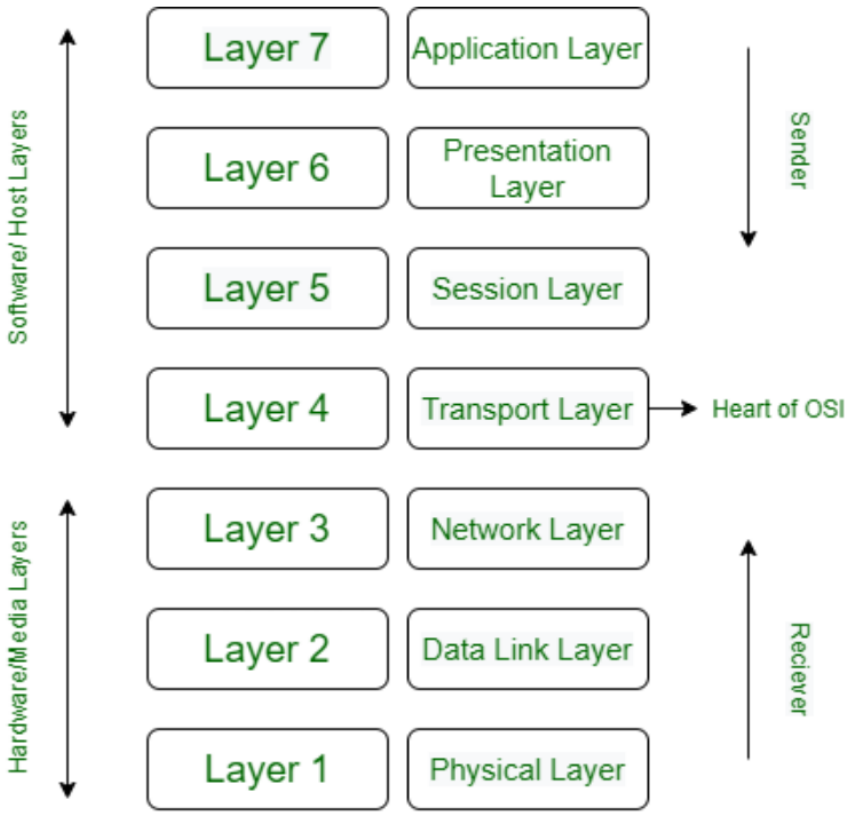


Figura 2 – Modelul ISO - OSI .

**Protocoale SCADA**

Protocoalele sistemului SCADA au evoluat din hardware și software de proprietate concepute special pentru sisteme SCADA. Protocoalele au fost dezvoltate de necesitate pentru a servi piața în plină dezvoltare a aplicațiilor de calculator în timp real situatii de control. Apoi, într-un efort de a profita de noile rețele dezvoltări, protocoalele SCADA au încorporat versiuni de Internet și locale tehnologii de rețele de zonă. Această mișcare a dus la o oarecare standardizare, dar și au expus sistemele SCADA la atacuri utilizate în mod obișnuit împotriva acestor tehnologii în medii IT.

**Protocoalele MODBUS**

La sfârșitul anilor 1970, Modicon, Incorporated, a dezvoltat protocolul MODBUS. MODBUS este poziționat în nivelul 7 al modelului OSI și acceptă comunicații client-server între PLC-uri Modicon și alte dispozitive aflate in retea.Protocolul MODBUS definește metodele pentru accesul la un PLC, pentru comunicatia intre echipa,mante care au implementat acest protocol si ofera mijloace pentru detectarea și raportarea erorilor. Pentru a profita de capabilității echipamentelor de instrumentație care suporta comunicație pe Ethernet a fost realizata si varianta Modbus TCP. Aceasta se bazează pe modelul OSI, deși nu sunt folosite toate nivelele modelului.

**Protocolul DNP3**

DNP3 este un protocol SCADA deschis care este utilizat pentru comunicații seriale sau utilizand IP între dispozitivele de control. Este utilizat pe scară largă in aplicatiile din cadrul companiilor de apă și furnizorii de energie electrică pentru schimbul de date și instrucțiuni de control între stațiile principale de control și calculatoarele sau controlerele de la distanță numite stații externe.

Comenzile tipice emise de stația de control principală sunt „deschidere a supapei”, „porniți un motor” și „furnizați date despre o anumită stație de control”. Stația de control principală poate furniza, de asemenea, semnale analogice de ieșire către o stația externă. O stație externă oferă stației de control principal informații precum presiuni, starea unui întrerupător semnale analogice reprezentând temperatura sau puterea și fișierele de informații. DNP3 s-a adaptat, de asemenea, la tehnologiile Internet folosind TCP/IP pentru schimb de mesaje DNP3.

**Protocol industrial CAN**

Protocolul CAN a fost dezvoltat de către Robert Bosch, GMBH in special pentruindustroia auto. Suporta comunicații seriale de până la 1 Mbps, suportă fizic până la 110 noduri pe o rețea cu două fire, semi-duplex. Protocoalele operează la nivelul 1, nivelul fizic, și la nivelul 2, legătura de date stratul, al modelului OSI.

Comunicațiile CAN se bazează pe „Carrier Sense Multiple Access/ Collision Detection „ CSMA/CD ca metoda de detectare a colizoiunilor in retea. Astfel in retea exista mai multe dispozitive care trasmit date pe o magistrala comuna. Când un dispozitivul simte că magistrala este liberă (fără semnal purtător pe magistrală), încearcă să transmit peste madistrala. În cazul în care un alt dispozitiv încearcă să comunice prin magistrala in acelasui timp, dispozitivele detectează această coliziune, se retrag și încearcă din nou la a timp aleatoriu mai târziu. Astfel, cu această abordare, timpii de transmisie specifici de-a lungul rețeaua nu poate fi garantată. Pentru a compensa această situație, CAN oferă priorități de transmisie nodurilor folosind o schema de CSMA/CD + AMP (arbitraj pe prioritatea mesajului). CSMA/CD + AMP folosește un identificator unic care include o evaluare de prioritate într-un mesaj în locul sursei și adresele nodului destinație așa cum sunt utilizate în arbitrajul convențional CSMA/CD metodă. Cu cât valoarea identificatorului este mai mică, cu atât prioritatea este mai mare atribuite mesajului. Lungimea acestui identificator variază, fiind de 11 biți în Specificația CAN partea A și lungimea de 29 de biți pentru specificația CAN partea B.

**Protocol CIP**

Protocolul „Common Industrial Protocol” este o familie deschisă de protocoale, implementat în straturile de aplicație, prezentare și sesiune ale modelului OSI. Astfel, CIP formează un strat superior comun de protocoale care poate fi utilizat deasupra diferitelor straturi inferioare, cum ar fi cele care folosesc EtherNet/IP, DeviceNet, și ControlNet, toate acestea fiind amintite in cele ce urmeaza. De asemenea include un protocol de mesagerie care acceptă mesagerie explicită și I/O. Protocolul CIP este întreținut de ControlNet International (CI) și de furnizorul Open DeviceNet Asociația (ODVA).

Protocolul CIP cuprinde obiecte de comunicarea , care sunt utilizate pentru a defini valorile maxime ale datelor, tipul și caracteristicile conexiunii și momentul conexiunii. De asemenea, furnizează o bibliotecă de obiecte cu 46 de clase, care include obiecte de supervizor de control, obiecte de port, obiecte identitate, obiecte punct de ieșire analogică, obiecte parametri, obiecte de intrare discrete, obiecte senzor de poziție și obiecte de acționare AC/DC.

**DeviceNet**

DeviceNet este un standard deschis care este utilizat pentru a conecta echipamente precum softstartere, senzori, comenzi ale supapelor, afișaje, interfețe pentru operator și calculatoare de control de nivel superior și PLC-uri. DeviceNet se bazează pe protocoalele CAN. De asemenea, utilizează familia de protocoale CIP, inclusiv bibliotecile sale de obiecte și profile de obiect pentru configurarea și controlul echipamentului și pentru a obține date de la dispozitivele locale prin protocoalele CAN la legătura de date și straturi fizice. Pentru a realiza un schimb de informații, de exemplu, DeviceNet

stabilește o instanță de conexiune folosind un obiect de identitate, un mesaj obiect router, un obiect DeviceNet și un obiect conexiune. Identitatea obiectul conține informații precum profilul dispozitivului, numărul de revizuire și informații despre furnizor. Obiectul router de mesaje direcționează mesajele către corect destinație și obiectul DeviceNet stochează informații DeviceNet de nivel inferior cum ar fi numărul de identificare MAC. Obiectul de conexiune gestionează conexiunea de mesagerie. DeviceNet acceptă rate de comunicare de 125 kbps, 250 kbps și 500 kbps si poate gestiona un număr maxim de 64 de noduri.

**ControlNet**

ControlNet este o rețea deschisă pentru utilizare în timp real, si este o retea determinista care se utilizează in cadrul aplicațiilor SCADA. De asemenea, utilizează capabilitățile obiectului de protocol CIP și poate suporta până la 99 de noduri în rețea la o rată de date de 5 Mbps. Este conceput pentru aplicații care cuprind mai multe controlere și interfețe de operator și acceptă schimbul de date I/O în timp real, precum și de informații de mesagerie. Determinismul ControlNet provine din încorporarea algoritmul concurent CTDMA (Time Domain Multiple Access) care permite unui nod în rețea sa transmită date la un interval specificat numit timp de actualizare a rețelei sau NUT (Network Update Time). Astfel, informațiile critice sunt transmise în timpul unui interval NUT in timp ce informațiile necritice sunt trimise în perioade neprogramate atunci când sunt disponibile.

**EtherNet/IP**

Protocolul EtherNet/IP aplică, de asemenea, CIP prin codificarea mesajelor CIP în cadre Ethernet. Pe lângă clasele de obiecte CIP de bază, EtherNet/IP utilizează un obiect TCP/IP pentru implementare protocolului TCP/IP și un obiect de legătură Ethernet cuprinzând parametrii pentru stabilirea unei legături EtherNet/IP. Protocolul CIP operează la nivelul aplicație al OSI care furnizează biblioteca de obiecte de aplicație, la nivelul de prezentare furnizarea de servicii de mesagerie și la nivelul de sesiune mesaj de sprijin managementul rutare și conexiuni. Deoarece Ethernet folosește CSMA/CD, care funcționează prin detectarea coliziunilor, încercând să retrimită la intervale aleatorii comunicațiile care nu se pot realiza, nu le putem numi comunicații deterministe. Această situație pune probleme pentru achiziția de date în timp real si pentru controlul in timp real. Este implementata o metoda de reducere a coliziunilor si acestea influențează din ce in ce mai putin funcționarea unei comunicații prin acest protocol mai ales după ce viteza de comutație a crescut foarte mult ajungând la viteze de Gigabit Ethernet (10 Gbps). Este redusa astfel latența de comunicare .Un al treilea factor de atenuare este disponibilitatea de a utiliza date prin protocol UDP care este mult mai rapid decât TCP la care conexiune este confirmata și comunicația este încărcata si cu date pentru verificări ale erorilor de transmisie.În cele din urmă, IEEE a dezvoltat specificația 802.1P pentru prioritizarea traficului rețelei prin încorporarea unui câmp de antet pe 3 biți care prioritizează mesajele și permite gruparea pachetelor în diferite clase de trafic prioritar. Astfel aplicațiile de timp real cum ar fi aplicațiile cu „Motion" sau chiar aplicațiile de „safety" utilizează Ethernet IP ca rețea de comunicație intre echipamente.

**Profibus**

Profibus (Process Fieldbus) este un standard de rețea utilizând un principiu de comunicație serială de deschis pentru utilizare în aplicații de control și achiziție de date critice în timp. Se încadrează in Standardul internațional european de fieldbus, EN 50 170, și definește funcțional, caracteristicile electrice și mecanice ale unui FieldBus Serial. Profibus este similar cu FieldBus Foundation, dar oferă rate de transmisie de 31,25 Kbps, 1 Mbps și 2,5 Mbps în nivelul fizic. Deoarece Profibus este un standard deschis, poate găzdui dispozitive din diferite producători. Profibus se află pe nivelele aplicație, legătura

de date și fizic ale modelului OSI. Oferă determinism pentru aplicațiile de control în timp real și acceptă rețele de comunicații multimaster și master-slave.

Există trei versiuni de **Profibus** si acestea sunt:

- Profibus Process Automation (Profibus PA): Conectează dispozitivele de achiziția și controlul pe o magistrală serial comună .De asemenea, este posibil ca elementele din camp sa fie alimentate de retea prin intermediul magistralei. Profibus PA utilizează funcțiile de bază și extensiile disponibile în Profibus DP.
- Profibus DP : Oferă comunicare de mare viteză între sistemele de control și dispozitivele de control distribuite. Profibus DP a fost imbunatatit prin adăugarea de capabilități de diagnosticare , mesaje de alarma si parametrizare si a devenit ProfibusDPV1.
- Profibus Fieldbus (FMS): Dezvoltată pentru a fi susținută un număr mare de aplicații și interconexiuni de rețea de nivel superior dintre aplicații la rate medii de transmisie. Oferă o selecție mare de funcții și este, în general, mai complicat de implementat decât Profibus PA sau Profibus DP.

## 3. Controlul instalațiilor utilizând echipamente RTU (Remote Terminal(Telemetry) Unit) si PLC (Programable Logical Controller) .

Sistemele RTU in cadrul sistemelor SCADA au rolul de a achiziționa date si a le transfera către nivelul 2, „Supervisory Control". Sisteme RTU sunt in general PLC-uri care se pot si programa sau doar configura, pot executa anumite taskuri pe lângă cel de comunicație. O cerința importanta pentru sistemele RTU in cadrul sistemelor SCADA este aceea de a retine pe perioada mare de timp de ordinul orelor sau zilelor datele achiziționate daca comunicația cu nivelul 2 nu este funcționala. Un PLC (Programable Logical Controler) este format dintr-un controler care utilizează o memorie programabilă pentru stocare instrucțiuni și implementeza funcții precum logica, secvențierea, sincronizarea, numărarea și aritmetica în vederea controlului mașinilor și proceselor și sunt proiectate pentru a fi operate de ingineri cu cunoștințe limitate de calculatoare și limbaje de calcul. Termenul de logica este folosit deoarece programare se ocupă în primul rând de implementarea operațiilor logice și de comutare, de exemplu. dacă apare A sau B comutați pe C, dacă apar A și B comutați D. Intrare dispozitive, de ex. senzori, cum ar fi întrerupătoarele și dispozitivele de ieșire din sistem fiind controlat, de ex. motoarele, supapele etc., sunt conectate la PLC. Operatorul introduce apoi o secvență de instrucțiuni, adică un program, în memoria PLC-ului. Controlerul monitorizează apoi intrările și ieșirile conform acestui program și efectuează regulile de control pentru care a fost programat.PLC-urile sunt similare cu computerele, dar computerele sunt optimizate pentru sarcini de calcul și afișare, PLC-urile sunt optimizate pentru sarcini de control și mediu industrial. Astfel, PLC-urile sunt:

- Robuste și concepute pentru a rezista la vibrații, temperatură, umiditate si zgomot.
- Au interfața pentru intrări și ieșiri deja în interiorul controlerului.
- Sunt ușor de programat și au o programare ușor de înțeles limbaj care se preocupă în primul rând de logică și comutare operațiuni.

Primul PLC a fost dezvoltat în 1969. Ele sunt acum utilizate pe scară largă și controleaza aplicatii simple cu 20 digitale intrări/ieșiri pana la aplicatii foarte complede cu mii sau zeci de mii de de intrări/ieșiri, digitale sau analogice. De obicei, un sistem PLC are componentele funcționale de bază ale initatii de procesare, memorie, unitate de alimentare, secțiune de interfață de intrare/ieșire, interfața de comunicații.

## 4. Software SCADA. Funcțiile aplicațiilor SCADA. Monitorizare, alarmare, achiziție si salvare de date, raportare.

Aplicațiile de baza utilizate dezvoltare de aplicații SCADA sunt realizate in principal de producătorii de echipamente de control. Astfel firmele mari care au dezvoltat protocoale de comunicație de-a lungul timpului, au dezvoltat PLC-uri dezvolta si aplicați software SCADA. Putem aminti:

- Rockwell Automation a dezvoltat suita de programe FactoryTalkView si o promovează pentru realizarea aplicațiile de vizualizare si SCADA.
- Siemens a dezvoltat WinCC ca software de dezvoltare a aplicațiilor SCADA
- Schneider Electric a achiziționat Citect Scada pentru dezvoltarea aplicațiilor SCADA.

Aceste aplicații sunt dezvoltate pentru a comunica in special cu echipamentele de control proprii însă pot integra din ce in ce mai mult si echipamente de la alți producători. Unele sisteme sunt mai închise si este mai greu sa extinzi aceste sisteme , însă se observa o direcție de dezvoltare la toate companiile de a realiza sisteme din ce in ce mai deschise care permit integrarea tuturor echipamentelor care comunica pe protocoale standard de comunicație. Exista pe piața din ce in ce mai multe echipamente de conversie de protocoale si astăzi este destul de facil sa poți integra in orice sistem SCADA , orice echipament standard. Toate aceste aplicații software au nevoie de licențiere pentru a putea fi folosite si funcție de producător licențierea se face pe numar de ecrane sau pe număr de taguri, pe servere , pe clienți pe taguri arhivate in Historian, pe redundanta, etc.

O alta categorie de sisteme SCADA este reprezentata de aplicații care nu țin de un anume producător de echipamente ci sunt construite sa integreze un număr mare de sisteme de control de la diverși producători. Aceste sisteme se dezvolta din ce in ce mai mult in direcția WEB si avem așa numitele sisteme SCADA bazate pe Web. Un astfel de software de dezvoltare de aplicații SCADA bazate pe WEB este cel dezvoltate de Inductive Automation si se numește Ignition.

Acesta aplicație este multiplatformă și rulează pe Windows, Linux și Mac OS X. Exista astfel opțiuni de implementare pe hardware fizic, medii virtuale și servicii manageriate; există chiar și o imagine oficială Docker Hub. Ignition se poate implementa pe :

- Dispozitive: PC-uri Embeded, laptop-uri, desktop-uri, servere, Fog Computers
- Servicii manageriate: AWS EC2 , AWS ECS , AWS Outposts, Azure Virtual Machines, Azure Containers, Google Compute Engine
- Mașini virtuale și containere: VMware, Parallels, VirtualBox, Hyper-V, Docker

Licențierea in Ignition se refera la modulele pe care intenționam sa le utilizam in cadrul proiectului. Exista o configurație de baza care apoi se poate extinde module către pot asigura cu toate funcționalitățile necesare. Pentru o aplicație licențiata, numărul de taguri, ecrane si clienți este nelimitat. De asemenea este nelimitat si numărul de taguri din Historian. Caracteristicile importante ale acestui sistem SCADA dar si a tuturor celorlalte sisteme sunt următoarele:

1. Achiziția de date.

Software-ul SCADA Ignition include un set cuprinzător de instrumente de achiziție a datelor au modulul OPC UA incorporat în modulul de baza cu scopul de a se conecta la aproape orice tip de PLC și facilitatea de conectare la orice bază de date tip SQL. Ignition poate fi configurat sa salveze date in orice baza de date tip SQL s, rezultând un istoric performant de date pentru un proces industrial și se poate conecta la dispozitivele IIoT prin protocolul MQTT.

2. Dezvoltarea rapidă a oricărui tip de proiect SCADA

Realizarea rapida a aplicațiilor sub Ignition este facilitată de un mediu puternic de dezvoltare integrat (IDE), care oferă toate instrumentele necesare. Mediul de dezvoltare este integrat în platformă, astfel încât este disponibil instantaneu, funcționează pe orice sistem de operare important și vine cu un număr nelimitat de clienți concurenți.

3. Monitorizare în timp real

Ignition este proiectat pentru a simplifica transferul de date, astfel încât valoarea tagurilor să fie vizibile în timp real. Sistemul de monitorizare în timp real Ignition oferă posibilitatea de a vizualiza rapid starea instalației pe orice dispozitiv.

4. Controlul procesului cu ajutorul dispozitivelor HMI

Utilizand Ignition se pot porni și opri procesele, se pot monitoriza mai multe instalații în mai multe locații și se poate verifica starea întregii fabrici la un moment dat. Ignition conține modulul de design prin intermediul căruia se pot crea cu ușurință ecrane HMI optimizate pentru a realiza toate cerințele de monitorizare și control.

5. Instrumente de vizualizare .

Se pot dezvolta tablouri de bord dinamice cu instrumente puternice pentru analiza datelor. Sistemul utilizează o biblioteca completa de hărți și tabele personalizabile, pentru a monitoriza indicatorii de performanță cheie, pentru a vedea tendințele la un moment dat și multe altele.

6. Implementarea ușoară pe Web cu o scalabilitate deosebită

Cu Ignition se pot lansa instantaneu un număr nelimitat de clienți fără întreruperi, pe întreaga durata de funcționare, pe aproape orice dispozitiv cu conexiune la un server central. Cu arhitecturi pentru aproape orice tip de sistem și un model nelimitat de licențiere, Ignition se poate potrivi oricăror dimensiuni de implementare și poate crește ușor odată cu nevoile companiei.

7. Alte caracteristici:
   - Alarmare SCADA: Informații în timp real despre starea instalațiilor în orice locație.
   - Rapoarte dinamice: Se pot crea o gamă completă de rapoarte dinamice, bogate în date și se pot transmite în orice locație.
   - Gestionarea tranzacțiilor: Stocarea facilă a datelor salvate, proceduri de memorare stocate, sincronizare bidirecțională a datelor.
   - Istoric de date industrial: Dezvoltat pe baza de date tip SQL, istoricul de date salvate în timp reprezintă un modul foarte performant.
   - Acces mobil la aplicații: Este facilitat prin posibilitatea de a controla sistemele utilizând telefoanele tip smart și tabletele.
   - Simboluri grafice: Sistemul conține o biblioteca de mii de elemente grafice personalizate utilizabile în proiectele dezvoltate.
   - Securitate SSL: Ignition folosește SSL pentru a asigura protecția datelor.
   - Design concurențial al aplicațiilor: Posibilitatea de a dezvolta aplicațiile într-un mediu concurent și nelimitat pentru clienții de dezvoltare.
   - Scalabilitate: Scalare facilă de la un client către tot nivelul întreprinderii.
   - Sisteme critice: Se adaugă toleranța la erori pentru sistemele critice prin adăugarea de servere redundante.

## 5. Servere si clienți in arhitecturi SCADA.

In general in aplicațiile SCADA arhitecturile sunt de tip client - server cu posibilități de redundanta la nivel de server. Multiplele echipamente de tip server pot gestiona aplicații de baza de date, de alarme, HMI si altele.Sistemul SCADA Ignition, dezvoltat de Inductive Automation este in sistem SCADA dezvoltat pe o platforma WEB si include un set de instrumente performante pentru controlul și achiziția de date (SCADA) - toate într-o singură platformă, universală, deschisă și scalabilă. Ignition reprezintă o nouă generație de sisteme SCADA care rezolva probleme importante și dificile ale sistemelor SCADA. Ignition permite controlul facil al proceselor, monitorizarea, afișarea și analiza tuturor datelor fără limitări.

Arhitecturile de sistem SCADA in Ignition sunt diverse, plecând de la cele simple pana la cele complexe.

1. Arhitecturi simple. Acestea pot fi cu redundanta sau fără, In figura următoare este prezentat acest tip de arhitectura.


Figura 3 – Arhitectura simpla SCADA cu redundanta.

Arhitectura de bază cu redundanță este cea mai potrivită pentru aplicațiile care necesită un sistem SCADA scalabil, gestionat centralizat, folosind un singur server Ignition on-premise (cu un server redundant) și anumite module. Această implementare este cea mai rentabilă configurație. Sunt disponibile conexiuni, taguri, baze de date și clienți nelimitați pe web.Însă arhitectura poate fi si fără redundanta, serverul Ignition – Gateway -ul rulând doar pe o singura mașina.

2. Arhitecturi Scale – Out cu redundanta.

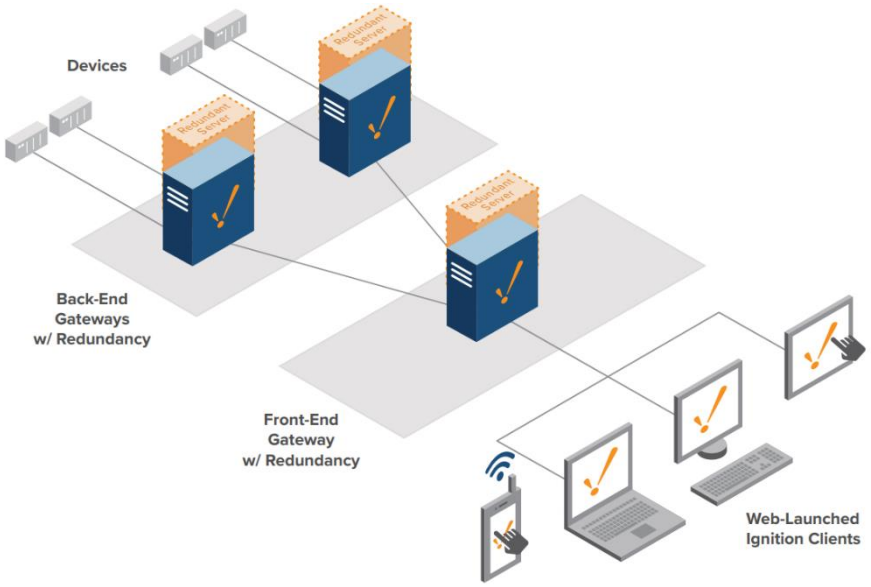


Figura 4 – Arhitectura Scale – Out SCADA cu redundanta.

Cu arhitectura Scale-Out, task-urile sunt împărțite între gateway-uri back-end (cu servere redundante) care gestionează datele dispozitivului și gateway-uri front-end care gestionează aplicațiile client. Această arhitectură se extinde cu ușurință fără a supraîncărca nici un singur Gateway.

3. Arhitectura „ Hub & Spoke”.

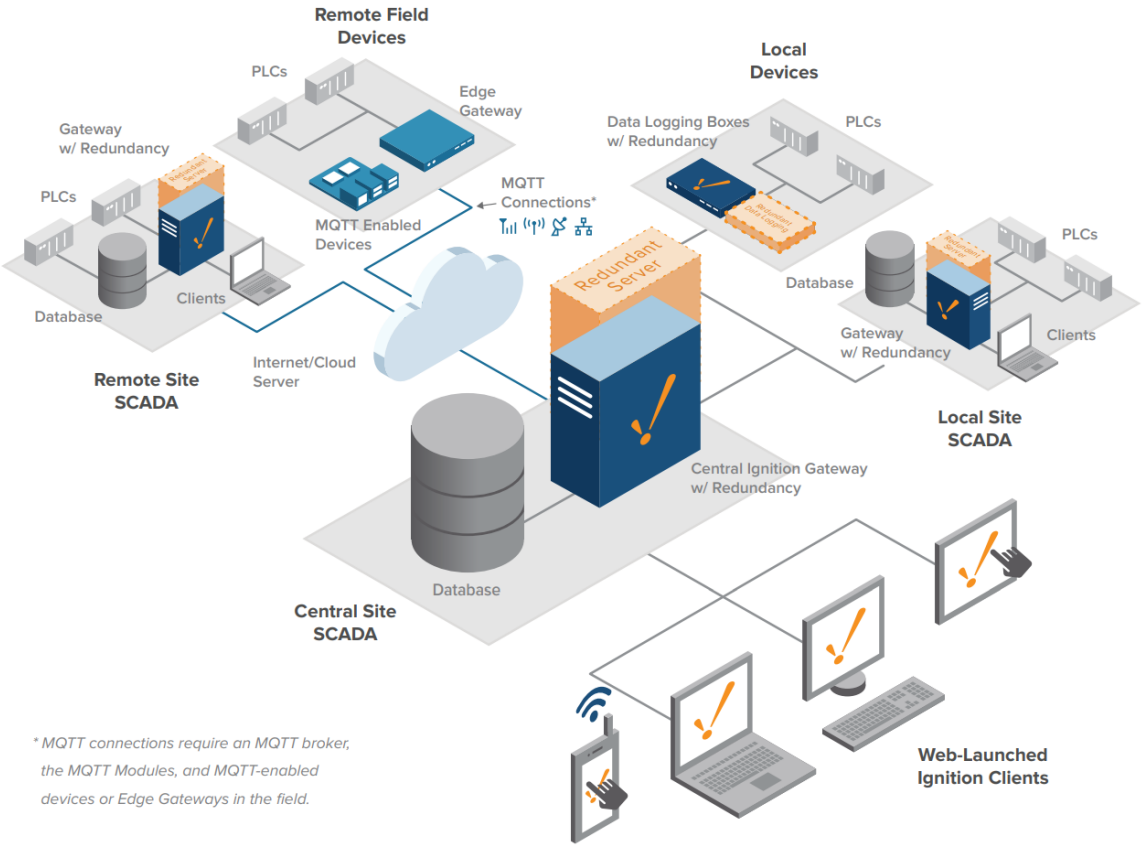


Figura 5 – Arhitectura SCADA „ Hub & Spoke” cu redundanta.

Arhitectura „hub and spoke” este formată din două componente. Componenta „HUB” constă dintr-un Gateway central Ignition (cu redundanță) cu module Vision, Reporting și Mobile și un server de bază de date. Componenta „SPOKE” constă dintr-un Gateway Ignition (cu redundanță) cu module OPC UA și SQL Bridge, dedicate pentru înregistrarea datelor.Fiecare site este complet independent, funcționând cu propriul istoric, alarme și clienți, Gateway-ul clientului fiind utilizat pentru coordonare și stocarea istoricului pe termen lung. In cazul in care cade comunicație intre serverul central si cel remote, datele sunt stocate in cel remote pana la revenirea acesteia. Odată cu revenirea comunicație, datele care trebuie salvate in istoricul pe teren lung se sincronizează automat. Observam ca o instalație aflata la distanta comunica cu gateway–ul central prin protocol MQTT care este un modul de comunicație.

4. Arhitectura „ Enterprise”.

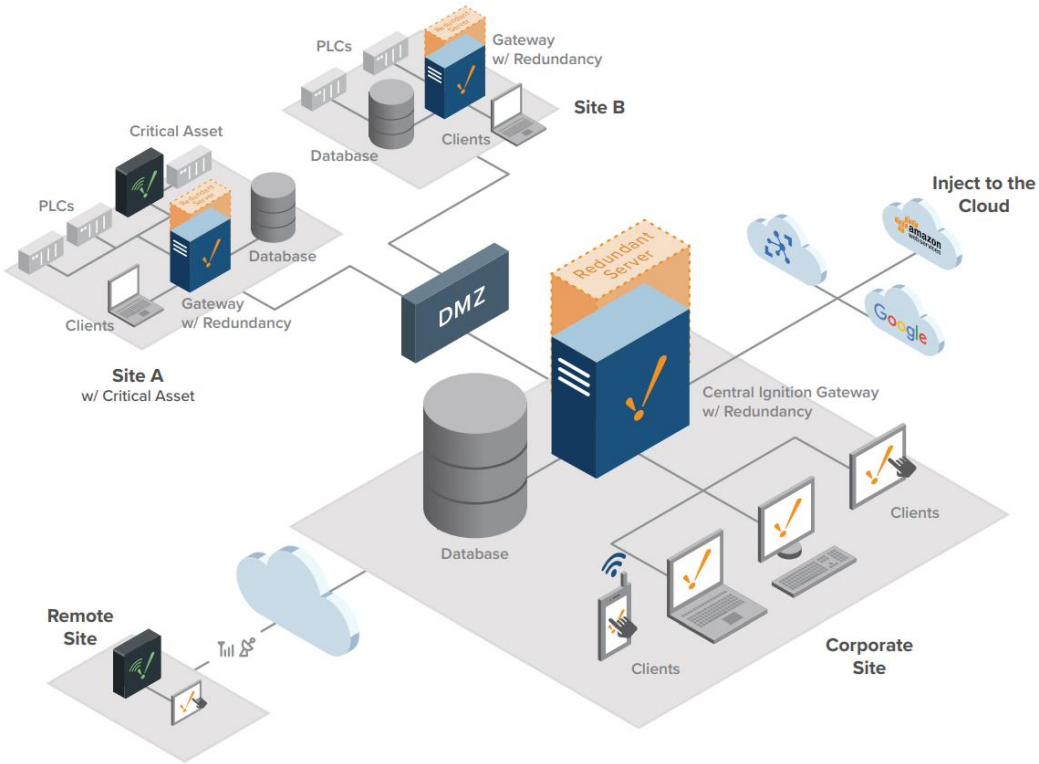


Figura 6 – Arhitectura SCADA „ Enterprise” cu redundanta.

Arhitectura Enterprise Ignition (cu redundanță) ne permite să creați un sistem conectat si securizat in același timp pe mai multe nivele. Astfel la un Server Central al companiei se conectează mai multe site -uri care conțin date critice .

Utilizând Ignition Edge ne asiguram că datele de la activele critice nu sunt niciodată alterate. Conectarea printr-un DMZ oferă un nivel suplimentar de securitate pentru transferul și accesarea datelor.

Ignition Gateway se conectează cu ușurință la servicii de Cloud precum Microsoft Azure, AWS, IBM Cloud și Google Cloud pentru stocare și analiză.

5. Arhitectura „ IIOT”.

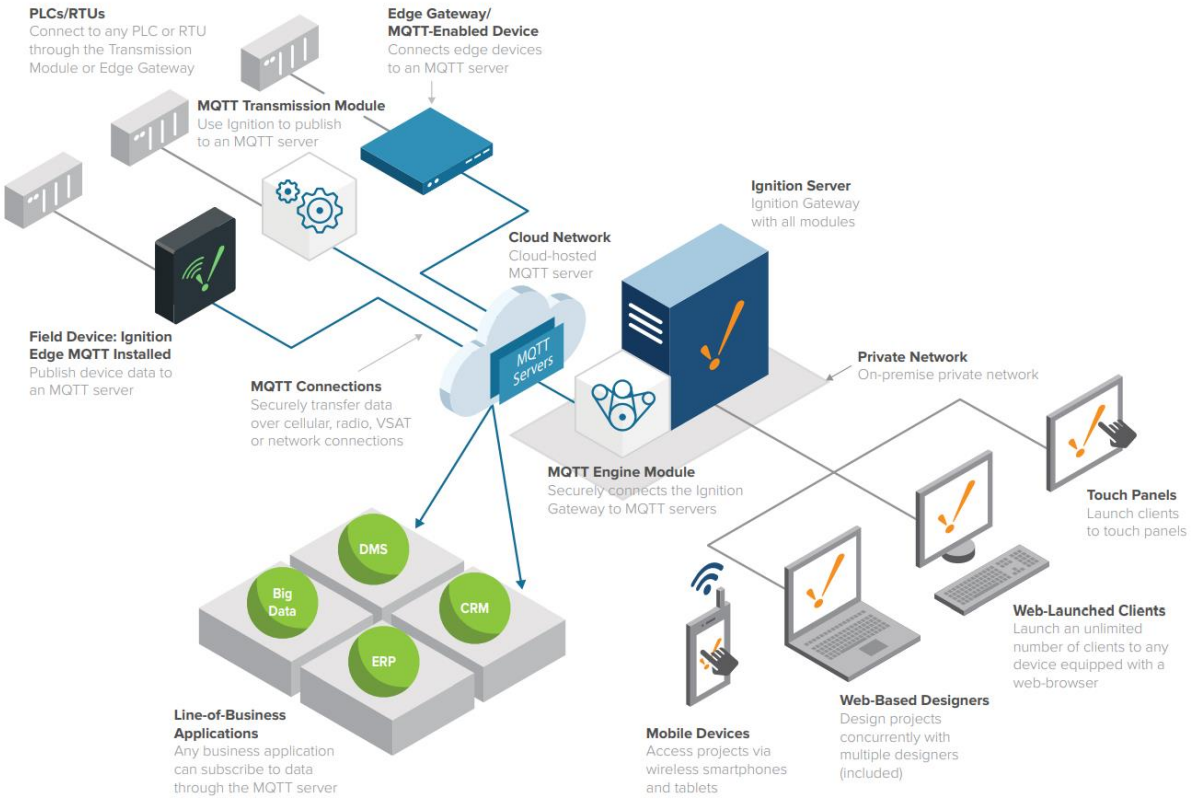


Figura 7 – Arhitectura SCADA „ IIOT”.

Ignition IIoT poate colecta date de pe orice dispozitiv de la marginea rețelei (Edge), poate publica acele date către un broker central și poate transmite acele date către aplicațiile industriale și de linie de „business” abonate.

Ignition IIoT se poate conecta la PLC-uri din teren prin utilizarea modulului de transmisie MQTT, a dispozitivelor de câmp cu Ignition Edge MQTT instalat și/sau a gateway-urilor Edge activate cu MQTT și a dispozitivelor de câmp care utilizează specificația Cirrus Link Sparkplug MQTT.

Aceste date sunt publicate unui broker MQTT, acest broker poate fi localizat „on-premise”, în cloud sau in configurație hibrida dintre cele două.

Modulul MQTT Engine situat pe un Gateway Ignition se poate abona la orice date publicate de la broker, aceste date putând fi utilizate în orice aplicație Ignition.

6. Arhitectura „ Cloud Hybrid”.

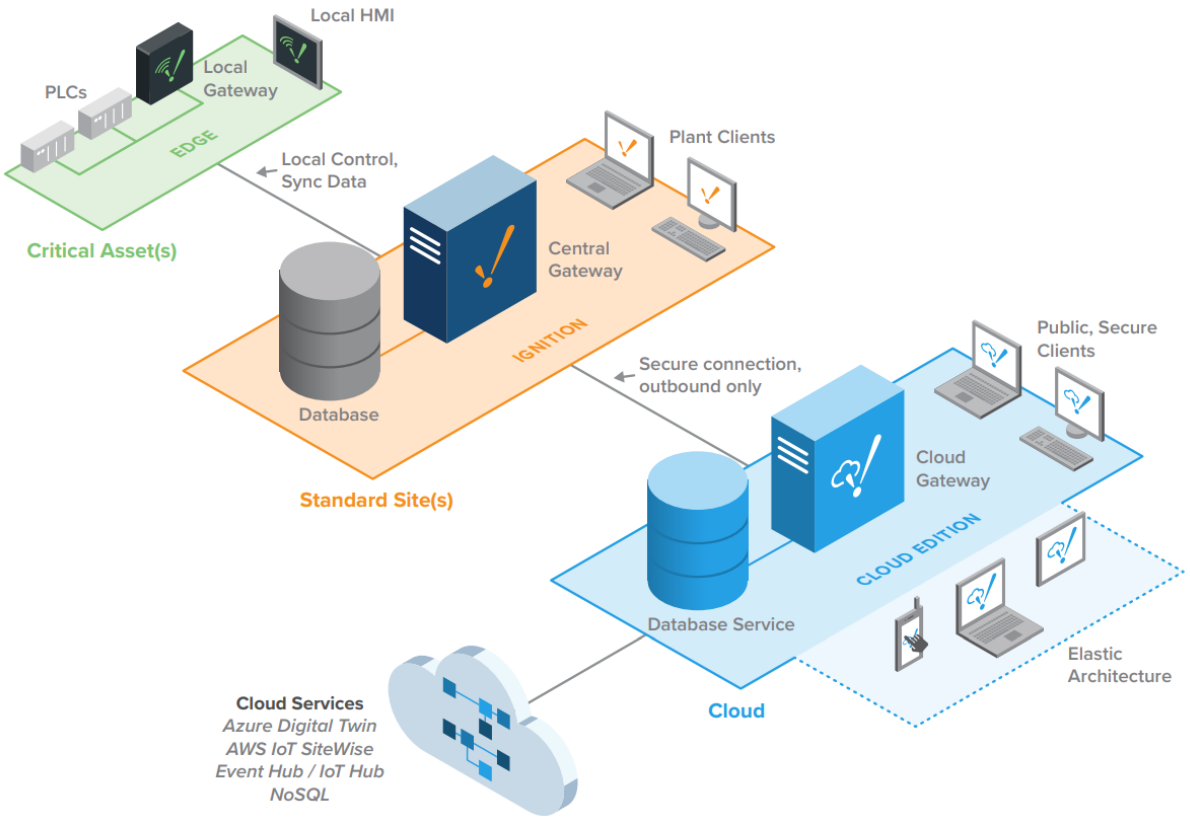


Figura 8 – Arhitectura SCADA „ Cloud - Hybrid”.

Arhitectura „Cloud Hybrid” ne permite să securizam și să distribuim datele la orice nivel. Utilizarea Ignition Edge ne asigura că datele de la activele critice nu sunt niciodată compromise iar utilizarea Ignition standard ne permite conectarea la mai multe site-uri la un server corporativ central.

Ignition Cloud Edition este utilizat pentru a construi arhitecturi de întreprindere flexibile care se extind la cerere folosind servicii de Cloud Public sau privat.

Deoarece AWS și Azure găzduiesc Ignition Cloud Edition, utilizatorii nu sunt responsabili pentru întreținerea hardware-ului serverului cloud; aceasta înseamnă că se poate furniza și elibera putere suplimentară de stocare și cloud computing în funcție de cerere, permițând dezvoltarea și să implementarea de aplicații de „Enterprise” ,mai rapid.

## 6. Securitatea sistemele SCADA – amenințări de securitate, soluții pentru asigurarea securității in sistemele SCADA. Directiva NIS, legea 362/2018.

Majoritatea rețelelor, inclusiv rețelele sistemelor SCADA, au anumite probleme de securitate comună si elemente de control corespunzătoare. O considerație importantă pentru retele SCADA este că acestea nu își pot permite întârzieri nedeterministe, mecanisme de Securitate care necesită capacități mari de memorie, blocarea operatorilor și relative timpi lungi de procesare. Cu toate acestea, unele dintre măsurile de securitatea fundamental disponibile sistemelor SCADA sunt similare cu cele utilizate pentru OSI și Arhitecturi stratificate TCP/IP. Cele mai bune practici de rețea includ protejarea confidențialitatea, integritatea, disponibilitatea a datelor împreună cu furnizarea de servicii de autentificare și acces.

In Romania Legea nr. 362/2018 privind asigurarea unui nivel comun ridicat de securitate a rețelelor și sistemelor informatice a intrat în vigoare si transpune așa-numita Directivă NIS (Directiva (UE) 2016/1148 a Parlamentului European și a Consiliului din 6 iulie 2016 privind măsuri pentru un nivel comun ridicat de securitate a rețelelor și a sistemelor informatice în Uniune) și are drept scop creșterea nivelului de pregătire a statelor UE pentru a face față la incidentele de securitate informatică și respectiv creșterea gradului de încredere a cetățenilor în Piața Digitală Unică.

Aceasta se adresează specific:

1. Operatorilor de Servicii Esențiale (OSE) din 7 sectoare de activitate economică astfel:
   I. Energie
   II. Transport
   III. Sectorul bancar
   IV. Infrastructuri ale pieței financiare
   V. Sectorul sănătății
   VI. Furnizarea și distribuirea de apă potabilă
   VII. Infrastructură digitală
2. Furnizorilor de Servicii Digitale (FSD) din trei categorii, respectiv: piețe online, motoare de căutare online, servicii de cloud computing .

Legea stabilirea de măsuri și cerințe pentru asigurarea efectivă a securității precum si obligativitatea notificarea incidentelor survenite.

În vederea asigurării unui nivel comun de securitate a rețelelor și sistemelor informatice, operatorii de serviciile esențiale și furnizorii de servicii digitale au obligația de a respecta normele tehnice elaborate de CERT-RO care elaborează, cu consultarea autorităților care reglementează sectoarele și subsectoarele prevăzute, ghiduri în sprijinul implementării măsurilor minime de securitate pentru operatorii și furnizorii de servicii esențiale.

Normele tehnic aplicabile operatorilor de servicii esențiale se stabilesc în baza cel puțin a următoarelor categorii de activități de asigurare a securității rețelelor și sistemelor informatice:

1. managementul drepturilor de acces;
2. conștientizarea și instruirea utilizatorilor;
3. jurnalizarea și asigurarea trasabilității activităților în cadrul rețelelor și sistemelor informatice;
4. testarea și evaluarea securității rețelelor și sistemelor informatice;
5. managementul configurațiilor rețelelor și sistemelor informatice;
6. asigurarea disponibilității serviciului esențial și a funcționării rețelelor și sistemelor informatice;
7. managementul continuității funcționării serviciului esențial;
8. managementul identificării și autentificării utilizatorilor;
9. răspunsul la incidente;
10. mentenanța rețelelor și sistemelor informatice;
11. managementul suporturilor de memorie externă;

12. asigurarea protecției fizice a rețelelor și sistemelor informatice;
13. realizarea planurilor de securitate;
14. asigurarea securității personalului;
15. analizarea și evaluarea riscurilor;
16. asigurarea protecției produselor și serviciilor aferente rețelelor și sistemelor informatice;
17. managementul vulnerabilităților și alertelor de securitate

O parte din masurile de securitate aplicabile sistemelor SCADA sunt descrise in continuare.

**Firewall-uri**

Un element cheie de securitate de protecție care este necesar oricărei rețele conectate la o rețea neîncrezătoare, cum ar fi Internetul, este un firewall. Firewall oferă protecția împotriva virușilor, viermilor și a altor tipuri de coduri rău intenționate, precum și de la intruziunile în rețea. O problemă cu firewall-urile aplicate sistemelor SCADA este că majoritatea firewall-urilor nu acceptă gestionarea protocoalelor SCADA. Această situație este cercetat de o serie de organizații și unele SCADAaware firewall-urile sunt în curs de dezvoltare.

**Firewall-uri proxy**

Firewall-urile de nivel proxy sau de aplicație funcționează la nivelul 7 al modelului OSI. În dicționar, un proxy este definit ca o persoană autorizată să acționeze în numele altuia; un agent sau înlocuitor. Astfel, software-ul proxy poate fi plasat între un utilizator și un server pentru a ascunde identitatea utilizatorului. Serverul vede proxy-ul și nu poate identifica utilizatorul. Scenariul este valabil și în situația inversă în care utilizatorul interacționează cu software-ul proxy în fața serverului și nu se poate identifica serverul sau rețeaua asociată acestuia. Apoxy firewall este eficient în ecranare o rețea dintr-o rețea exterioară neîncrezătoare, cum ar fi Internetul.

**Zonă demilitarizată**

Firewall-urile pot fi folosite pentru a implementa arhitecturi de rețea de securitate care sunt eficient pentru sistemele SCADA. Aceste arhitecturi se bazează pe conceptul de o zonă demilitarizată sau DMZ. Un DMZ este o regiune care oferă o separare între o rețea externă sau publică și o rețea internă sau privată. În pentru ca un firewall să suporte un DMZ, acesta trebuie să aibă mai multe interfețe externe și listele corespunzătoare de control al accesului, acolo unde este necesar. Cateva diferite Arhitecturile folosesc DMZ-uri, dar există două care sunt aplicabile în special medii de achiziție și control de date. Aceste arhitecturi sunt unice firewall DMZ și un firewall dual DMZ. Ele pot servi scopului separării o rețea de întreprindere corporativă din rețeaua de control în timp ce furnizează o conexiune pentru ambele la o rețea publică, cum ar fi Internetul.

**Un singur firewall DMZ**

Într-un singur firewall DMZ, un firewall este folosit pentru a filtra pachetele de date din, pentru de exemplu, o rețea de întreprindere către rețeaua locală de control și de la o rețea externă rețea. DMZ conține elementele care trebuie accesate de către calculatoare de întreprindere, precum și conexiunea la rețeaua publică exterioară. Această arhitectură este prezentată în Figura 3-16. Deoarece nu există firewall între DMZ și rețeaua de control, controlul rețeaua este potențial vulnerabilă dacă DMZ este pătruns de un atac din rețeaua externă sau prin rețeaua întreprinderii.

## 7. Bibliografie


a. Internation Society of Automation: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa112
b. https://www.geeksforgeeks.org/how-communication-happens-using-osi-model/
c. https://inductiveautomation.com/ignition/architectures
d. Directoratul National de Securitate Cibernetica: https://dnsc.ro/pagini/ansrsi
e. Flexible Solutions for Your Supervisory Control and Data Acquisition Needs - Rockwell Automation Publication AG-SG001G-EN-P - April 2015
f. SCADA System - Application Guide - Publication AG-UM008C-EN-P - February 2005
g. Securing SCADA Systems, Ronald L. Krutz, Wiley Publishing, Inc.

# Activitate practică 1

## Alegerea echipamentelor unui sistem de automatizare.

**SE DA:**

Inginerii tehnologi  si hidraulisti vor furniza o **diagrama P&ID**, o **lista cu motoare si traductoare** si o **filozofie de control**  pentru o instalație  nou proiectata.

Schema instalației este prezentata in figura 1.

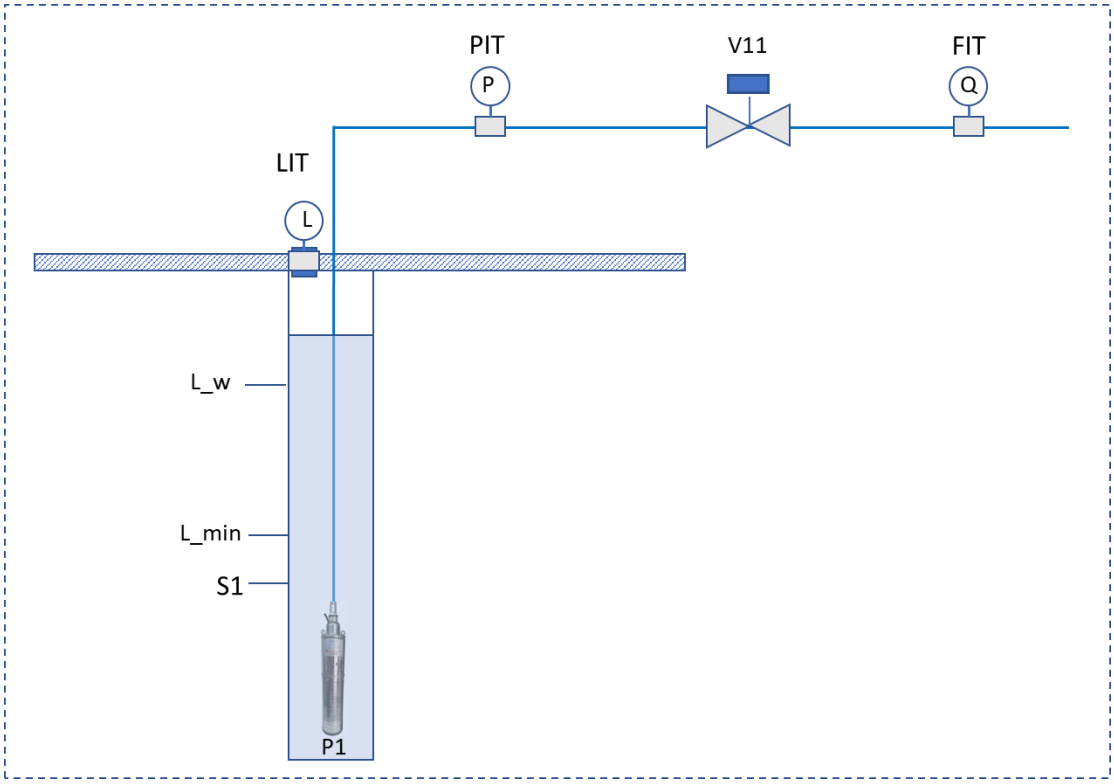


Figura 1 – Instalație de foraj apa bruta

Funcționarea instalației este prezentata in continuare:

Instanția reprezintă un sistem de foraj a apei brute , extrase din pânza freatica care mai apoi este transferata către o stație de tratare a apei.

**Apa bruta** este extrasa dintr-un foraj construit la o adâncime de 30 de metri.  Extragerea apei brute se realizeze cu o pompa submersibila de 5kw montata in foraj, care este acționata cu un convertizor de frecventa.

**Forajul** are prevăzut cu un traductor de nivel (**LIT1**) care măsoară nivelul apei din foraj. Atunci când nivelul scade sub un nivel care este parametrizat din interfața de operare si care are valoarea inițiala de **2m** pompa se oprește. După ce pompa se oprește datorita scăderii nivelului in foraj, se așteaptă pana când nivelul se reface ajungând la un pragul de funcționare la care pompa va porni. Acest prag

este de asemenea parametrizat si are valoarea inițiala de **18 m.** Pentru siguranța, pragurile software de oprire a pompei este dublat de un senzor de siguranța de tip plutitor care va da un semnal digital - **S1.** Acesta va fi montat puțin mai jos decât nivelul setat ca limita pentru valoarea citita de la traductor .

Pe țeava care transporta apa din foraj către stația de tratare este prevăzut un traductor de presiune in domeniul 0 – 10 bari care furnizează un semnal de 4 – 20 mA. După traductorul de presiune este prevăzută o vana de izolare , **V11**. Este necesara o ieșire digitala pentru comanda acesteia si doua intrări pentru confirmările de închis si deschis a vanei. După vana de izolare este montat si un debitmetru care va da informații de măsura a debitului instantaneu si va afișa si totalizatorul de debit prin foraj.

**SE CERE:**

Dimensionați un sistem de automatizare cu PLC pentru instalația data utilizând aplicația IAB (Integrated Arhitecture Buider) prezentata in curs.

Se vor respecta următoarele cerințe suplimentare:

1. Realizarea unui tabel in excel sau un program similar cu obiectele si componentele instalației care sa cuprindă numărul de I/O si tipul acestora necesare pentru controlul instalației.
2. Pornind de la numărul de I/O se va realiza un proiect in IAB pentru un sistem cu PLC Micro800.
3. Se vor utiliza intrări suplimentare de tip plugin si expansion slot.
4. Se va adăuga o interfața de operare din gama PV800 de 10”.
5. Conexiunea cu PLC-ul se va realiza pe o rețea EthernetIP prin intermediul unui switch Stratix fără management cu 8 porturi.
6. Pompa va fi acționata prin intermediul unui convertizor de frecventa de tip PowerFlex 525 conectat pe EthernetIP.
7. Se va prevedea o centrala de măsura parametri energetici de tip ET1000 conectata pe EthernetIP.
8. Se va prevedea o sursa de alimentare pentru echipamentele din dulapul de automatizare.

## Obiective

1. Calculul numărului de semnale de intrare si ieșire necesare pentru controlul instalației date.
2. Crearea unei liste cu echipamente de automatizare necesare pentru realizarea unui tablou de automatizare cu PLC care sa controleze instalația propusa si sa respecte funcționalitățile acesteia.
3. Utilizarea aplicației software IAB pentru configurarea si validarea unui sistem de control pentru instalația data.

# Activitate practică 2

# Elaborarea unei aplicații de PLC pentru instalația data

**SE DA:**

1. Lista cu echipamente de automatizare necesare pentru realizarea unui tablou de automatizare cu PLC care sa controleze instalația propusa si sa respecte funcționalitățile acesteia.
2. Tabelul in excel sau un program similar cu obiectele si componentele instalației care cuprinde numărul de I/O si tipul acestora necesare pentru controlul instalației.

**SE CERE:**

Se va dezvolta o aplicație pentru PLC-ul rezultat in urma Aplicației Practice 1care va respecta următoarele cerințe:

1. Programul PLC va fi dezvoltat in CCW (Connected Components Workbench) , folosind limbajul Ledder Logic si va avea numele AP2_data.
2. Se va configura PLC-ul din lista de echipamente cu modulele necesare.
3. Se va atribui PLC-ului adresa de IP dintr-o clasa C privata: 192.168.0.10
4. Se va crea lista cu taguri ale proiectului plecând de la tabelul dezvoltat in AP1 si conform datelor de proiectare. Tagurile vor fi importate in aplicația CCW.
5. Se vor implementa variabile locale care sa semnifice condiții de funcționare.
6. Se vor scala mărimile analogice ce nivel, presiune si debit atât in mărimi electrice (mA) cat si in mărimile fizice caracteristice fiecărui traductor.
7. Se va implementa butonul de avarie generala.
8. Se va implementa funcționarea sistemului in doua regimuri de funcționare – **MANUAL** si **AUTOMAT** impus printr-o cheie software de pe interfața de operare.
9. In regim **MANUAL**, elementele de execuție vor funcționa in felul următor: pompa P1 va porni sau o frecventa fixa stabilita de operator doar daca vana de separație este deschisa; vana de separație V1 se va deschide si se va închide prin acțiunea a doua butoane software de pe interfața de operare – Închidere si Deschidere.
10. In regim **AUTOMAT** , se va implementa următorul algoritm de control pentru sistemul de foraj: la apăsarea unui buton **START AUTOMAT** , vizibil numai in acest **Regim Automat** , va porni pompa P1 daca are condiții de pornire - nivel mai mare decât un prag si daca vana V11 este deschisa.
11. **Secvența de pornire** se desfășoară astfel: - Se verifica nivelul in foraj si daca este mai meare decât L_min atunci se deschide vana V11 si după ce aceasta confirma ca este deschisa, atunci va porni pompa P1. Frecventa cu care va porni pompa este de 40 Hz.

12. **Secvența de oprire** se desfășoară astfel: - daca nivelul in foraj  a ajuns la nivelul minim atunci se oprește pompa de foraj apoi se închide vana V11. Se așteaptă pana când nivelul in foraj ajunge la valoarea L_W si apoi se reia secvență de pornire.
13. La apăsarea unui buton **STOP AUTOMAT** pompa de foraj se va opri apoi se va închide si vana V11.
14. Se va contoriza timpul de funcționare pentru pompa de foraj.

## Obiective

1. Proiectarea unei aplicații software pentru PLC din seria Micro800.
2. Utilizarea CCW pentru dezvoltarea acesteia.
3. Implementarea acestei aplicații in limbajul de programare ledder logic.

# Activitate practică 3

# Elaborarea unei aplicații HMI pentru instalația data

**SE DA:**

1. Aplicația dezvoltata in activitatea practica 2.

**SE CERE:**

Se se dezvolte o aplicație pentru interfața de operare Panle View 800 – HMI (human machne interface ) care va respecta următoarele cerințe:

1. In Proiectul care conține programul PLC dezvoltat in CCW (Connected Components Workbench) , in cadrul **Activității Practice 2**, se va adăuga un echipament PanelView 800 de 10”.
2. Se va configura PanelView-ul astfel încât sa comunice cu PLC-ul din proiect.
3. Se vor importa toate tagurile din aplicația de PLC in aplicația de HMI.
4. Se vor seta proprietăți implicite pentru ecranele ce se vor dezvolta.
5. Se vor crea un număr de patru ecrane. Fiecare ecran va conține butoane de navigare care vor face posibila navigarea intre ecrane. Fiecare ecran va conține un titlu , data si ora afișata in partea de sus a acestuia.
6. Un prim ecran va fi de introducere, in care se vor afișa date despre instalație, starea de funcționare. Se vor prevedea butoanele de selecție de Regim Automat si Manual, precum si butonul de Start Instalație. Se vor afișa condițiile de funcționare. Regimul de funcționare va fi afișat in fiecare ecran.
7. Al doilea ecran va conține instalația de foraj cu toate elementele componente. Aici vor fi afișate si butoanele de deschidere – închidere pentru vana, Start si Stop pentru pompa si impunerea de frecventă în regim manual cat si in regim automat. Vor fi afișate si valorile fizice furnizate de traductorii de presiune, nivel, debit. Se va afișa si contorul de debit.
8. Al treilea ecran va conține parametri pentru scalarea semnalelor analogice. Se vor introduce parametri de scalare, se vor afișa mărimile furnizate de traductor in CAN-uri, semnal unificat , valoare fizica corespondenta.
9. Al patrulea ecran este dedicat semnalelor de alarma.

## Obiective

1. Proiectarea unei aplicații software pentru interfața HMI de tip PanelView800.
2. Utilizarea software-ului CCW pentru dezvoltarea acesteia.

# SENZORI SOFTWARE

**- note de curs –**

## I. Introducere

Datorită schimbărilor climatice din ultimul deceniu, mai ales a prezenței secetei în România, disponibilitatea apei a devenit o problemă îngrijorătoare datorită nevoii tot mai mari pentru irigarea suprafețelor de către agricultori și nu numai cât și cererea creșterii populației și dezvoltarea industriei. Ținând cont de aceste cerințe, o eficientizare a resursei de apă este ca apa reziduală să fie tratată pentru a putea fi deversată în circuitul natural.

Misiunea de a trata apa reziduală nu este deloc o sarcină simplă, deoarece acest proces trebuie să respecte standardele reglementate de Uniunea Europeană privind calitatea apei. Procesul de tratarea apelor uzate necesită instrumentație adecvată capabilă să genereze informații relevante [1] care să reflecte acuratețea variabilelor care sunt într-o continuă schimbare, datorită, în special schimbărilor climatice. Instrumentația existentă în literatura de specialitate implică utilizarea senzorilor hardware și este de trei tipuri: instrumentație on-line, off-line și in-line. Informațiile furnizate de senzori sunt de o reală importanță atât pentru calitatea apei cât și pentru controlul întregului proces de tratare a apei.

Deși, utilizarea senzorilor este o soluție modernă și la îndemână în ceea ce privește monitorizarea procesului de tratare al apelor, o atenție deosebită trebuie acordată asupra duratei ciclului de viață și a uzurii acestora. Având în vedere situația economică și dificultatea în identificarea defecțiunilor hardware ale senzorilor, este esențială găsire de soluții pentru a îmbunătăți fiabilitatea senzorilor sau înlocuirea senzorilor hardware cu senzori software.

## II. Monitorizarea parametrilor de proces în stațiile de epurare a apelor uzate via senzori software

Monitorizarea parametrilor în procesele de epurare reprezintă o componentă fundamentală în automatizarea instalațiilor de epurare a apelor uzate. Scopul principal al monitorizării este de a obține o optimizare a întregului proces de epurare/tratare a apelor uzate. Prin monitorizarea constantă a parametrilor cheie, se urmărește atingerea următoarelor obiective:

- **Minimizarea pierderilor de timp și resurse:** Prin evaluarea continuă a parametrilor, se pot detecta și corecta rapid abaterile de la parametrii ideali, reducând astfel pierderile de timp și resurse în procesul de epurare.
- **Planificarea mentenanței și a reparațiilor:** Monitorizarea constantă oferă date esențiale pentru planificarea întreținerii și reparațiilor preventive. Astfel, se pot evita defecțiunile majore și se poate menține instalația în funcțiune în cea mai bună stare.
- **Eficiența energetică:** Prin monitorizarea și ajustarea parametrilor în timp real, se poate optimiza consumul de energie al instalației. Acest lucru duce la o reducere semnificativă a costurilor operaționale și la o economie de energie substanțială pe termen lung.

În concluzie, monitorizarea parametrilor în procesele de epurare a apelor uzate nu este doar o activitate esențială, ci și o investiție inteligentă pentru optimizarea procesului de epurare, reducerea pierderilor și creșterea eficienței în utilizarea resurselor.

## III. Definiții și beneficii senzori software

- **Senzor** – este un dispozitiv care măsoară o mărime fizică (presiune, lumină, temperatură, umiditate etc) și o transformă într-un semnal care poate fi citit de către un observator printr-un instrument [2].
- **Senzor software** – termenul combină cuvintele „software” deoarece modelele de evaluare a semnalului senzorului sunt de obicei programe de calculator și „senzori” deoarece aceste modele furnizează informații prezum senzorii hardware [3]. Aceasta înseamnă că semnalul este produs de software în loc de senzor hardware (vezi Figura 1).

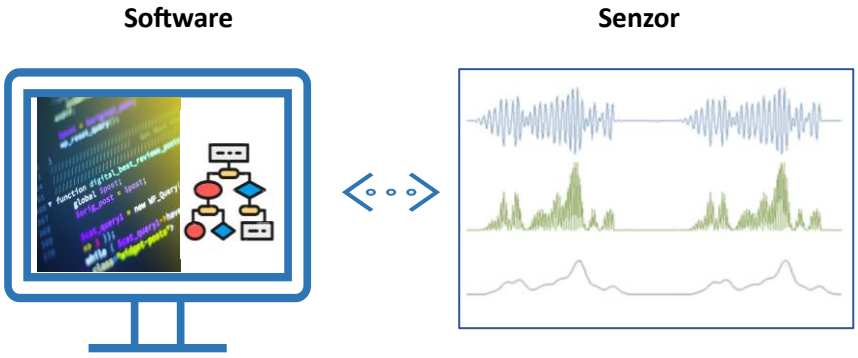


***Fig. 1*** Reprezentare grafică senzor software

Stațiile de epurare a apelor utilizează o varietate de senzori pentru a monitoriza și controla diferiți parametri în timpul procesului de tratare a apelor uzate. Acești senzori contribuie la asigurarea funcționării eficiente și a calității apei tratate.

O clasificare a senzorilor utilizați în procesul de tratare al apelor este ilustrată în Figura 2, însă dintre cei mai comuni întâlniți sunt:

- **Senzori de nivel:** Acești senzori monitorizează nivelul apei în diverse părți ale stației de epurare, cum ar fi rezervoarele, bazinul de sedimentare sau canalele de colectare.
- **Senzori de debit:** Senzorii de debit măsoară cantitatea de apă care intră sau iese din diferite etape ale procesului de epurare. Aceste date sunt esențiale pentru a controla fluxurile de apă.
- **Senzori de oxigen dizolvat:** Oxigenul dizolvat în apă este important pentru supraviețuirea bacteriilor care descompun materiile organice în procesul de epurare. Senzorii de oxigen dizolvat ajută la monitorizarea concentrației de oxigen în timp real.
- **Senzori de pH:** pH-ul apei este un indicator important pentru controlul proceselor chimice din stația de epurare. Senzorii de pH asigură că valoarea pH-ului se menține în intervalul potrivit pentru reacțiile chimice necesare.
- **Senzori de turbiditate:** Turbiditatea măsoară cât de clară sau cât de tulbure este apa. Senzorii de turbiditate ajută la evaluarea calității apei și la monitorizarea eficacității procesului de epurare.
- **Senzori de temperatură:** Temperatura apei poate afecta procesele microbiologice și chimice din stația de epurare. Senzorii de temperatură asigură monitorizarea acestei variabile.
- **Senzori de substanțe chimice:** În funcție de necesități, stațiile de epurare pot utiliza senzori specializați pentru a măsura concentrații specifice de substanțe chimice, cum ar fi clorul sau substanțele toxice.
- **Senzori de nivel de soliditate:** Acești senzori măsoară nivelul de materiale solide (de exemplu, nisip sau sediment) în apă, ajutând la gestionarea sedimentelor și a deșeurilor.

- **Senzori de gaz:** Unele stații de epurare pot folosi senzori pentru detectarea gazelor prezente în timpul procesului de epurare, precum hidrogenul sulfurat (H2S), care poate fi toxic.
- **Senzori de gaz metan:** Acești senzori pot fi utilizați în procesele de digestie anaerobă pentru a monitoriza producția de gaz metan.


*Fig. 2* Tipuri de senzori utilizați în industria apei

**Ce sunt senzorii software în industria apelor uzate?**

Senzorii software din industria apelor uzate se referă la instrumente digitale sau aplicații care utilizează algoritmi software pentru a monitoriza și analiza datele legate de procesele de tratare a apelor uzate. Acești senzori folosesc modele bazate pe date și strategii de control pentru a furniza informații și pentru a optimiza operațiunile stației de tratare a apelor uzate.

**Avantajele utilizării senzorilor software în industria apelor**

În epoca tehnologiei avansate, folosirea senzorilor software pentru gestionarea proceselor de tratare a apelor uzate a devenit o metodă din ce în ce mai semnificativă. Senzorii software permit colectarea și monitorizarea datelor în timp real, eliminând necesitatea instalării senzorilor hardware adiționali. Integrarea senzorilor software în controlul proceselor de tratare a apei oferă numeroase avantaje și

contribuie la îmbunătățirea eficienței, performanței și durabilității proceselor de tratare. O serie dintre principalele avantaje ale utilizării senzorilor software în industria apelor sunt [4]:

- **Monitorizarea și control în timp real**: Senzorii software permit monitorizarea în timp real prin intermediul internetului a parametrilor critici, cum ar fi calitatea apei, nivelul apei, presiunea și consumul. Aceste avantaje asigură detectarea rapidă a problemelor și permite luarea de măsuri imediate.
- **Costuri reduse**: Senzorii software elimină necesitatea de înlocuire dacă vorbim de uzură sau întreținerea instalațiilor de tratare ale apelor uzate;
- **Eficiență operațională**: Datele în timp real oferite de senzori software facilitează optimizarea proceselor. Acest lucru duce la economii de energie, apă și alte resurse, contribuind la o operare mai eficientă.
- **Flexibilitate:** Senzorii software permit configurare personalizată și pot fi adaptați în funcție de necesitățile instalațiilor de tratare a apelor uzate;
- **Scalabilitate:** Operațiune care permite înlocuirea și adăugarea de senzori fără a fi nevoie de modificări ale instalațiilor de tratare a apelor uzate.
- **Intervenție rapidă**: Senzorii software detectează devierile de la parametrii normali și pot declanșa alarme. Lucru permite intervenția rapidă pentru a preveni potențiale avarii sau scăpări.
- **Acces facil la date și gestionare de la distanță**: Datele colectate pot fi ușor de accesat prin intermediul internetului fără a mai fi nevoie de unități hardware;
- **Actualizări software**, senzorii pot fi îmbunătăți prin adăugarea de noi funcționalități sau de a remedia probleme;
- **Predicția:** Senzorii software oferă posibilitatea de a deduce măsurători ale variabilelor nemăsurabile;
- **Permite calculul fiabil** al parametrilor în cazul în care nu este disponibil niciun senzor hardware;
- **Reduce riscul contaminării bioreactorului**;

## IV. Modele pentru senzori software

Metodologia generală de dezvoltare a unui senzor software implică mai multe etape, cum ar fi:

- Colectarea datelor,
- Inspecția datelor,
- Selectarea datelor istorice,
- Preprocesarea datelor,
- Selectarea unui model matematic,
- Instruirea și validarea modelului,
- Integrarea senzorului software
- Monitorizarea și întreținerea senzorului software.

În general, există trei modele pentru senzori software (vezi Figura 3) [4]:

- model driven,
- model data-driven,
- model gray-box

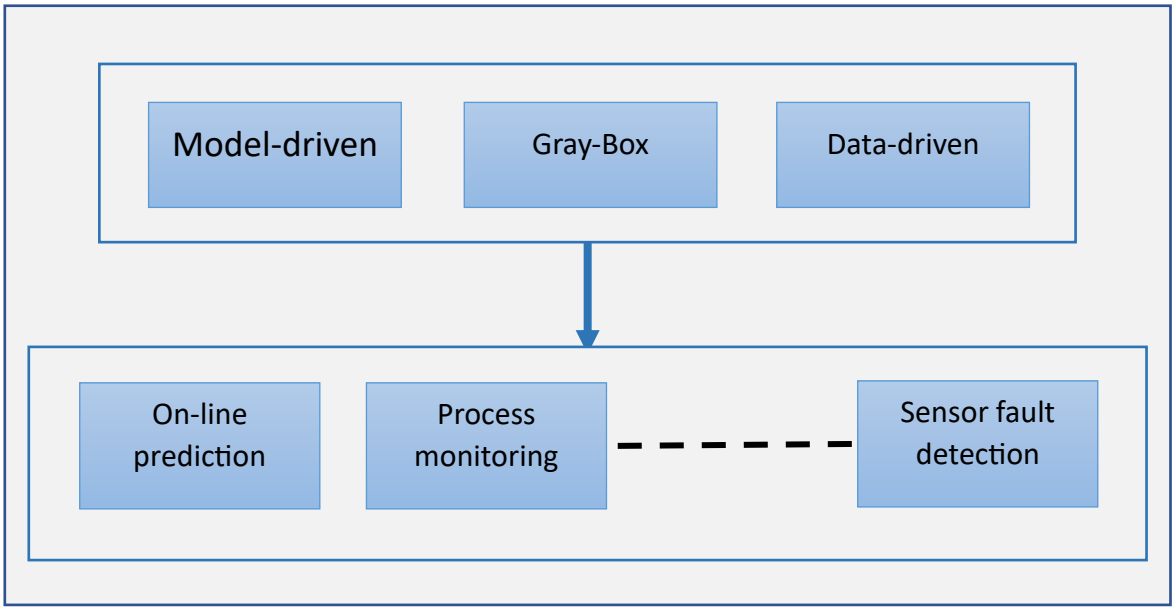


***Fig.* 3** Modele senzori software [4]

**Senzorii model-driven** (sau cutie albă) se bazează pe cunoașterea fenomenologică completă a procesului [4]. Sunt dispozitive care funcționează pe baza unor modele matematice și algoritmi. Acești senzori sunt concepuți pentru a măsura, monitoriza și colecta date în conformitate cu modelele matematice predefinite sau cu algoritmii specifici.

**Senzorii data-driven** (sau cutie neagră) se bazează pe date istorice [4] și sunt dispozitive care colectează date și oferă rezultate fără a dezvălui detalii despre modul în care funcționează procesul intern sau algoritmul folosit. Acești senzori sunt centrați pe colectarea și furnizarea de date fără a expune procesul de generare a acestor date

**Senzorii gray-box** sunt o combinație de senzori software bazați pe model și pe date [4]. Acești senzori oferă o combinație de caracteristici, oferind atât date precise, cât și transparență controlabilă în ceea ce privește funcționarea lor internă

## V. Aplicabilitatea senzorilor în procesul de tratare a apelor uzate

În ceea ce privește interesul aplicabilității senzorilor în procesul de tratare a apelor uzate este unul crescut deoarece colectarea unui număr cât mai mare de date oferă operatorilor umani capabilitatea de a anticipa luarea de decizii. Astfel, în abordarea [5], a fost testată și validată o metodă alternativă rentabilă pentru monitorizarea parametrilor esențiali de calitate a apei uzate, cum ar fi TP (Total Phosphor) și COD (Chemical Oxygen Demand), într-o stație de tratare a apelor uzate la scară largă. În [6], s-a demonstrat eficiența unei metode de detectare a defecțiunilor senzorului DO (dissolved oxygen), based on Principal Components Analysis (PCA). Rezultate satisfăcătoare au fost raportate în [7] , unde s-a demonstrat eficiența unei tehnici de învățare soft sensors for complex phenomena to predict ammonium. For prediction of effluent chemical oxygen demand (COD) and total nitorgen (TN) with large variations în [8] an machine learning model based on IFFNN coupled with genetic algorithm was developed. În cazul estimării concentrațiilor de nutrienți din efluenți într-o stație de tratare biologică a apelor uzate, s-a abordat o metodă de învățatre hibridă care combină algoritmul genetic cu sistemul de inferență neuro-fuzzy adaptiv(GA-ANFIS) [9]. The results indicate that the hybrid GA-ANFIS soft sensors outperform ANFIS-based soft sensors in terms of effluent prediction accuracy [9]. Cu toate acestea, în studiul [10] s-au dezvoltat senzori soft cu mai multe ieșiri folosind modelul de regresie liniară

multivariată (MLR), mașină vectorială relevantă multivariată (MRVM) și modele de regresie a proceselor gaussiene multivariate (MGPR). Metoda propusă a fost validată prin aplicarea algoritmului la o stație de apă uzată fiind simulată cu modelul BSM1. În plus, în [11], sunt analizate rezultatele the neural network based MLP (multilayer perceptron) model that provides a better estimate than the corresponding MLR (traditional multiple linear regression). For soft-sensor modelling of the effluent COD, TN and TP concentration in a municipal ASP, Kim et al. [12] used MPCA and FFNN.

## VI. Studiu de caz - Implementarea în WWTP a unui senzor software bazat pe rețele neuronale LSTM

Prezentul studiu de caz oferă o examinare cuprinzătoare a încorporării ANN-urilor ca estimatori software în contextul epurării apelor uzate, cu un accent deosebit pe prezicerea concentrațiilor de amoniu în efluent.

Senzorul software propus, a fost testat în cadrul modelului BSM2 din Simulink pentru a determina cantitatea de $S_{NH,e}$ (amoniu) din efluent și de a prezice în timp util posibilele valori crescute ale acestuia [13].

- **Setul de date și modelul de simulare**: Pentru realizarea acestei aplicații, s-a folosit un set de date reprezentativ colectat din modelul BSM2, care include informații despre valorile și parametrii cheie, măsurătorile de calitate ale apei și variabilele de control. Acest set de date a fost utilizat pentru antrenarea și testarea rețelei neuronale LSTM [13].
- **Antrenarea și validarea rețelei neuronale LSTM**: Pentru a realiza predicții precise, rețeaua neuronală LSTM a fost antrenată utilizând algoritmi de învățare automată pe setul de date disponibil. Apoi, a fost validată performanța rețelei utilizând datele de validare pentru a evalua acuratețea predicțiilor sale [13].
- **Evaluarea performanței senzorului software**: Pentru a evalua performanța senzorului software implementat, au fost comparate predicțiile acestuia cu valorile reale măsurate în cadrul modelului de simulare WWTP, BSM2. Au fost utilizați mai mulți indicatori de performanță, cum ar fi rădăcina erorii medii pătratice (RMSE), eroare medie procentuală absolută (MAPE) și coeficientul de determinare ($R^2$), pentru a determina acuratețea și eficacitatea senzorului software [13].

Măsurătorile de efluent și efluenți ale stațiilor de epurare au fost colectate pe o perioadă de un an, cu un interval de prelevare de 15 minute.

Figura 4 ilustrează structura blocului de predicție a efluentului, precum și structura internă a blocului de procesare a datelor.

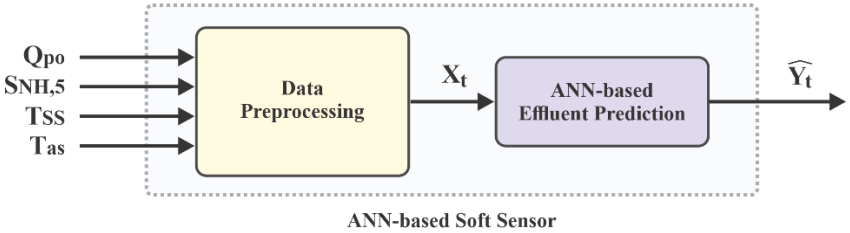


***Fig. 4*** *Structura completă a senzorului software bazat pe RNA* [13]

După cum putem vedea în figura de sus, intrările în sistem includ măsurătorile de influent și disponibile obținute de la stațiile de epurare. Aceste măsurători includ debitele de apă și concentrațiile de nutrienți, cum ar fi concentrația de amoniu în al cincilea rezervor al bioreactorului ($SNH$,5), debitul de ieșire de la primul clarificator ($Qpo$), temperatura mediului ($Tas$) și totalul de solide în suspensie ($STp$). Scopul ANN-ului (Artificial Neural Networks ) este pentru a prezice concentrația efluentului ($y\hat{t}$). ANN-urile se bazează pe o metodologie de predicție care implică utilizarea celulelor de memorie pe termen lung și pe termen scurt (LSTM).

Tehnicile de preprocesare a datelor joacă un rol critic în optimizarea performanței și reducerea complexității rețelelor neuronale artificiale (ANN). Prin urmare, abordarea de preprocesare a datelor include trei etape principale, și anume: fereastra glisantă, normalizarea datelor, antrenament bazat pe K-Fold.

Fereastra glisantă este o tehnică utilizată în preprocesarea datelor care implică împărțirea unei secvențe de date în segmente sau ferestre mai mici. Încorporează două variabile fundamentale, și anume lungimea ferestrei (WL) și orizontul de predicție (PH).

În cazul de față, configurația WL și PH a fost stabilită după cum urmează: O lungime a ferestrei (WL) de 10 ore a fost selectată ca interval de timp adecvat pentru a reține valorile observate la fiecare moment de prelevare și pentru a include măsurătorile precedente. A fost luat în considerare un Orizont de Predicție (PH) de 4 ore. Acest parametru definește perioada de timp în care previziunile concentrațiilor efluenților pot fi furnizate în avans, facilitând astfel luarea proactivă a deciziilor. După cum puteți vedea în Figura 5, sistemul asimilează datele anterioare înregistrate timp de 10 ore pentru fiecare măsurătoare nouă. Specificațiile arhitecturale ale stației de epurare luate în considerare au determinat ca timpul de reținere necesar să fie de 14 ore. Tehnica ferestrei glisante este implementată în așa fel încât la fiecare mișcare a ferestrei să fie generată o nouă măsură, în timp ce cea mai veche măsură este eliminată conform principiului First-In-First-Out (FIFO).

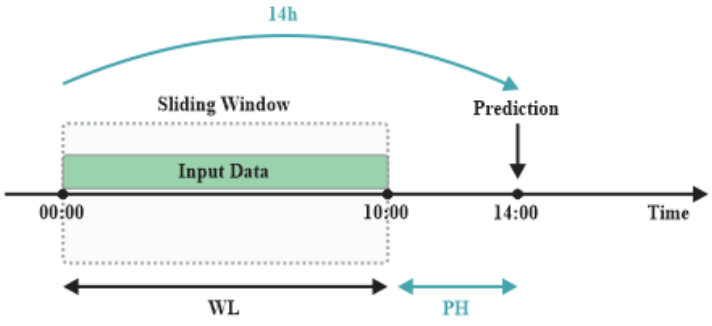


***Fig. 5*** *Implementarea principiului de fereastră glisantă* [13]

K – Fold Based Training funcționează pe două principii fundamentale: împărțirea setului de date în subseturi de dimensiuni egale și executarea proceselor de antrenament. În cazul nostru specific, setul de date include măsurătorile afluentului și efluenților din modelul BSM2 al WWTP. Numărul de pliuri ($K$) a fost ales cu atenție pentru a aloca 70% din setul de date complet pentru antrenarea ANN-urilor, rezervând în același timp 30% pentru scopuri de testare și validare. În cadrul acestor 30%, 15% este desemnat pentru validare, în timp ce restul de 15% servește ca subset de testare. Obiectivul este de a obține modele de predicție distincte prin fiecare proces de antrenament, rezultând un total de modele. Setul de date este utilizat pentru fiecare model, cu subseturi folosite pentru instruire și un subset dedicat testării și validării. În cele din urmă, modelul care arată o precizie superioară de predicție în rândul tuturor proceselor de antrenament este selectat pentru aplicarea finală.

.Studiul de față utilizează trei metrici distincte pentru a evalua performanța modelului. Aceste valori includ eroarea medie pătrată ($RMSE$), eroarea procentuală medie absolută ($MAPE$) și coeficientul de determinare ($R2$). Combinația acestor valori oferă o evaluare amănunțită a eficacității senzorului soft bazat pe rețeaua neuronală artificială (ANN). Rezultatele evaluării indică faptul că modelul ANN prezintă o acuratețe remarcabilă în predicțiile sale (vezi Figura 6 și Figura7).

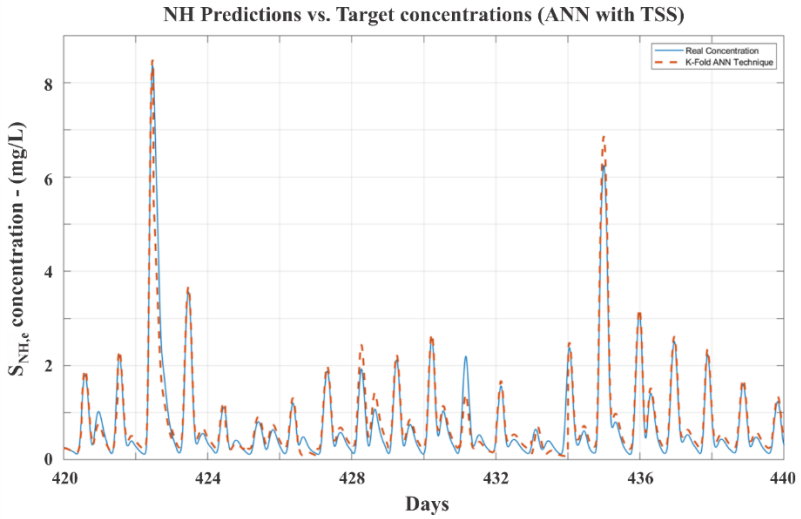


***Fig.6*** *Rezultatele implementării senzorului software, ANN cu TSS* [13]

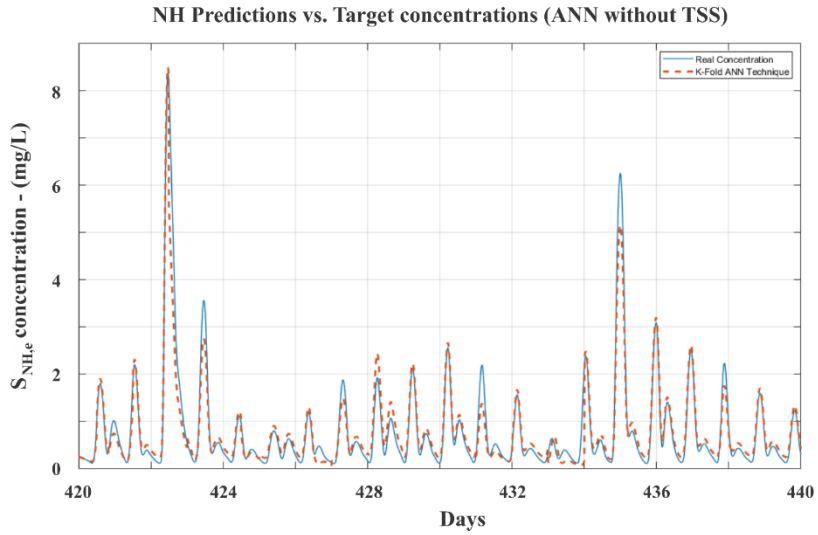


***Fig.7*** *Rezultatele implementării senzorului software, ANN fără TSS* [13]

Dup cum se poate observa rezultatele obținute în urma implementării senzorului software bazat pe rețele neuronale LSTM într-un model de simulare WWTP demonstrează că această abordare tehnologică aduce îmbunătățiri semnificative în monitorizarea și controlul proceselor de tratare a apelor uzate

**Bibliografie**


[1] L. CONDRACHI Eng Scientific supervisor and P. Marian BARBU, "CONTRIBUTIONS REGARDING THE AUTOMATIC CONTROL OF ANAEROBIC DIGESTION PROCESSES," 2022.

[2] "Tipuri de senzori. Senzori smart pentru proiectele tale automatizate IoT – Robofun Blog de Robotică & Electronică." https://blog.robofun.ro/2020/04/13/tipuri-de-senzori-senzori-smart-pentru-proiectele-tale-automatizate-iot/ (accessed Oct. 30, 2023).

[3] "What is a Soft Sensor or Software Sensor? - Körber Pharma." https://www.koerber-pharma.com/blog/what-is-a-soft-sensor-or-software-sensor (accessed Oct. 30, 2023).

[4] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Comput. Chem. Eng.*, vol. 33, no. 4, pp. 795–814, Apr. 2009, doi: 10.1016/J.COMPCHEMENG.2008.12.012.

[5] A. Nair, A. Hykkerud, and H. Ratnaweera, "Estimating Phosphorus and COD Concentrations Using a Hybrid Soft Sensor: A Case Study in a Norwegian Municipal Wastewater Treatment Plant," *Water 2022, Vol. 14, Page 332*, vol. 14, no. 3, p. 332, Jan. 2022, doi: 10.3390/W14030332.

[6] A. V. Luca, M. Simon-Várhelyi, N. B. Mihály, and V. M. Cristea, "Data Driven Detection of Different Dissolved Oxygen Sensor Faults for Improving Operation of the WWTP Control System," *Process. 2021, Vol. 9, Page 1633*, vol. 9, no. 9, p. 1633, Sep. 2021, doi: 10.3390/PR9091633.

[7] M. Alvi, T. French, R. Cardell-Oliver, P. Keymer, and A. Ward, "Cost Effective Soft Sensing for Wastewater Treatment Facilities," *IEEE Access*, vol. 10, pp. 55694–55708, 2022, doi: 10.1109/ACCESS.2022.3177201.

[8] Y. Xie *et al.*, "Enhancing Real-Time Prediction of Effluent Water Quality of Wastewater Treatment Plant Based on Improved Feedforward Neural Network Coupled with Optimization Algorithm," *Water 2022, Vol. 14, Page 1053*, vol. 14, no. 7, p. 1053, Mar. 2022, doi: 10.3390/W14071053.

[9] H. Liu, M. Huang, and C. K. Yoo, "A fuzzy neural network-based soft sensor for modeling nutrient removal mechanism in a full-scale wastewater treatment system," *New pub Balaban*, vol. 51, no. 31–33, pp. 6184–6193, 2013, doi: 10.1080/19443994.2013.780757.

[10] H. Xiao, B. Bai, X. Li, J. Liu, Y. Liu, and D. Huang, "Interval multiple-output soft sensors development with capacity control for wastewater treatment applications: A comparative study," *Chemom. Intell. Lab. Syst.*, vol. 184, pp. 82–93, Jan. 2019, doi: 10.1016/J.CHEMOLAB.2018.11.007.

[11] H. Poutiainen, H. Niska, H. Heinonen-Tanski, and M. Kolehmainen, "Use of sewer on-line total solids data in wastewater treatment plant modelling," *Water Sci. Technol.*, vol. 62, no. 4, pp. 743–750, Aug. 2010, doi: 10.2166/WST.2010.317.

[12] H. Haimi, M. Mulas, F. Corona, and R. Vahala, "Data-derived soft-sensors for biological wastewater treatment plants: An overview," *Environ. Model. Softw.*, vol. 47, pp. 88–107, Sep. 2013, doi: 10.1016/J.ENVSOFT.2013.05.009.

[13] A. E. Țîru, I. Vasiliev, L. Diaconu, R. Vilanova, D. Voipan, and H. Ratnaweera, "Integration of ANN for Accurate Estimation and Control in Wastewater Treatment," *2023 IEEE 28th Int. Conf. Emerg. Technol. Fact. Autom.*, pp. 1–4, Sep. 2023, doi: 10.1109/ETFA54631.2023.10275569.

# Activitate practică 1

# Fișiere de date colectate de la senzori. Încărcare date brute

# Simulare senzor software

## Context

Implementarea unui un senzor software implică o serie de aspecte esențiale precum:

- Colectarea datelor,
- Inspecția datelor,
- Selectarea datelor istorice,
- Preprocesarea datelor,
- Selectarea unui model matematic,
- Instruirea și validarea modelului,
- Integrarea senzorului software
- Monitorizarea și întreținerea senzorului software

Dispozitivele IoT (Internet of Things) generează constant date în mediul nostru. Python este o unealtă puternică pentru analizarea acestor date. În cadrul acestui laborator, veți explora diverse tipuri și module de date Python pentru a învăța cum să citiți, interpretați și transformați datele dintr-un fișier în altul.

## Obiective

În acest prim modul, veți dezvolta o funcție care vă permite să încărcați datele generate de un senzor și stocate în fișiere separate. Aceste date sunt exprimate sub diverse forme numerice și sunt înregistrate în formatul standard CSV.

## Utilizarea limbajului Python pentru analiza și prelucrarea datelor

### 1.1 Noțiuni introductive

Manipularea fișierelor în Python este o operație esențială atunci când lucrați cu date stocate în fișiere. Blocul *with* este un mod recomandat de a deschide și de a lucra cu fișiere în Python, deoarece asigură gestionarea corectă a resurselor și evită scurgerile de memorie.

1.Deschiderea unui fișier:

Pentru a deschide un fișier în Python, utilizați funcția *open()*[1]. Această funcție primește două argumente principale: numele fișierului și modul de deschidere (citire, scriere, etc.).

Exemplu:

```
with open('nume_fisier.txt', 'r') as file:
```

2.Blocul *with*:

Blocul *with* este folosit pentru a crea un context în care resursele (cum ar fi fișierele) sunt gestionate în mod automat. La intrarea în blocul *with*, fișierul este deschis și la ieșirea din bloc, fișierul este închis automat, indiferent dacă execuția programului a fost normală sau a apărut o excepție.

3.Lucrul cu fișierul:

În interiorul blocului *with*, puteți lucra cu fișierul folosind variabila *file*, care este un obiect fișier deschis. Puteți citi, scrie sau modifica conținutul fișierului utilizând metode și operații specifice obiectului fișier.

Exemplu:

```
with open('nume_fisier.txt', 'r') as file:
    data = file.read()
    print(data)
```

4.Închiderea fișierului:

Blocul *with* asigură închiderea automată a fișierului atunci când execuția ajunge la sfârșitul acestuia. Nu mai este nevoie să închideți manual fișierul folosind *file.close()*. Închiderea fișierului este importantă pentru eliberarea resurselor și pentru evitarea scurgerilor de memorie.

### 1.2 Cerințe

1. Setul de date utilizat în acest laborator este dispersat într-un fișier numit *SENSOR_1.CSV*, aflat în folder-ul *datasets*. Aceste date reprezintă informații provenite de la un dispozitiv echipat cu numeroși senzori. Informațiile au fost colectate în mod aleator pe o perioadă de câteva zile și includ măsurători referitoare la temperatură, umiditate, consumul de energie și concentrația de particule din aer într-o anumită zonă. Colectarea datelor a avut loc pe parcursul a 24 de ore.

Scopul exercițiului este să creați o funcție Python care să încarce aceste date din fișierele CSV. Urmați pașii de mai jos pentru a rezolva exercițiul:

1. Creați o funcție numită *load_sensor_data* care primește un singur parametru, *directory_path*, reprezentând calea către directorul care conține fișierele CSV cu datele senzorului.
2. Inițializați o listă goală numită *sensor_data* pentru a stoca datele.
3. Obțineți lista fișierelor CSV din directorul specificat și parcurgeți fiecare fișier.
4. În interiorul buclei pentru fiecare fișier, utilizați un bloc *with* pentru a deschide fișierul CSV.
5. Utilizați *csv.DictReader* pentru a citi datele din fișier și adăugați fiecare înregistrare la lista *sensor_data*.
6. Returnați lista *sensor_data* cu datele încărcate.
7. Exemplu de utilizare: Apelați funcția *load_sensor_data* cu calea către directorul care conține. cu fișierele *CSV* și afișați primele câteva înregistrări pentru a verifica funcționarea corectă.

Notă: Asigurați-vă că specificați corect calea către directorul cu fișierele CSV pentru a obține rezultate corecte.

2. Dezvoltați un senzor software care are capacitatea de a monitoriza nivelul de COD (Chemical Oxygen Demand) într-o stație de tratare a apelor uzate.

1. Senzorul software trebuie să genereze valori aleatoare pentru nivelul COD între 10 și 1000 mg/L. Acesta trebuie să fie simplu de utilizat și să furnizeze date simulate într-un format ușor de citit.
2. Trebuie să afișeze nivelul COD simulat la fiecare 0.1 secunde.
3. Programul trebuie să ruleze într-o buclă infinită pentru a simula măsurători periodice.

## Bibliografie

[1] "3.8.17 Documentation." https://docs.python.org/3.8/ (accessed June 07, 2023).

## Anexa 1

```
import csv

import os


def load_sensor_data(directory_path):

    # Inițializăm o listă goală pentru a stoca datele

    sensor_data = []


    # Obținem lista fișierelor CSV din director

    csv_files = [file for file in os.listdir("C:/Users/User/Desktop/Digiwater") if file.endswith('.csv')]

    for csv_file in csv_files:

        file_path = os.path.join("C:/Users/User/Desktop/Digiwater", csv_file)


        # Deschidem fiecare fișier CSV și citim datele

        with open(file_path, 'r') as file:

            csv_reader = csv.DictReader(file)

            for row in csv_reader:

                sensor_data.append(row)  # Adăugăm fiecare rând la lista de date


    return sensor_data
```

```
# Exemplu de utilizare
if __name__ == "__main__":
    directory_path = "director_cu_fisiere_CSV"  # Schimbați cu calea către directorul cu fișierele CSV
    loaded_data = load_sensor_data(directory_path)

    # Tipărim primele 5 înregistrări pentru exemplificare
    for i, data in enumerate(loaded_data[:10]):
        print(f"Inregistrare {i + 1}: {data}")
```

## Anexa 2

```
import random
import time

def simulate_cod_sensor():
    while True:
        # Generăm o valoare aleatoare pentru COD între 10 și 1000 mg/L
        cod_level = random.uniform(10, 1000)

        # Afișăm nivelul COD simulat
        print(f"Nivel COD curent: {cod_level:.2f} mg/L")

        # Pauza de 0.1 oră între măsurători
        time.sleep(0.1)
        # # Verificăm dacă utilizatorul dorește să oprească simularea
        # user_input = input("Pentru a opri simularea, apăsați 'q' și apoi Enter: ")
        # if user_input == 'q':
        #     break

if __name__ == "__main__":
    simulate_cod_sensor()
```

# Big Data in procesele de tratare a apelor uzate – note de curs

## 1. Big Data – Concepte

În era digitală actuală, volumele de date generate și colectate în fiecare zi sunt uriașe, iar această explozie de date a dat naștere conceptului de "Big Data". Big Data se referă la colecții masive de date, atât structurate, cât și nestructurate, care nu pot fi gestionate și analizate eficient cu instrumente și tehnologii tradiționale. Acest fenomen a schimbat fundamental modul în care organizăm, stocăm, analizăm și înțelegem informațiile în lumea modernă.

Big Data este alimentat de o varietate de surse, inclusiv dispozitive IoT (Internet of Things), social media, sisteme de monitorizare, tehnologii de senzori avansate și multe altele. Aceste date vin sub forma textelor, imaginelor, videoclipurilor, mesajelor audio și multe altele, transformând modul în care gestionăm și utilizăm informațiile în toate domeniile, de la afaceri și medicină la cercetare științifică și guvernare.

Există patru caracteristici principale care definesc Big Data, cunoscute sub acronimul "V": Volum, Viteză, Variație și Valoare.

a) **Volum**: Big Data este caracterizat prin cantități masive de date. Aceste volume de date sunt de obicei mult mai mari decât ce puteau gestiona sistemele tradiționale de baze de date. Astfel, stocarea și gestionarea acestor volume enorme de date necesită infrastructuri și tehnologii specializate.

b) **Viteză**: Datele sunt generate și se înmulțesc cu o viteză impresionantă. Informațiile pot fi actualizate în timp real, iar capacitatea de a răspunde rapid la schimbări și evenimente este crucială. Această caracteristică este adesea legată de IoT, unde datele sunt generate la intervale foarte scurte.

c) **Variație**: Big Data poate avea o varietate semnificativă în ceea ce privește formatele și sursele. Datele pot fi text, imagini, video, date geospațiale și multe altele. Integrarea și analiza acestor date variate reprezintă o provocare importantă.

d) **Valoare**: Principalul scop al Big Data este să extragă valoare din aceste volume mari și diverse de date. Cu ajutorul analizei avansate, se pot descoperi modele, tendințe și insights care pot sprijini luarea de decizii, inovația și eficiența în diverse domenii.

Big Data oferă oportunități semnificative, dar vine și cu provocări majore. Gestionarea datelor la o scară atât de mare necesită investiții semnificative în infrastructură, tehnologie și expertiză. Probleme legate de securitate și confidențialitate sunt, de asemenea, foarte importante, deoarece cu cât mai multe date sunt disponibile, cu atât crește și riscul de expunere a informațiilor sensibile.

În ciuda provocărilor, Big Data reprezintă un motor puternic al inovației și al îmbunătățirii proceselor în toate domeniile. Cu instrumente și tehnologii avansate, Big Data ne oferă ocazia de a înțelege mai bine lumea din jurul nostru, de a dezvolta soluții mai eficiente și de a face prognoze mai precise. Este un domeniu în continuă dezvoltare, care promite să aducă beneficii semnificative societății și economiei globale.

## 2. Tehnologii Big Data

Tehnologiile Big Data reprezintă un set de instrumente, tehnici și tehnologii folosite pentru gestionarea, stocarea, prelucrarea și analiza volumelor masive de date. Aceste tehnologii sunt esențiale în lumea actuală a datelor, deoarece permit organizațiilor să extragă valoare din cantități imense de informații. Iată o prezentare teoretică a principalelor tehnologii Big Data:

a) **Hadoop**

Hadoop Distributed File System (HDFS): Acesta este un sistem de fișiere distribuit care permite stocarea datelor pe cluster. Datele sunt împărțite în blocuri și replicate pe mai multe noduri pentru reziliență.
MapReduce: Este un model de programare și un cadru pentru procesarea datelor mari în paralel. Este folosit pentru a efectua operații de filtrare și agregare pe seturile de date mari, cum ar fi sortarea și filtrarea.

**b) Apache Spark**

Procesare în memorie: Spark utilizează memorie în loc de disc pentru prelucrare, ceea ce îl face mult mai rapid decât MapReduce.
API-uri versatile: Oferă API-uri pentru Python, Scala și Java, facilitând dezvoltatorilor crearea de aplicații complexe de analiză de date.
Suport pentru analiză de fluxuri: Spark Streaming permite prelucrarea și analiza datelor în timp real, ceea ce este util în domenii precum analiza log-urilor sau detectarea fraudelor.

**c) Baze de date NoSQL**

MongoDB: O bază de date orientată pe documente, folosită pentru stocarea datelor nestructurate sau semistructurate, cum ar fi documentele JSON.
Cassandra: O bază de date distribuită și scalabilă, ideală pentru aplicații cu cerințe ridicate de performanță și scalabilitate.
Redis: O bază de date de tip memorie, folosită pentru stocarea datelor cache și pentru prelucrare în timp real.

**d) Baze de date columnare**:
Amazon Redshift: O bază de date columnară de tip data warehouse, ideală pentru analiza datelor în cloud.
Google BigQuery: O bază de date de analiză de date în timp real bazată pe cloud, care permite analiza datelor în timp real.

**e) Data Warehouses**

Teradata: O soluție enterprise pentru gestionarea datelor și analiza business intelligence.
Snowflake: Un data warehouse bazat pe cloud, scalabil și eficient în gestionarea datelor.

**f) Machine Learning și Inteligență Artificială**

Big Data este esențial pentru antrenarea modelelor de învățare automată pe seturi de date mari și diverse. TensorFlow, scikit-learn și PyTorch sunt biblioteci populare folosite pentru dezvoltarea modelelor de ML.

**g) Stream Processing**

Apache Kafka: Un sistem de mesagerie distribuit, care permite prelucrarea fluxurilor de date în timp real.
Apache Flink și Apache Storm: Framework-uri pentru prelucrarea stream-urilor în timp real.

**h) Cloud Computing**

Serviciile cloud oferă resurse scalabile pentru stocarea datelor și analiza acestora.
AWS, Microsoft Azure și Google Cloud Platform oferă servicii Big Data și Machine Learning.

**i) Ecosisteme Big Data**

Ecosistemele oferă o suită de instrumente și servicii pentru gestionarea completă a datelor mari. Cloudera, Hortonworks și MapR sunt exemple notabile.

**j) Data Integration Tools**

Instrumentele ETL precum Apache Nifi, Talend și Informatica permit extragerea, transformarea și încărcarea datelor din surse diverse în mediul Big Data.

Aceste tehnologii sunt utilizate într-o varietate de domenii, de la analiza financiară și medicală până la analiza log-urilor și optimizarea lanțului de aprovizionare. În fiecare domeniu, tehnologiile Big Data sunt esențiale pentru a extrage insights valoroase din datele existente și pentru a lua decizii mai informate. Cu toate acestea, trebuie să se ia în considerare resursele necesare, costurile și cerințele de securitate pentru a implementa cu succes soluții Big Data.

## 3. Utilizarea tehnicilor de Big Data in statiile de tratare ape uzate

Apa este o resursă esențială pentru viața pe Pământ, iar gestionarea și tratarea adecvată a apelor uzate reprezintă o preocupare majoră în societatea modernă. Cu creșterea populației globale și industrializarea continuă, gestionarea și tratarea apelor uzate devin din ce în ce mai complexe. În acest context, Big Data se dovedește a fi o unealtă inovatoare cu un rol semnificativ în procesele de tratare a apelor uzate, aducând o serie de beneficii pentru eficiența și calitatea acestor procese.

Unul dintre cele mai importante aspecte ale utilizării Big Data în tratarea apelor uzate este colectarea datelor. Senzorii avansați și instrumentele de monitorizare colectează date esențiale privind calitatea apei, nivelurile de poluare, cantitățile de apă uzată și multe altele. Aceste date reprezintă baza pentru înțelegerea stării actuale a sistemului de tratare a apelor uzate. Prin colectarea acestor date detaliate, operatorii au la dispoziție informații esențiale pentru luarea deciziilor corecte în timp real. Un alt aspect crucial al Big Data este analiza datelor. Cantitățile masive de date colectate pot fi analizate pentru a identifica tendințele și asemănările care pot fi folosite pentru a îmbunătăți procesele de tratare a apelor uzate. Aceasta oferă oportunitatea de a dezvolta modele predictive și de a anticipa schimbările în calitatea apei sau nivelurile de poluare. Prin înțelegerea mai profundă a datelor, se pot identifica soluții mai eficiente pentru tratarea apei uzate.

Optimizarea proceselor este un alt beneficiu adus de Big Data în tratarea apelor uzate. Analiza datelor poate dezvălui modalități de ajustare a proceselor, cum ar fi dozele de substanțe chimice, controlul debitelor de apă și reducerea consumului de energie. Acest lucru nu numai că contribuie la economisirea resurselor, dar și la reducerea impactului asupra mediului înconjurător. Big Data este, de asemenea, util în detectarea evenimentelor anormale. Datele colectate pot fi folosite pentru a identifica incidente de poluare a apei sau alte probleme neașteptate. Operatorii pot primi alerte în timp real cu privire la aceste situații, permițându-le să reacționeze rapid și să minimizeze impactul asupra mediului.

Prognosticarea necesităților de întreținere este un alt aspect esențial. Prin analiza datelor Big Data, se pot anticipa necesitățile de întreținere a echipamentelor și infrastructurii folosite în tratarea apei uzate. Acest lucru ajută la reducerea timpului de nefuncționare și a costurilor de întreținere, asigurând în același timp funcționarea continuă a sistemelor de tratare a apelor uzate.

Pe lângă toate aceste avantaje, Big Data contribuie la reducerea costurilor în operațiunile de tratare a apelor uzate prin optimizarea proceselor și gestionarea eficientă a resurselor. Aceasta înseamnă o utilizare mai eficientă a resurselor financiare și materiale, ceea ce are un impact pozitiv asupra sustenabilității și a bugetului organizațiilor de tratare a apelor uzate.

Un alt aspect important este respectarea reglementărilor. Big Data poate fi folosit pentru a genera rapoarte precise și pentru a demonstra respectarea normelor de mediu și reglementările guvernamentale privind calitatea apei. Aceasta asigură că sistemele de tratare a apelor uzate funcționează în conformitate cu standardele și că apa eliberată îndeplinește toate cerințele de calitate. În plus, Big Data este esențial în monitorizarea calității apei în timp real. Acest lucru asigură că apa tratată este sigură pentru mediul înconjurător și pentru consumul uman. Datele colectate permit detectarea rapidă a oricăror probleme în calitatea apei, iar operatorii pot lua măsuri imediate pentru a remedia situația. Big Data reprezintă o revoluție în tratarea apelor uzate, aducând o serie de beneficii semnificative pentru eficiența și calitatea acestor procese. Colectarea datelor, analiza lor, optimizarea proceselor, detectarea evenimentelor anormale, prognosticarea necesităților de întreținere, reducerea costurilor, respectarea reglementărilor și monitorizarea calității apei sunt toate aspecte esențiale pe care Big Data le aduce în acest domeniu vital. Acesta oferă operatorilor posibilitatea de a lua decizii mai informate și de a răspunde mai eficient la nevoile de tratare a apelor uzate, contribuind la protejarea mediului și la asigurarea unei aprovizionări adecvate cu apă potabilă pentru societate.

**Exemple de aplicatii Big Data:**

1) **Rețele de Senzori**: Tehnologia senzorilor este la baza aplicațiilor Big Data în WWTP. Acești senzori sunt plasați strategic în toată stația de epurare pentru a monitoriza diferiți parametri, cum ar fi calitatea apei, ratele de curgere, temperatura și altele. Aceștia colectează în mod continuu date și le transmit către un sistem central pentru analiză.

2) **Stocarea și Gestionarea Datelor**: Pentru a gestiona volumele mari de date generate, sunt necesare sisteme solide de stocare și gestionare a datelor. Acest lucru implică, de obicei, utilizarea bazelor de date distribuite și a depozitelor de date care pot stoca și recupera date eficient.

3) **Monitorizare în Timp Real**: Tehnologiile Big Data permit monitorizarea în timp real a proceselor WWTP. Operatorii pot accesa fluxurile de date în timp real și pot primi alerte cu privire la orice devieri sau anomalii, permițând răspunsuri imediate pentru a optimiza procesele de tratare.

4) **Analiza Datelor și Învățarea Automată**: Tehnici avansate de analiză a datelor și de învățare automată sunt utilizate pentru a extrage informații valoroase din datele colectate. Aceste tehnologii pot ajuta la identificarea modelelor, la prevenirea defecțiunilor echipamentelor, la optimizarea dozelor chimice și la îmbunătățirea performanței generale a stației.

5) **Întreținerea Predictivă**: Întreținerea predictivă este crucială pentru WWTP. Utilizând datele istorice și modelele de învățare automată, programările de întreținere pot fi optimizate pentru a reduce timpul de nefuncționare și pentru a minimiza defecțiunile echipamentelor. Acest lucru asigură funcționarea eficientă și rentabilă a stației de tratare.

6) **Vizualizarea Datelor**: Instrumentele de vizualizare a datelor sunt folosite pentru a crea panouri de control și rapoarte ușor de înțeles pentru operatorii stației și factorii de decizie. Reprezentările vizuale ale datelor ajută la identificarea rapidă a tendințelor și problemelor.

7) **Monitorizare și Control la Distanță**: Tehnologiile Big Data permit monitorizarea și controlul la distanță al proceselor WWTP. Acest lucru este deosebit de valoros pentru facilitățile de tratare mari, unde răspunsurile imediate pe teren pot să nu fie posibile.

8) **Integrare cu Tehnologii Inteligente**: Integrarea cu alte tehnologii inteligente, cum ar fi dispozitivele Internet of Things (IoT), poate îmbunătăți și mai mult capacitățile WWTP. Senzorii IoT pot oferi surse suplimentare de date pentru monitorizare și control.

## 4. Analiza Big Data via Python

**Big Data Analytics** este un domeniu care se concentrează pe procesarea și analiza datelor masive pentru a obține insights valoroase. **pache Spark** este un framework open-source de procesare a datelor distribuit, proiectat pentru a gestiona volume mari de date și a oferi performanță în timp real. Spark utilizează concepte precum **RDD-uri** (Resilient Distributed Datasets) și operațiuni de transformare și acțiune pentru a efectua analiza datelor pe un cluster de calcul distribuit. RDD-urile sunt structurile de date fundamentale în Apache Spark. Acestea sunt seturi de date distribuite care pot fi paralelizate și rezistente la fault-uri. RDD-urile pot fi create din date din surse externe și pot fi procesate în mod distribuit pe nodurile clusterului.

**PySpark** este o bibliotecă Python care permite lucrul cu date masive folosind Apache Spark, un framework de procesare a datelor distribuit. Iată câteva concepte teoretice fundamentale în Big Data Analytics cu PySpark. Utilizarea PySpark în analiza datelor din statiile de epurare a apelor uzate (WWTP) poate fi extrem de benefică pentru optimizarea proceselor de tratare a apelor uzate, îmbunătățirea eficienței și reducerea costurilor. Exemple cum PySpark poate fi aplicat în acest context:

1) **Colectarea și Pregătirea Datelor**: PySpark poate fi folosit pentru a colecta și prelucra datele provenite de la senzorii din WWTP, inclusiv date legate de calitatea apei, nivelurile de poluare, debitul de apă și multe altele. Datele pot fi importate din surse diverse, inclusiv fișiere CSV, baze de date, sau direct de la senzori.
2) **Stocarea și Gestionarea Datelor**: Datele pot fi stocate într-un format distribuit, precum RDD-uri sau DataFrames, pentru a permite prelucrarea distribuită cu ajutorul Spark. De asemenea, PySpark poate fi utilizat pentru a integra datele din surse variate și pentru a efectua transformări asupra lor, precum curățarea datelor sau extragerea de informații relevante.
3) **Analiza și Procesarea Datelor**: Cu ajutorul PySpark, puteți efectua analiza avansată a datelor din WWTP. Acest lucru poate include identificarea tendințelor, corelațiilor și modelelor din datele

colectate. De exemplu, puteți analiza cum variază nivelurile de poluare în funcție de sezon sau de condițiile meteorologice.

4) **Optimizarea Proceselor:** PySpark poate fi folosit pentru a optimiza procesele din WWTP. De exemplu, puteți utiliza algoritmii de învățare automată pentru a optimiza dozele de substanțe chimice sau pentru a prognoza nevoile de întreținere a echipamentelor.
5) **Detectarea Evenimentelor Anormale:** PySpark poate ajuta la detectarea evenimentelor anormale sau incidentelor de poluare în WWTP. Puteți crea modele de detecție a anomaliei care să alerteze operatorii în timp real în caz de probleme, astfel încât să poată reacționa rapid.
6) **Vizualizarea Datelor:** PySpark oferă facilități pentru a crea vizualizări relevante și ușor de înțeles ale datelor. Aceste vizualizări pot ajuta la comunicarea eficientă a informațiilor către personalul WWTP și factorii de decizie.
7) **Machine Learning în WWTP:** PySpark oferă un modul MLlib care poate fi utilizat pentru construirea și antrenarea modelelor de învățare automată în WWTP. Aceste modele pot fi folosite pentru a face predicții și pentru a sprijini deciziile legate de tratamentul apei uzate.
8) **Streaming de Date în Timp Real:** Cu Spark Streaming, o componentă a Apache Spark, puteți analiza datele din WWTP în timp real. Acest lucru este util pentru monitorizarea în timp real a calității apei și pentru detectarea evenimentelor neașteptate.

9) **Integrarea cu Tehnologii Inteligente:** Integrarea cu tehnologii inteligente, cum ar fi dispozitivele IoT pentru colectarea datelor în timp real, poate completa analiza datelor din WWTP și permite luarea de decizii mai rapide și mai eficiente.

Utilizarea PySpark în combinație cu datele din WWTP permite îmbunătățirea eficienței proceselor de tratare a apelor uzate, reducerea costurilor de operare și asigurarea respectării reglementărilor privind protecția mediului. Aceasta poate contribui la protejarea mediului înconjurător și la asigurarea unui tratament eficient al apelor uzate.

Documentația oficială PySpark este o resursă excelentă pentru a învăța mai multe despre acest framework de procesare a datelor și pentru a găsi informații detaliate cu privire la funcționalitățile și API-urile sale: Atunci când lucrați cu PySpark, documentația oficială este cea mai bună resursă pentru a găsi răspunsuri la întrebările dvs. și a învăța cum să utilizați eficient acest framework puternic de procesare a datelor:
https://spark.apache.org/docs/latest/api/python/getting_started/index.html

### 4.1 Utilizare Pyspark in Google Colab

Google Colab (Colaboratory) este o platformă gratuită de tip Jupyter Notebook oferită de Google, care rulează în cloud și vă permite să lucrați cu Python și alte librării. De asemenea, puteți utiliza PySpark în Google Colab pentru analiza datelor Big Data. Iată pașii de bază pentru a începe să lucrați cu PySpark în Google Colab.

PySpark, framework-ul Apache Spark pentru programarea în limbajul Python, este conceput pentru a procesa și analiza mari volume de date în mod eficient și scalabil. Elementele de bază din PySpark sunt descrise in continuare. Acestea vă ajută să gestionați, să prelucrați și să analizați mari volume de date în mod eficient și distribuit. PySpark este puternic, versatil și folosit într-o varietate de domenii, de la analiza datelor la învățarea automată și analiza datelor în timp real.

**SparkSession**: Acesta este punctul de intrare principal în PySpark și este utilizat pentru a crea o sesiune Spark. Prin intermediul sesiunii Spark, puteți configura setările, încărca datele, crea DataFrames și RDD-uri și interacționa cu clusterele Spark.

**DataFrame**: DataFrame este o structură de date tabulară, similară cu un tabel dintr-o bază de date relațională. Acesta este un concept fundamental în PySpark și este utilizat pentru a manipula datele în mod eficient. Datele pot fi încărcate într-un DataFrame din diverse surse, cum ar fi fișiere CSV, baze de date sau alte surse de date structurate.

**Resilient Distributed Dataset (RDD):** RDD este un alt concept de bază în Apache Spark și reprezintă o colecție distribuită și imutabilă de date. RDD-urile sunt folosite pentru prelucrarea datelor în mod distribuit pe clustere Spark și sunt utile în scenarii mai avansate.

**Transformări și Acțiuni**: PySpark oferă o serie de transformări (transformations) și acțiuni (actions) pe DataFrames și RDD-uri. Transformările sunt operații lenese care transformă un DataFrame existent într-unul nou, în timp ce acțiunile declanșează efectiv operații și întorc rezultate. Exemple de transformări includ select(), filter(), groupBy(), în timp ce exemple de acțiuni includ show(), count(), saveAsTable().

**Partiții**: Datele dintr-un DataFrame sau RDD sunt împărțite în mai multe partiții pentru a permite procesarea distribuită pe mai multe noduri. Partițiile sunt unități de bază pentru distribuție și prelucrare paralelă a datelor.

**Clustere Spark:** Pentru a utiliza PySpark în mod eficient, este necesară o infrastructură de cluster. Un cluster Spark este compus dintr-un grup de noduri care lucrează împreună pentru a procesa și analiza datele. Clusterele pot fi gestionate folosind soluții precum Apache Hadoop, YARN sau în medii cloud, cum ar fi AWS EMR sau Google Dataproc.

**Stocarea și Paralelismul**: PySpark oferă opțiuni pentru a stoca date în mod distribuit și a asigura paralelismul în prelucrarea datelor. Acest lucru asigură faptul că datele pot fi procesate eficient pe clusterul Spark.

**Librării Python:** puteți folosi librăriile Python, cum ar fi NumPy, Pandas, Matplotlib și altele, pentru analiza datelor și vizualizare în cadrul mediului PySpark.

**Optimizare și Tuning:** Apache Spark oferă capacități avansate de optimizare, precum stocarea în memorie și planificare eficientă a operațiilor, pentru a accelera prelucrarea datelor.

**Machine Learning cu MLlib**: Spark MLlib este o bibliotecă pentru învățarea automată și oferă algoritmi de învățare automată pentru clasificare, regresie, clusterizare și altele.

Mai jos, sunt pasii de instalare si utilizare PySpark in Google Colab:

**Pas 1**: Google Colab vine deja cu Python instalat, dar trebuie să instalați PySpark și să setați o sesiune Spark. Folosiți următorul cod pentru a face acest lucru într-o celulă de cod în Colab:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-
bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pas 2**: Să setați variabilele de mediu pentru Java și Spark. Va trebui, de asemenea, să ajustați versiunea Spark în calea specificată pentru a se potrivi cu versiunea instalată:

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"
```

**Pas 3**: Adăugați directoarele Spark și PySpark la calea sistemului:

```
import findspark
findspark.init()
```

**Pas 4**: Importare PySpark si creare sesiune SparkSession. Acest exemplu demonstrează cum să configurați Spark pentru a funcționa în mod local pe Google Colab.

```
from pyspark.sql import SparkSession

# Creati o sesiune Spark
spark = SparkSession.builder.appName("example").getOrCreate()
```

**Pas 5**: Lucrul cu PySpark. După ce ați configurat o sesiune Spark, puteți începe să lucrați cu PySpark în celelalte celule din notebook. Puteți crea RDD-uri sau DataFrames și să efectuați operații de analiză de date folosind biblioteca PySpark. Puteți încărca datele în Google Colab din diverse surse, cum ar fi Google Drive, Dropbox sau direct dintr-un URL. După ce ați încărcat datele, puteți utiliza PySpark pentru a le prelucra.

```
# Creati un DataFrame
data = [("Alice", 34), ("Bob", 45), ("Catherine", 29)]
columns = ["Name", "Age"]
df = spark.createDataFrame(data, columns)
```

```
# Afisati DataFrame
df.show()

# Realizati o operatie de analiza simpla
df.select("Name", "Age").filter(df.Age > 30).show()
```

**Pas 6**: Exportul Rezultatelor. După ce ați efectuat analiza datelor și ați obținut rezultatele dorite, puteți exporta rezultatele în fișiere sau puteți partaja notebook-ul cu colegii sau alții.

**Pas 7**: Când ați terminat, nu uitați să opriți sesiunea Spark pentru a elibera resursele:

```
spark.stop()
```

Analiza datelor de apă uzată poate implica analiza și vizualizarea datelor referitoare la calitatea și cantitatea apei uzate. Vom utiliza limbajul Python și unele biblioteci comune de știința datelor, cum ar fi Pandas, Matplotlib și Seaborn, pentru a explora și vizualiza datele de apă uzată. Presupunem că aveți un set de date care conține informații despre calitatea și cantitatea apei uzate în decursul timpului. Iată un exemplu simplificat de modul în care puteți aborda analiza:

În primul rând, trebuie să încărcați datele de apă uzată dintr-un fișier CSV sau Excel într-un DataFrame Pandas. Pentru acest exemplu, să presupunem că aveți un fișier CSV numit "date_apa_uzata.csv".

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("AnalizaApaUzata").getOrCreate()

# Încărcați datele într-un DataFrame PySpark
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)
```

Explorați setul de date pentru a înțelege structura și conținutul acestuia. Puteți verifica primele câteva rânduri ale setului de date, statisticile de rezumat și tipurile de date.

```
# Afișați primele câteva rânduri ale DataFrame-ului
data_apa_uzata_df.show()

# Afișați statistici de rezumat
data_apa_uzata_df.describe().show()

# Verificați tipurile de date
data_apa_uzata_df.printSchema()
plt.ylabel("Cantitate (m³/zi)")
plt.show()
```

Vizualizați datele folosind PySpark cu Matplotlib și Seaborn:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Convertiți DataFrame-ul PySpark într-un DataFrame Pandas pentru
vizualizare
data_apa_uzata_pandas_df = data_apa_uzata_df.toPandas()

# Creați un grafic de serie temporală
plt.figure(figsize=(12, 6))
sns.lineplot(x="Data", y="CantitateApaUzata",
data=data_apa_uzata_pandas_df)
plt.title("Cantitatea de Apă Uzată în Timp")
plt.xlabel("Data")
plt.ylabel("Cantitate (m³/zi)")
plt.show()

# Creați un histogramă pentru calitatea apei uzate
plt.figure(figsize=(8, 6))
sns.histplot(data_apa_uzata_pandas_df["CalitateApaUzata"], bins=20,
kde=True)
plt.title("Distribuția Calității Apei Uzate")
plt.xlabel("Calitate")
plt.ylabel("Contor")
plt.show()
```

Realizați analiza datelor și obțineți înțelegerea utilizând capacitățile PySpark pentru transformarea datelor, filtrarea și agregarea acestora. Faceți recomandări pe baza analizei datelor. Nu uitați să opriți sesiunea PySpark când ați terminat:

```
spark.stop()
```

Acest exemplu vă permite să efectuați analiza datelor de apă uzată folosind PySpark pentru prelucrarea datelor distribuite. Un studiu de caz real privind analiza datelor de apă uzată poate implica mai mult preprocesarea datelor, modelarea și cunoștințe specifice domeniului. În plus, puteți extinde acest exemplu prin incorporarea modelelor de învățare automată pentru analiza predictivă sau detectarea anomalilor, dacă setul dvs. de date permite astfel de analize.

## 4.2. Exemple de operatii de prelucrare asupra datelor

Cu PySpark, puteți efectua o serie de operații de prelucrare și analiză a datelor pe datele colectate de la stațiile de epurare a apelor uzate. Iată câteva operații tipice pe care le puteți realiza:

- Preprocesare a datelor: Curățare și transformare a datelor pentru eliminarea datelor lipsă sau incorecte.

- Standardizarea și normalizarea datelor, în cazul în care datele sunt colectate din surse variate sau au unități diferite.
- Filtrare și selecție de date: Filtrarea datelor pentru a extrage subconjuncturi relevante pentru analiză.
- Selecția de date în funcție de anumite criterii, cum ar fi intervalul de timp sau parametrii specifici de calitate a apei.
- Agregare de date: Calculul statisticilor de bază, cum ar fi media, deviația standard, minimul și maximul pentru diferiți parametri de calitate a apei.
- Agregarea datelor în funcție de intervale de timp sau locații specifice pentru a obține perspective globale sau regionale asupra calității apei uzate.
- Analiză a seriei temporale: Realizarea analizei de serie temporală pentru a identifica tendințe, modele sezoniere și variații ale parametrilor apei uzate.
- Detectarea de anomalii în datele de serie temporală care ar putea indica probleme în funcționarea stației de epurare.
- Analiză de corelație: Evaluarea corelațiilor între diferiți parametri ai apei uzate pentru a înțelege relațiile dintre aceștia.
- Determinarea impactului unor factori asupra calității apei uzate.
- Analiză de regresie: Utilizarea de modele de regresie pentru a prognoza valori viitoare ale parametrilor apei uzate pe baza datelor istorice.
- Vizualizare și raportare: Crearea de grafice, diagrame și rapoarte pentru a comunica rezultatele analizei către părțile interesate sau pentru luarea deciziilor.
- Analiza de costuri și eficiență: Evaluarea costurilor și eficienței operațiunilor la stația de epurare a apelor uzate.
- Identificarea posibilelor îmbunătățiri pentru reducerea costurilor sau optimizarea proceselor.
- Machine Learning și Inteligență Artificială: Implementarea de modele de învățare automată pentru a face previziuni, de exemplu, pentru a anticipa problemele înainte de a se întâmpla.
- Implementarea de algoritmi AI pentru optimizarea proceselor de control și monitorizare.

**Exemplu 1**: Să considerăm un exemplu simplificat în Python, folosind PySpark, pentru a efectua unele operațiuni comune pe datele de apă uzată de la o stație de epurare a apelor uzate. În acest exemplu, presupunem că aveți un mediu PySpark configurat și că aveți un set de date într-un fișier CSV numit "date_apa_uzata.csv".

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg, stddev, max

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluApaUzata").getOrCreate()

# Încărcați datele de apă uzată într-un DataFrame PySpark
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Afișați primele câteva rânduri ale DataFrame-ului
data_apa_uzata_df.show()
```

```
# Calculați și afișați media și deviația standard a coloanei
"CantitateApaUzata"
medie_cantitate_apa =
data_apa_uzata_df.select(avg(col("CantitateApaUzata"))).first()[0]
deviatie_standard =
data_apa_uzata_df.select(stddev(col("CantitateApaUzata"))).first()[0]
print(f"Medie Cantitate Apă Uzată: {medie_cantitate_apa}")
print(f"Deviație Standard a Cantității de Apă Uzată: {deviatie_standard}")

# Găsiți data când a fost înregistrată calitatea maximă a apei uzate
data_max_calitate =
data_apa_uzata_df.select("Data").filter(data_apa_uzata_df.CalitateApaUzata
== max(data_apa_uzata_df.CalitateApaUzata)).first()
print(f"Data cu Calitatea Maximă a Apei Uzate: {data_max_calitate.Data}")

# Grupați datele pe lună și calculați media Cantității de Apă Uzată pentru
fiecare lună
data_apa_uzata_df = data_apa_uzata_df.withColumn("Luna",
data_apa_uzata_df["Data"].substr(1, 7))
medie_lunara_cantitate_apa =
data_apa_uzata_df.groupBy("Luna").agg(avg("CantitateApaUzata").alias("Medi
eCantitateApa"))
medie_lunara_cantitate_apa.show()

# Opriți sesiunea Spark
spark.stop()
```

În exemplul de mai sus, pasii sunt:

- incarcarea datele de apă uzată într-un DataFrame PySpark.
- calcularea și afisarea mediei și deviației standard a coloanei "CantitateApaUzata".
- cautarea datei la care a fost înregistrată calitatea maximă a "CalitateApaUzata".
- gruparea datelor pe lună și calcularea mediei "CantitateApaUzata" pentru fiecare lună.

Acesta este un exemplu simplu care ilustrează cum puteți utiliza PySpark pentru a efectua operații pe datele de apă uzată. Puteți extinde acest exemplu cu analize mai complexe, vizualizare sau tehnici de învățare automată în funcție de cerințele specifice.

**Exemplu 2:** În acest exemplu, ne concentrăm pe transformarea datelor și agregarea acestora.

```
# Import PySpark modules
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

# Crearea unei sesiuni Spark
spark = SparkSession.builder.appName("ExempluApaUzata2").getOrCreate()
```

```
# Încărcarea datelor de apă uzată într-un DataFrame PySpark (înlocuiți
"date_apa_uzata.csv" cu fișierul dvs. de date)
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Crearea unei noi coloane "QualityCategory" bazată pe "CalitateApaUzata"
data_apa_uzata_df = data_apa_uzata_df.withColumn("QualityCategory",
when(col("CalitateApaUzata") >= 80, "Calitate Înaltă")
                    .when(col("CalitateApaUzata") >= 50, "Calitate Medie")
                    .otherwise("Calitate Scăzută"))

# Calculul cantității totale de apă uzată pe lună
data_apa_uzata_df = data_apa_uzata_df.withColumn("Luna",
data_apa_uzata_df["Data"].substr(1, 7))
cantitate_totala_lunara =
data_apa_uzata_df.groupBy("Luna").sum("CantitateApaUzata")

# Identificarea lunii cu cea mai mare cantitate totală de apă uzată
luna_maxima_cantitate =
cantitate_totala_lunara.orderBy("sum(CantitateApaUzata)",
ascending=False).first()
print(f"Luna cu Cea Mai Mare Cantitate Totală:
{luna_maxima_cantitate['Luna']}")

# Salvarea DataFrame-ului transformat într-un nou fișier CSV
data_apa_uzata_df.write.csv("date_apa_uzata_transformate.csv",
header=True)

# Oprește sesiunea Spark
spark.stop()
```

În exemplul de mai sus, pasii sunt:

- crearea unei coloane noi "CategorieCalitate" pe baza coloanei "CalitateApaUzata," categorizând calitatea ca "Calitate înaltă," "Calitate medie" sau "Calitate scăzută."
- calcularea cantitatii totală de apă uzată pe lună și identificarea lunii cu cea mai mare cantitate totală.
- salvarea DataFrame-ului transformat într-un nou fișier CSV, care include coloana "CategorieCalitate."

Acest exemplu demonstrează cum să efectuați transformări ale datelor, agregări și să salvați rezultatele într-un nou fișier folosind PySpark. Puteți adapta și extinde aceste operațiuni pentru a se potrivi nevoilor dvs. specifice de analiză a datelor de apă uzată.

**Exemplu 3**: acest exemplu ilustrează cum puteți utiliza PySpark pentru sarcini de analiză a datelor de apă uzată mai avansate, inclusiv integrarea datelor, învățarea automată și modelarea predictivă. Asigurați-vă

că înlocuiți "date_apa_uzata.csv" și "date_meteorologice.csv" cu numele fișierelor reale ale seturilor de date.

```
# Import PySpark modules
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression

# Crearea unei sesiuni Spark
spark = SparkSession.builder.appName("ExempluApaUzata3").getOrCreate()

# Încărcarea datelor de apă uzată într-un DataFrame PySpark
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Încărcarea datelor meteorologice într-un DataFrame PySpark (înlocuiți cu
fișierul dvs. de date meteorologice)
date_meteorologice_df = spark.read.csv("date_meteorologice.csv",
header=True, inferSchema=True)

# Unirea celor două DataFrame-uri folosind o coloană comună (de exemplu,
"Data")
data_combinata_df = data_apa_uzata_df.join(date_meteorologice_df, "Data",
"inner")

# Realizarea pregătirii datelor, inclusiv vectorizarea caracteristicilor
coloane_caracteristici = ["CalitateApaUzata", "Temperatura",
"Precipitații"]
asamblor = VectorAssembler(inputCols=coloane_caracteristici,
outputCol="caracteristici")
date_pregatite = asamblor.transform(data_combinata_df)

# Definirea unui model de Regresie Liniară
rl = LinearRegression(featuresCol="caracteristici",
labelCol="CantitateApaUzata")

# Împărțirea datelor în seturi de antrenament și testare
date_antrenament, date_testare = date_pregatite.randomSplit([0.7, 0.3])

# Antrenarea modelului de Regresie Liniară
model_rl = rl.fit(date_antrenament)

# Realizarea predicțiilor pe datele de testare
previziuni = model_rl.transform(date_testare)
```

```
# Evaluarea performanței modelului
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(labelCol="CantitateApaUzata",
predictionCol="previziune", metricName="rmse")
rmse = evaluator.evaluate(previziuni)
print(f"Eroare Medie Patratică (RMSE): {rmse}")

# Oprește sesiunea Spark
spark.stop()
```

În exemplul de mai sus, pasii sunt:

- incarcarea a două seturi de date, unul conținând date de apă uzată și celălalt conținând date meteorologice.
- concatenarea acestor seturi de date folosind o coloană comună, în acest caz, coloana "Data."
- pregătirea datelor, inclusiv vectorizarea caracteristicilor, pentru a pregăti datele pentru învățarea automată.
- definirea unui model de Regresie Liniară sa îl antrenăm pe setul de date combinat.
- divizarea datelor în seturi de antrenament și testare și evaluăm performanța modelului folosind Root Mean Squared Error (RMSE).

# Bibliografie


1. Al-Jasser, A. O., & Ghani, M. (2015). Big Data and Water: A Review of the State-of-the-Art and Future Opportunities. Journal of Hydroinformatics, 17(1), 16-32.
2. Palani, S., Goyal, R., & Bhatt, V. S. (2019). Big Data Analytics for Efficient Wastewater Treatment in Smart Cities. Journal of Water Process Engineering, 30, 100622.
3. Kaushal, R. K., & Patel, P. (2016). Data-Driven Modeling and Optimization of Wastewater Treatment Plants. Water Research, 88, 351-366.
4. Qasim, S. R., Hu, Y., & Chiang, P. C. (2016). Wastewater Treatment Plants: Planning, Design, and Operation. CRC Press.
5. Chen, Y., & Wang, X. (2019). Big Data Analytics in Water Resources Engineering: A Survey. CRC Press.
6. De Moura, L. J., Silva, F. M., & Luvizotto, E. (2019). Data Analytics and PySpark for Enhancing Wastewater Treatment Plants Efficiency. Chemical Engineering Transactions, 74, 1813-1818.
7. Fernandez, M. S., Torrens, R., & Ayesa, E. (2020). Leveraging PySpark for Real-Time Data Analysis and Decision Support in Wastewater Treatment Plants. Procedia Computer Science, 173, 123-130.
8. Sun, Z., et al. (2019). Real-Time Monitoring of a Municipal Wastewater Treatment Plant Using PySpark for Data Analysis. Water Research, 155, 143-153.
9. Smith, R., & Patel, A. (2020). PySpark-Based Data Integration and Analysis for Enhanced WWTP Efficiency. Journal of Environmental Engineering, 146(5), 04020010.
10. Chen, X., et al. (2018). Applying PySpark for Predictive Maintenance in Wastewater Treatment Facilities. IFAC-PapersOnLine, 51(12), 1445-1450.
11. Global Water Intelligence. (Accessed November 2, 2023). https://www.globalwaterintel.com/
12. Water Online. (Accessed November 2, 2023). https://www.wateronline.com/
13. International Water Association. (Accessed November 2, 2023). https://www.iwa-network.org/

# Laborator – Aplicatia 1

**Obiective:**

- Cunoasterea tehnicilor de analiză big data care se pot aplica pe seturile de date experimentale colectate din stațiile de tratare ape uzate;
- Însușirea termenilor specifici și înțelegerea importanței monitorizării online ca aspect important al digitalizarii stațiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În era digitală, analiza datelor a devenit o unealtă crucială pentru optimizarea proceselor în diferite domenii, inclusiv gestionarea resurselor de apă uzată. Acest referat de laborator se concentrează pe utilizarea PySpark, un framework de procesare a datelor în limbajul Python, pentru a analiza date legate de calitatea apei uzate și factorii meteorologici asociate cu aceasta. Scopul este de a dezvolta un model de regresie liniară care să poată prezice cantitatea de apă uzată în funcție de variabilele observate.

**Calitatea Apei Uzate:** Datele de calitate a apei uzate pot include parametri precum concentrația de poluanți (de exemplu, BOD, COD, azot, fosfor, metale grele), indicatori de pH, soliditate, temperatura apei și alți parametri relevanți. Aceste date pot fi colectate de la stațiile de tratare a apei uzate sau laboratoare specializate.

**Date Meteorologice:** Datele meteorologice pot include informații precum temperatura aerului, precipitațiile, umiditatea, direcția și viteza vântului, presiunea atmosferică și altele. Aceste date pot fi colectate de la stații meteorologice locale sau din surse meteorologice oficiale.

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatia1").getOrCreate()
```

**Pasul 3: Incarcați Datele**

Încărcăm datele de calitate a apei uzate dintr-un fișier CSV și datele meteorologice din alt fișier CSV. Aceste date vor fi stocate în DataFrame-uri PySpark

```
# Încărcați datele de apă uzată într-un DataFrame PySpark
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Încărcați datele meteorologice într-un DataFrame PySpark (înlocuiți cu
fișierul dvs. de date meteorologice)
date_meteorologice_df = spark.read.csv("date_meteorologice.csv", header=True,
inferSchema=True)
```

**Pasul 4: Concatenați datele**

Unim cele două DataFrame-uri folosind o coloană comună, cum ar fi "Data". Aceasta va permite să analizăm datele din cele două surse în mod corespunzător.

```
# Uniți cele două DataFrame-uri folosind o coloană comună ("Data") folosind
metoda "join"
data_combinata_df = data_apa_uzata_df.join(date_meteorologice_df, "Data",
"inner")
```

**Pasul 5: Pregătiți Datele**

Realizăm pregătirea datelor, inclusiv vectorizarea caracteristicilor de interes. În acest caz, coloanele de interes includ "CalitateApaUzata", "Temperatura" și "Precipitații".

```
# Definim coloanele de caracteristici
coloane_caracteristici = ["CalitateApaUzata", "Temperatura", "Precipitații"]

# Creăm un asamblor pentru vectorizarea caracteristicilor
asamblor = VectorAssembler(inputCols=coloane_caracteristici,
outputCol="caracteristici")

# Aplicăm asamblorul pentru a transforma datele
date_pregatite = asamblor.transform(data_combinata_df)
```

**Pasul 6: Implementați modelul de Regresie Liniară**

Definim un model de regresie liniară care va fi antrenat pe datele pregătite. Acest model va ajuta la prezicerea cantității de apă uzată în funcție de caracteristicile observate.

```
from pyspark.ml.regression import LinearRegression

# Definim modelul de Regresie Liniară
rl = LinearRegression(featuresCol="caracteristici",
labelCol="CantitateApaUzata")
```

**Pasul 7: Divizați datele și antrenați modelul**

Divizăm datele în seturi de antrenament și testare și apoi antrenăm modelul de regresie liniară.

```
# Divizăm datele în seturi de antrenament și testare
date_antrenament, date_testare = date_pregatite.randomSplit([0.7, 0.3])

# Antrenăm modelul de Regresie Liniară
model_rl = rl.fit(date_antrenament)
```

**Pasul 8: Realizați predicții și evaluați performanța**

Folosim modelul antrenat pentru a realiza predicții pe datele de testare și apoi evaluăm performanța modelului folosind metrica RMSE (Rădăcina Eroare Medie Patratică).

```
# Evaluați performanța modelului
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(labelCol="CantitateApaUzata",
predictionCol="previziune", metricName="rmse")
rmse = evaluator.evaluate(previziuni)
print(f"Rădăcina Eroare Medie Patratică (RMSE): {rmse}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 9: Interpretați rezultatele și explicați operațiile realizate.**

# Laborator – Aplicatia 2

**Obiective:**

- Cunoasterea tehnicilor de analiză big data care se pot aplica pe seturile de date experimentale colectate din stațiile de tratare ape uzate;
- Însușirea termenilor specifici și înțelegerea importanței monitorizării online ca aspect important al digitalizarii stațiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În acest laborator, vom utiliza PySpark pentru a analiza date referitoare la calitatea apei uzate. Scopul este de a dezvolta un model de clasificare pentru a determina dacă apa uzată este de "Calitate Bună" sau "Calitate Scăzută" pe baza caracteristicilor observate.

Un exemplu de sursă pentru un set de date legat de calitatea apei uzate și caracteristicile asociate:

**UCI Machine Learning Repository - Water Quality Data Set:**

Acest set de date conține informații despre calitatea apei uzate și caracteristicile apei, cum ar fi temperatură, pH, soliditate și altele. Puteți descărca setul de date de la UCI Machine Learning Repository folosind următorul link: https://www.kaggle.com/datasets/adityakadiwal/water-potability

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatie2").getOrCreate()
```

**Pasul 3: Incărcați Datele**

Încărcați datele de calitate a apei uzate și caracteristicile asociate în DataFrame-uri PySpark.

```
# Încărcați datele de calitate a apei uzate într-un DataFrame PySpark
wastewater_df = spark.read.csv("wastewater_data.csv", header=True,
inferSchema=True)

# Încărcați caracteristicile asociate cu apa uzată (exemplu: temperatură, pH,
conductivitate)
features_df = spark.read.csv("water_features.csv", header=True,
inferSchema=True)
```

**Pasul 4: Pregătiți datele**

Realizați pregătirea datelor, inclusiv prelucrarea caracteristicilor și etichetarea datelor pentru clasificare.

```
from pyspark.ml.feature import VectorAssembler

# Definiți coloanele de caracteristici pentru vectorizare
feature_columns = ["Temperature", "pH", "Conductivity"]

# Creează un asamblor pentru a combina caracteristicile într-o singură
coloană
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")

# Aplică asamblorul pentru caracteristici
data = assembler.transform(features_df)

# Etichetați datele: "Good" sau "Low" calitate a apei uzate
from pyspark.sql.functions import when
wastewater_df = wastewater_df.withColumn("QualityCategory",
    when(wastewater_df["WastewaterQuality"] >= 80, "Good").otherwise("Low")
)
```

**Pasul 5: Implementați modelul de clasificare**

Definiți un model de clasificare, cum ar fi Random Forest, și antrenați-l pentru a clasifica apa uzată ca "Calitate Bună" sau "Calitate Scăzută".

```
from pyspark.ml.classification import RandomForestClassifier
```

```
# Definiți modelul de clasificare
rf = RandomForestClassifier(featuresCol="features",
labelCol="QualityCategory", numTrees=10)

# Divizați datele în seturi de antrenament și testare
train_data, test_data = data.randomSplit([0.7, 0.3])

# Antrenați modelul pe datele de antrenament
model = rf.fit(train_data)
```

**Pasul 6: Evaluați performanța modelului**

Evaluează performanța modelului de clasificare folosind metrici precum precizia, recuperarea și F1-score.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Realizați predicții pe datele de testare
predictions = model.transform(test_data)

# Evaluator pentru metrici de clasificare
evaluator = MulticlassClassificationEvaluator(labelCol="QualityCategory",
metricName="accuracy")

# Calculați precizia modelului
accuracy = evaluator.evaluate(predictions)
print(f"Precizia modelului: {accuracy}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 7: Interpretați rezultatele și explicați operațiile realizate.**

# Laborator – Aplicatia 3

**Obiective:**

- Cunoasterea tehnicilor de analiza big data care se pot aplica pe seturile de date experiementale colectate din statiile de tratare ape uzate;
- Însușirea termenilor specifici si intelegerea importantei monitorizarii online ca aspect important al digitalizarii statiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În acest laborator, ne vom concentra pe utilizarea PySpark pentru analiza datelor de calitate a apei uzate și pentru dezvoltarea unui model de regresie liniară care să poată prezice consumul de apă uzată pe baza caracteristicilor observate.

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatie3").getOrCreate()
```

**Pasul 3: Încărcați Datele**

Încărcați datele de calitate a apei uzate și caracteristicile asociate în DataFrame-uri PySpark.

```
from pyspark.sql import SparkSession
# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluApaUzata2").getOrCreate()

# Încărcați datele de calitate a apei uzate într-un DataFrame PySpark
wastewater_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Încărcați caracteristicile asociate cu apa uzată (exemplu: temperatură, pH,
conductivitate)
features_df = spark.read.csv("date_caracteristici_apa.csv", header=True,
inferSchema=True)
```

**Pasul 4: Pregătiți datele**

Realizați pregătirea datelor, inclusiv prelucrarea caracteristicilor și etichetarea datelor pentru regresie.

```
from pyspark.ml.feature import VectorAssembler

# Definiți coloanele de caracteristici pentru vectorizare
feature_columns = ["Temperature", "pH", "Conductivity"]

# Creează un asamblor pentru a combina caracteristicile într-o singură
coloană
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")

# Aplică asamblorul pentru caracteristici
data = assembler.transform(features_df)

# Definiți coloana de etichetă (cantitatea de apă uzată)
label_column = "WastewaterFlow"
```

**Pasul 5: Implementați modelul de regresie Liniară**

Definiți un model de regresie liniară și antrenați-l pentru a prezice cantitatea de apă uzată.

```
from pyspark.ml.regression import LinearRegression

# Definiți modelul de regresie liniară
lr = LinearRegression(featuresCol="features", labelCol=label_column)

# Divizați datele în seturi de antrenament și testare
train_data, test_data = data.randomSplit([0.7, 0.3])
```

```
# Antrenați modelul de regresie liniară
model = lr.fit(train_data)
```

**Pasul 6: Realizați predicții și evaluați performanța**

Folosiți modelul antrenat pentru a face predicții pe datele de testare și evaluați performanța modelului.

```
# Realizați predicții pe datele de testare
predictions = model.transform(test_data)

# Evaluator pentru metricile de regresie
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(labelCol=label_column,
predictionCol="prediction", metricName="rmse")

# Calculați eroarea medie pătratică
rmse = evaluator.evaluate(predictions)

print(f"Rădăcina Eroare Medie Pătratică (RMSE): {rmse}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 7: Interpretați rezultatele și explicați operațiile realizate.**

**Conducerea automată a proceselor**
**- note de curs -**

# 1. Introducere

## Ce este conducerea automată a proceselor?

Ingineria sistemelor este disciplina care studiază sistemele proiectate de om cu scopul de a manipula răspunsul acestora. Ingineria sistemelor este o știință interdisciplinară care joacă un rol important în multe dintre științele inginerești precum ingineria electrică, ingineria mecanică, ingineria chimică, biotehnologie etc. Există o bază teoretică care poate fi aplicată tuturor acestor sisteme în ciuda diferențelor substanțiale dintre caracteristicile lor fizice.

Astfel, conducerea automată a proceselor este o ramură a ingineriei sistemelor care se ocupă cu controlul instalațiilor de producție din industria chimică, petrochimică, alimentară, biotehnologie etc. Conducerea automată a proceselor are un rol important în operarea corespunzătoare a instalațiilor în ceea ce privește siguranța, calitatea produselor și profitabilitatea. Chiar dacă procesele biotehnologice (e.g. tratarea biologică a apelor uzate) au caracteristici fizice diferite de ale unui robot, unei drone sau unei rachete, principiile teoriei sistemelor care stau la baza acestora sunt aceleași.

Teoria sistemelor este prezentă în viața de zi cu zi. Mașinile, frigiderele, mașinile de spălat, clădirile etc. au numeroase sisteme de conducere automată. Ceea ce este impresionant este faptul că aceste sisteme de conducere automată funcționează atât de eficient că abia dacă observăm existența lor. Ele sunt fiabile și ne fac viața mai bună și mai sigură, iar uneori sunt chiar ieftine.

Volumul de cunoștințe care a fost generat în ultimele decenii este în domeniul ingineriei sistemelor este considerabil, iar folosirea acestor cunoștințe pentru proiectarea și operarea sistemelor de conducere automată este vitală pentru dezvoltarea bio(proceselor).

## Sisteme de control în buclă închisă. Terminologie.

În această secțiune vor fi prezentate conceptele esențiale și terminologia utilizată la controlul în buclă închisă a unui proces (Kravaris & Kookos, 2021).

Vom lua drept exemplu sistemul de control "primitiv" al unui tanc de stocare lichid, prezentat în figura 1.1. Astfel, un debit de lichid alimentează tancul de stocare (*proces*), iar un operator (*regulator/controler*) încearcă să mențină nivelul din tanc (*variabilă măsurată/controlată*) la o valoarea dorită (*referință*), folosind o procedură logică (*algoritm de control*) bazată pe experiența sa. Mecanismul prin care operatorul poate menține nivelul tancului de stocare este deschiderea sau închiderea unei valve (*element final de control*) care ajustează debitul de alimentare (*variabilă manipulată*).

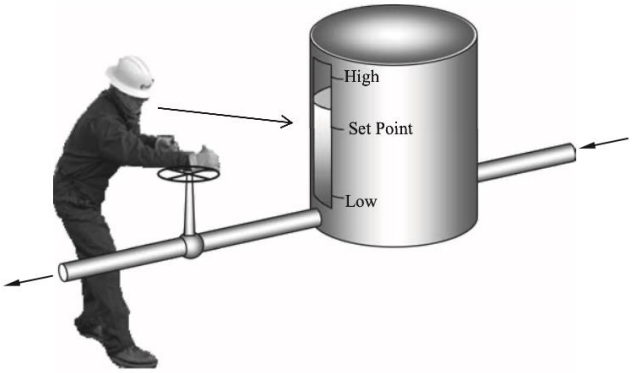


Figura 1.1. Sistem “primitiv” de control al nivelului uni tanc de stocare

De multe ori se iau în considerare procesele care funcționează în regim staționar. Aceasta este o simplificare logică ce permite inginerilor să proiecteze procese destul de complexe într-un timp rezonabil. Cu toate acestea, în realitate, procesele funcționează într-un mediu dinamic. Să ne imaginăm că proiectam un schimbător de căldură care utilizează apa de mare ca apă de răcire pentru a răci un flux de proces de la 100 °C la 50 °C. În etapa de proiectare trebuie să facem o estimare cu privire la temperatura apei de răcire (o valoare unică) și să presupunem că am ales o temperatură de 20 °C. Să ne gândim la șansele ca temperatura apei de răcire să fie exact 20 °C. Va eșua sistemul dacă temperatura reală a apei este de 15 °C sau de 10 °C? Răspunsul este da, sistemul nu va reuși să mențină temperatura la valoarea dorită de 50 °C, cu excepția cazului în care este instalată o supapă care poate ajusta în mod corespunzător debitul apei de răcire. În plus, trebuie instalat un senzor de temperatură pentru a măsura temperatura fluxului care iese din schimbătorul de căldură. Apoi, folosind temperatura măsurată și înregistrată, un operator poate verifica dacă temperatura este la valoarea corectă și poate regla în mod corespunzător debitul de apă de răcire pentru a corecta orice discrepanțe, așa cum se arată în figura 1.2. Temperatura apei de mare poate varia pe parcursul zilei, astfel încât operatorul va trebui să efectueze reglaje frecvente ale valvei pentru a menține temperatura fluxului  aproape de temperatura dorită.

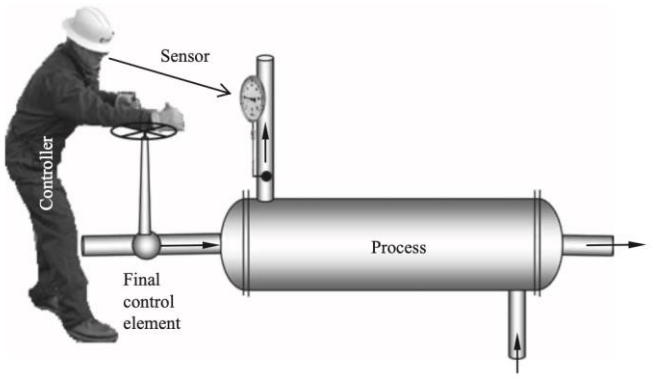


Figura 1.2. Sistem “primitiv” de control al temperaturii

Principalele tipuri de variabile întâlnite la acest sistem de control "primitiv" sunt următoarele:

- **Perturbațiile**. Orice proces este afectat de numeroși factori externi, iar mulți dintre aceștia variază într-un mod incontrolabil și imprevizibil. Aceștia sunt perturbațiile care fac ca procesul să devieze de la valoarea de referință. În cazul schimbătorului de căldură exemplificat mai sus, perturbațiile pot fi temperatura apei de răcire, temperatura și debitul fluxului de intrare sau îmbătrânirea echipamentului care poate crește rezistența la transferul de căldură. Unele perturbații pot fi măsurate în timp real, dar altele sunt dificil, costisitor sau chiar imposibil de măsurat.
- **Variabile manipulate**. Variabilele manipulate sunt acele variabile de proces care sunt ajustate de către regulator pentru a atinge obiectivele de control. O variabilă manipulată se mai numește și **variabilă de intrare** sau **variabilă de control**, pentru a indica faptul că reprezintă acțiunea de control care "intră" în proces. Una dintre cele mai frecvente variabile de intrare într-o stație de tratare biologică a apelor uzate este debitul de alimentare care poate fi manipulat cu ajutorul unei valve sau a unei pompe.
- **Variabile măsurate**. Variabilele măsurate sunt toate variabilele pentru care sunt instalați senzor sau dispozitive de măsurare care măsoară și transmit în mod continuu valoarea curentă a variabilei. Exemple de variabile măsurate într-o stație de tratare biologică a apelor uzate sunt temperatura, debitul, nivelul, pH-ul, oxigenul dizolvat etc. O variabilă măsurată pe care un regulator o menține la o valoarea dorită se numește **variabilă de ieșire** sau **variabilă controlată**. Valoarea la care se dorește a fi menținută variabila de ieșire de numește valoare de **referință** (en. set point). Această valoare este de obicei menținută constantă pe o perioadă lungă de timp dar uneori poate apărea necesitatea modificării acesteia, iar acest lucru trebuie făcut de un regulator. Dacă există o diferență între valoarea de referință și variabila controlată spunem ca există **eroare**. Matematic, eroarea este definită ca fiind diferența dintre valoarea de referință și variabila controlată, iar obiectivul regulatorului este de a face eroarea sa fie nulă.

În figura 1.3 este prezentată schema generală a unui sistem de control în buclă închisă unde pot fi observate toate tipurile de variabile care iau parte la proces, cât și elementele de bază ale sistemului de control și modul în care acestea sunt conectate între ele.

Elementul final de control (de obicei o valvă sau o pompă), împreună cu procesul și cu senzorul, formează sistemul fizic sau sistemul în **buclă deschisă**. Putem vedea în figura 1.3 că, atunci când senzorul este conectat la regulator, iar regulatorul acționează asupra elementului final de control, întregul sistemul are o structură circulară, ca o buclă, și se numește sistem în **buclă închisă**.

Un sistem de control în buclă închisă implică monitorizarea continuă a variabilei controlate și folosită acesteia pentru a face ajustări în proces prin modificări ale variabilei manipulate. Acțiunea regulatorului se bazează de obicei pe eroare, adică pe diferența dintre valoarea de referință și variabila de ieșire. În funcție de eroare (valoarea sa curentă, istoricul și tendința sa), controlerul ia măsuri corective. În termeni simpli, se poate descrie funcționarea unui sistem de control în buclă închisă astfel: monitorizare, detectare și corectare.

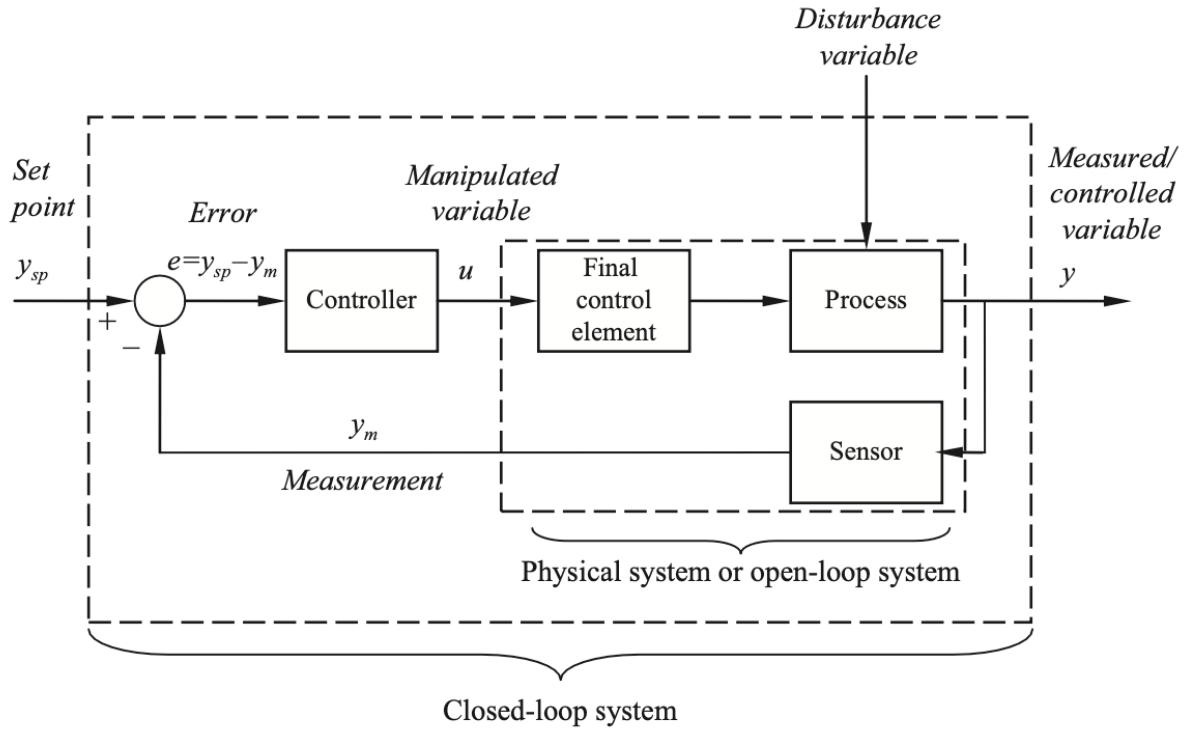


Figura 1.3. Elementele unui sistem de control în buclă închisă și interconexiunile acestora (Kravaris & Kookos, 2021)

Notațiile standard utilizate în controlul proceselor sunt, de asemenea, indicate în figura 3. Valoarea variabilei măsurate și controlate este notată cu $y$. Măsurătoarea este notată cu $y_m$, iar aceasta poate să nu corespundă cu $y$ într-un regim tranzitoriu, deoarece semnalul senzorului poate avea o întârziere față de modificarea variabilei fizice pe care o măsoară. Valoarea dorită sau valoarea de referință a variabilei controlate este notată cu $y_{sp}$. Semnalul de eroare $e = y_{sp} - y$ este, de asemenea, indicat în diagramă (cercul mic cu două săgeți spre interior cu semnele corespunzătoare și o săgeată spre exterior indică operația de scădere). Semnalul de eroare $e$ acționează regulatorul, care decide ajustările corespunzătoare, pentru a corecta eroarea și, în cele din urmă, pentru a o readuce la zero. Semnalul $u$ de la regulator stabilește valoarea variabilei manipulate a procesului, care este de fapt implementată de elementul de control final. În cele din urmă, senzorul detectează modificarea răspunsului sistemului, iar bucla este închisă.

## 2. Modelarea matematică a (bio)proceselor prin ecuații dinamice de bilanț de masă

Un model matematic este o reprezentare a cunoștințelor noastre despre un sistem fizic, care este "tradusă" într-un set de ecuații matematice. Scopul este de a utiliza modelul pentru a ne îmbunătăți înțelegerea comportamentului sistemului real (simularea modelului poate oferi informații), precum și pentru a proiecta și optimiza funcționarea procesului.

Modelele dinamice oferă o descriere cantitativă a comportamentului tranzitoriu al unui proces, în plus față de caracteristicile sale în regim staționar. Acest lucru este foarte util pentru a selecta condițiile de funcționare adecvate pentru proces, astfel încât să se evite starile tranzitorii nedorite. Este, de asemenea, foarte important pentru proiectarea regulatoarelor, deoarece un regulator poate modifica comportamentul dinamic al unui proces, în bine sau în rău.

La fel ca alte procese inginerești, procesele biotehnologice se supun unor legi universale, precum legea conservării masei care afirmă că într-un sistem, închis oricărui transfer de masă și energie, masa acestuia rămâne constantă în timp. Nu se poate nici crea, nici distruge materie, chiar dacă aceasta poate fi rearanjată în spațiu sau poate lua altă formă. Această lege fundamentală stă la baza dezvoltării ecuațiilor de bilanț de masă (sau bilanț de materiale) și reprezintă o unealtă importantă în studiul (bio)proceselor.

Bilanțul de masă este o manieră simplă de a cuantifica masa unui sistem la un anumit moment în timp:

$$\begin{pmatrix}\text{Masa sistemului}\\ \text{la momentul } t+\Delta t\end{pmatrix} = \begin{pmatrix}\text{Masa sistemului}\\ \text{la momentul } t\end{pmatrix} + \left(\begin{matrix}\text{Masa care intra în}\\ \text{sistem în intervalul}\\ t+\Delta t\end{matrix}\right) - \left(\begin{matrix}\text{Masa care iese din}\\ \text{sistem în intervalul}\\ t+\Delta t\end{matrix}\right) \tag{2.1}$$

Ecuația 2.1 este utilă atunci când se cunoaște foarte bine începutul și sfârșitul perioadei pe care se calculează bilanțul de masă, deoarece fiecare termen al ecuației este exprimat în unități de masă (e.g. g, kg, tone). Totuși, în științele inginerești se preferă exprimarea masei care este transportată prin sistem ca debit masic redat în unitate de masă pe unitate de timp (e.g. kg/h). În acest scop, ecuația 2.1 este împărțită la $\Delta t$ și rearanjată:

$$\frac{\begin{pmatrix}\text{Masa sistemului}\\ \text{la momentul } t+\Delta t\end{pmatrix} - \begin{pmatrix}\text{Masa sistemului}\\ \text{la momentul } t\end{pmatrix}}{\Delta t} = \frac{\left(\begin{matrix}\text{Masa care intra în}\\ \text{sistem în intervalul}\\ t+\Delta t\end{matrix}\right)}{\Delta t} - \frac{\left(\begin{matrix}\text{Masa care iese din}\\ \text{sistem în intervalul}\\ t+\Delta t\end{matrix}\right)}{\Delta t} \tag{2.2}$$

Se poate observa că termenul din partea stângă are semnificația ratei de acumulare a masei în sistem (viteza cu care se acumulează masa în sistem), iar termenii din dreapta au semnificația debitului masic (masa transportată prin sistem pe unitate de timp). Cu alte cuvinte:

$$\begin{pmatrix}\text{Rata de acumulare}\\ \text{de masă în sistem}\end{pmatrix} = \begin{pmatrix}\text{Debitul masic}\\ \text{de intrare}\end{pmatrix} - \begin{pmatrix}\text{Debitul masic}\\ \text{de ieșire}\end{pmatrix} \tag{2.3}$$

În această relație, termenul asociat ratei de acumulare de masă în sistem reprezintă rata de modificare a masei totale din sistem în raport cu timpul. Dacă debitul masic de ieșire este mai mare decât debitul masic de intrare, masa la $t + \Delta t$ va fi mai mică decât masa la momentul $t$, iar rata de acumulare va fi negativă, ”acumulare negativa”. Chiar dacă rata de acumulare a masei în sistem este negativă, masa totală a sistemului nu poate fi mai mică de 0.

Atunci când nu se acumulează masă în sistem (rata de acumulare este egală cu 0) se spune că sistemul este în stare staționară, relația 2.3 putând fi scrisă astfel:

$$\begin{pmatrix}\text{Debitul masic}\\ \text{de intrare}\end{pmatrix} = \begin{pmatrix}\text{Debitul masic}\\ \text{de iesire}\end{pmatrix} \tag{2.4}$$

Cuantificarea masei totale dintr-un sistem biotehnologic este utilă doar dacă se face pe elemente care nu se transformă în timpul procesului, de exemplu, bilanțul de masă poate fi scris pentru elemente atomice:

$$\begin{pmatrix}\text{Rata masică de}\\ \text{acumulare a}\\ \text{carbonului în sistem}\end{pmatrix} = \begin{pmatrix}\text{Debitul masic de}\\ \text{intrare în sistem}\\ \text{a carbonului}\end{pmatrix} - \begin{pmatrix}\text{Debitul masic de}\\ \text{ieşire din sistem}\\ \text{a carbonului}\end{pmatrix} \tag{2.5}$$

Majoritatea componenților, care prezintă interes într-un proces biotehnologic, sunt transformați: biomasa crește pentru că se hrănește cu substrat, produșii se formează sub influența biomasei etc. Un component poate să nu fie transportat din afara sistemului, ci să se formeze în interiorul sistemului din alți componenți care au fost aduși la rândul lor din afara sistemului. În concluzie, atunci când cuantificăm masa oricărui component care suferă transformări, este necesar să luăm în calcul rata de producere/consum a acestuia prin reacție. Bilanțul de masă scris pentru un component $i$ al sistemului poate fi scris astfel:

$$\begin{pmatrix}\text{Rata masică de}\\ \text{acumulare a unui}\\ \text{component în sistem}\end{pmatrix} = \begin{pmatrix}\text{Debitul masic de}\\ \text{intrare în sistem}\\ \text{al componentului}\end{pmatrix} - \begin{pmatrix}\text{Debitul masic de}\\ \text{ieşire din sistem}\\ \text{al componentului}\end{pmatrix} + \begin{pmatrix}\mathbf{Rata\ de\ producere}\\ \mathbf{a\ componentului}\\ \mathbf{prin\ reactie}\end{pmatrix} \tag{2.6}$$

Pentru simplificare am folosit ”producere” cu ”+”, înțelegând că la consumul componentului $i$ (e.g. consumul hranei/substratului pentru creșterea microorganismelor), termenul va fi negativ. Vom folosi relația 2.6, în această formă, ca fiind relația generală de bilanț de masă pe componenți, termenul de producere prin reacție putând deci lua și valori negative – ”producere negativă”. Relația 2.6 va fi folosită pentru cuantificarea oricărui component $i$ dintr-un proces biotehnologic.

Formularea corectă a ecuațiilor dinamice de bilanț este foarte importantă în modelarea matematică a proceselor biotehnologice, de aceea, metodologia de modelare trebuie sa fie coerentă și să furnizeze etapele necesare elaborării acestora.

**Elaborarea ecuațiilor dinamice de bilanț de masă**

O ecuație dinamică de bilanț de masă este o expresie matematică prin care se descrie dinamica numeroșilor componenți chimici sau biologici care sunt transportați prin sistem. Pentru elaborarea ecuațiilor dinamice de bilanț literatura de specialitate oferă câteva proceduri (Doran, 1995; Snape et al., 1995; Dochain, 2008), dar, oricare ar fi abordarea, înțelegerea procesului care urmează a fi modelat este esențială. În continuare, vor fi prezentate trei etape care pot fi parcurse pentru dezvoltarea ecuațiilor de bilanț de masa pentru orice component al unui proces biotehnologic.

*A. Alegerea zonei de bilanț*

O zonă de bilanț se alege în funcție de obiectivul modelării matematice și poate cuprinde volume foarte mici (e.g. un atom, o moleculă, o celulă etc.) sau extrem de mari (e.g. un continent, o planetă, un sistem solar etc.). Zonele de bilanț uzuale proceselor biotehnologice sunt de obicei limitate la un bioreactor sau o instalație care conține mai multe bazine, dar pot fi reduse la o bulă de gaz, un flocon de nămol activat sau un microorganism.

La stabilirea limitelor zonei de bilanț se va avea în vedere complexitatea sistemului din interiorul zonei de bilanț. Dacă un component modelat suferă transformări în mai multe puncte ale sistemului, acesta poate fi împărțit în sub-sisteme care vor fi modelate individual. Astfel, masa componentului în interiorul zonei de bilanț alese trebuie să fie constantă sau cvasi-constantă (astfel încât diferența dintre modelul matematic și mărimile măsurate să fie acceptabilă).

Figurile 2.1 și 2.2 ilustrează diversitatea zonelor de bilanț. Volumul de control poate fi continuu (figura 2.1 – exemplul unui reactor) sau discontinuu (figura 2.2 – frunzele unui copac). Modelarea se face la fel pentru ambele tipuri de volume de control.

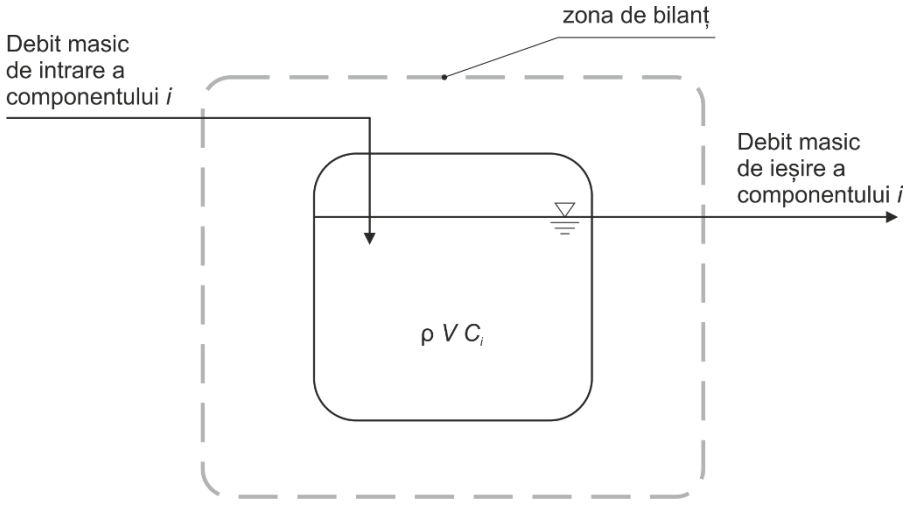


Figura 2.1. Zona de bilanț a unui reactor

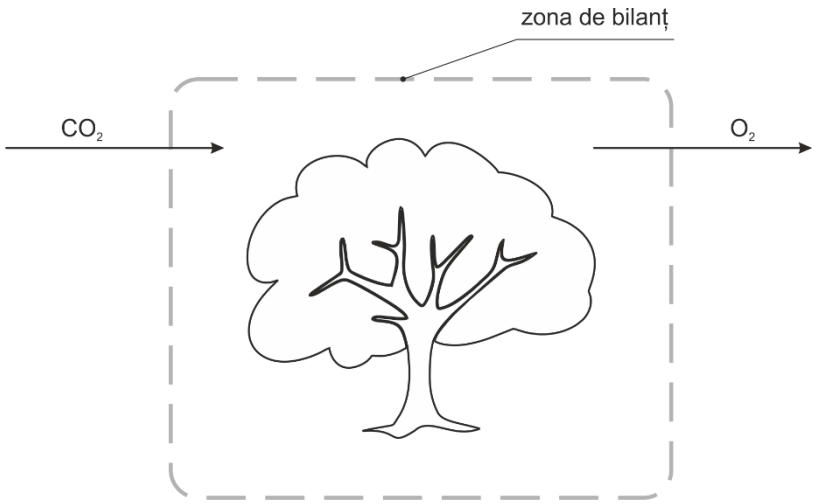


Figura 2.2. Zona de bilanț a unui arbore

După ce a fost aleasă zona de bilanț, trebuie stabilite componentele care urmează a fi modelate. În cazul unui proces biotehnologic vor fi stabilite speciile chimice sau biologice supuse investigării. Există numeroși compuși care pot fi subiectul bilanțului de masă, precum:

- biomasa microbiană (e.g. o anumită specie, o clasă de microorganisme, nămolul activat din procesele de tratare a apelor uzate),
- substraturile (e.g. sursa de carbon pentru creștere, substanțele organice poluante din apele uzate exprimate prin consumul chimic de oxigen – CCO, macro-nutrimente precum azotul și fosforul etc.),
- produșii de reacție (e.g. enzime, acizi organici, proteine, carbohidrați etc.),
- gazele dizolvate (e.g. oxigenul dizolvat, dioxidul de carbon dizolvat etc.),
- populațiile de organisme (e.g. microorganisme, plante, animale, oameni etc.)
- alți compuși de interes (e.g. pesticide, antibiotice etc.).

Procesele biotehnologice sunt, de fapt, procese care au loc în natură (procese biologice), dar care sunt intensificate în instalații speciale numite bioreactoare. Astfel, zona de bilanț care se alege în mod uzual atunci când avem de modelat procese biotehnologice este chiar bioreactorul. Deseori, instalațiile biotehnologice sunt compuse din mai multe bioreactoare, asociate cu bazine care servesc altor operații (e.g. preparare chimicale, stocare, decantare, degazare etc.). Aceste instalații complexe se împart în subsisteme care se modelează individual.

Modelarea proceselor biotehnologice trebuie redusă la concepte simple pentru a putea descrie cât mai ușor dinamica variabilelor de stare ale procesului. Variabilele de stare sunt componentele care caracterizează procesul și care pot fi măsurate sau estimate. Din punct de vedere al distribuției unui component într-un bioreactor, acestea pot fi:

- omogene – reactoare cu amestecare completă. Ele sunt sisteme cu parametri concentrați, care sunt descrise prin ecuații diferențiale ordinare la care variabila de derivare este timpul, $t$,
- eterogene (cu gradient de concentrație) – reactoare tubulare sau de tip piston. Ele sunt sisteme cu parametri distribuiți, care sunt descrise prin ecuații diferențiale cu derivate

parțiale la care mai apare o variabilă de derivare pe lângă timp (o coordonată spațială: adâncime, lungime).

*Reactorul cu amestecare completă*. Într-un reactor cu amestecare completă distribuția componentului care se dorește a fi modelat este uniformă în orice punct din volumul acestuia. Reactoarele cu amestecare continuă pot fi de trei tipuri:

- discontinue – tot substratul este adăugat de la început, este inoculat, iar pe tot timpul procesului nici nu se mai adaugă substrat proaspăt, nici nu se elimină conținutul reactorului. Cu alte cuvinte, nu există fluxuri masice de intrare și nici de ieșire. Rata de acumulare a componentului $i$ în bioreactor (ecuația 2.6) rezultă doar din ultimul termen, rata de producere a componentului prin reacție,
- semi-continue – la începutul procesului este inoculat doar un volum mic de substrat, iar pe tot timpul procesului se adaugă substrat proaspăt fără a se elimina conținutul bioreactorului. Alimentarea bioreactorului se oprește atunci când se atinge volumul util maxim. Există flux masic de intrare, dar nu există flux masic de ieșire.
- continue – bioreactorul este alimentat continuu și lucrează la volumul util maxim. Asta înseamnă că debitul masic de ieșire este egal cu debitul masic de intrare.

În figura 2.3 este ilustrat un bioreactor continuu cu amestecare completă. Concentrația oricărui component din fluxul masic de ieșire este egală cu concentrația acestuia în interiorul bioreactorului. Masa totală în sistem $M$ (kg) este dată de produsul dintre volumul tancului $V$ ($m^3$) și densitatea lichidului din tanc $\rho$ ($kg \cdot m^{-3}$), $M = V\rho$. Masa oricărui component $i$ din bioreactor este exprimată în unități de masă sau ca număr de moli, făcând produsul între volumul $V$ și concentrația componentului $i$, $C_i$ ($kg/m^{-3}$ sau $kmoli/m^{-3}$), astfel $i = C_i V$ va fi exprimată în kg sau kmoli.

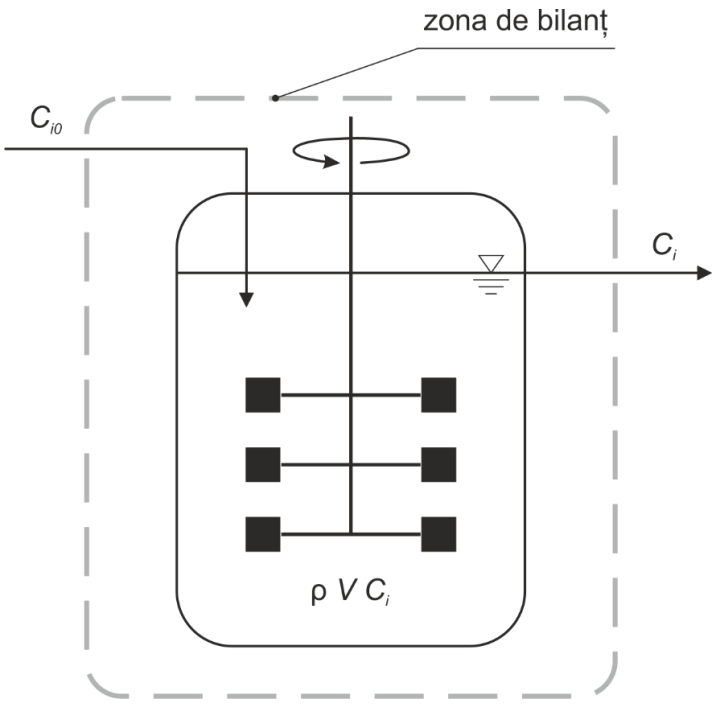


Figura 2.3. Zona de bilanț din jurul unui reactor cu amestecare completă

*Reactorul cu gradient de concentrație.* Un reactor cu gradient de concentrație este caracterizat prin faptul că distribuția componentului care se dorește a fi modelat nu este uniformă în orice punct din volumul acestuia, ci variază pe una dintre dimensiuni (înălțime sau lungime). Poate fi luat exemplul unui decantor care are o concentrație de particule aproape nulă în partea superioară și foarte mare în partea inferioară.

Acest tip de reactor este un concept important în modelarea bioproceselor. Într-un reactor tubular, concentrațiile componenților care sunt modelați variază de-a lungul reactorului (sau pe înălțime, dacă se modelează un decantor) chiar și atunci când acesta este operat în condiții staționare. Într-un reactor tubular ideal, concentrația componenților este constantă transversal, în orice zonă din reactor, dar variază axial. Cu alte cuvinte, concentrația unui component $i$ este constantă pe una dintre dimensiuni (ox sau oy), dar variază de-a lungul celeilalte dimensiuni (gradient de concentrație).

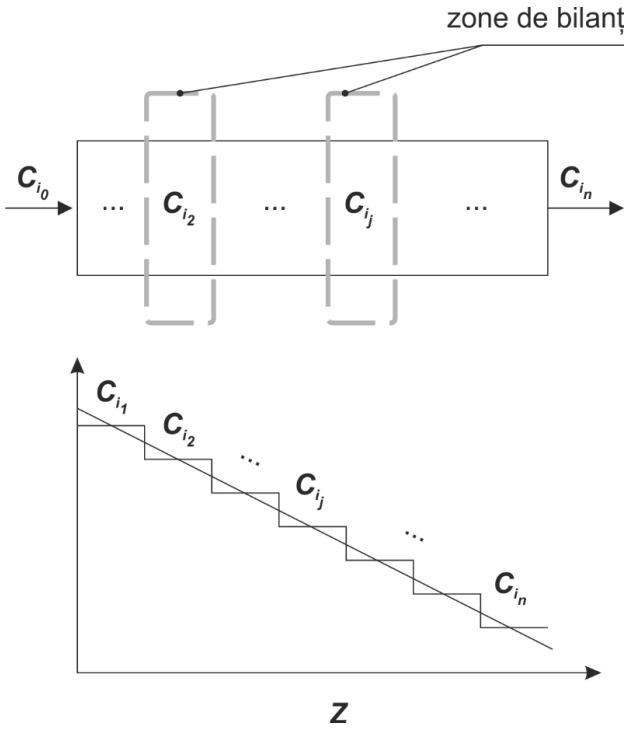


Figura 2.4. Zona de bilanț pentru un reactor de tip piston și gradientul aproximativ de concentrație

Figura 2.4 prezintă schema unui reactor tubular în care concentrația unui component $i$ scade de la intrare către ieșire. Acest reactor este modelat prin ecuații diferențiale cu derivate parțiale ($t$ și $z$ sunt variabilele de derivare). Totuși, concentrația componentului $i$ poate fi considerată constantă dacă se alege o zonă de bilanț suficient de mică, astfel încât concentrația acestuia să fie cvasi-constantă. Atât timp cât concentrația componentului $i$ poate fi considerată constantă, modelarea se va face prin ecuații diferențiale ordinare, singura variabilă de derivare rămânând timpul. Astfel, un reactor tubular poate fi împărțit în $n$ zone de bilanț, a căror concentrație $C_{i_j, j=\overline{1,n}}$ va fi modelată prin $n$ ecuații diferențiale ordinare. Cele $n$ zone de bilanț pot fi egale și atunci vorbim de sisteme cu parametri uniform distribuiți sau inegale, iar sistemele vor avea parametri neuniform distribuiți. Alegerea dimensiunii zonei de bilanț depinde foarte mult de

natura componentului modelat. Prin împărțirea reactorului tubular în mai multe zone în care concentrația componenților este uniformă, se poate afirma că volumul unui reactor tubular este format din mai multe reactoare cu amestecare completă.

Deseori, o instalație biotehnologică este formată din combinații de reactoare cu amestecare completă și cu gradient de concentrație, dar sunt și cazuri în care, în același reactor, concentrația unui component este uniformă în volumul reactorului, iar concentrația altui component are gradient, de exemplu:

- la tratarea biologică a apelor uzate un bazin aerob cu amestecare completă este înseriat cu un decantor pentru separarea nămolului activat de apa tratată;
- într-un fotobioreactor cu amestecare completă, toți componenții au concentrația uniformă, dar lumina este atenuată în interiorul culturii.

*B. Identificarea fluxurilor de transport*

După alegerea zonei de bilanț, a doua etapă importantă în dezvoltarea ecuațiilor dinamice de bilanț este identificarea fluxurilor de transport peste limitele sistemului.

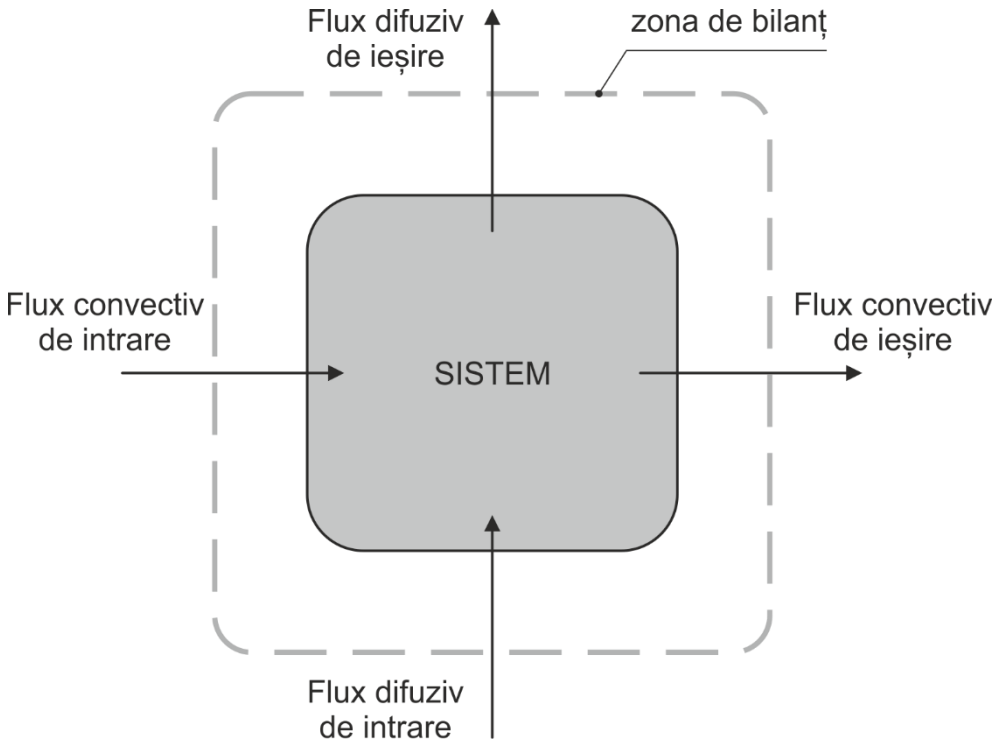


Figura 2.5. Zona de bilanț și fluxurile de intrare și de ieșire

Aceste fluxuri pot fi debite fizice bine definite și se pot împărți în (figura 2.5):

- fluxuri convective – sunt debite de lichide care transportă diverși componenți dizolvați sau în stare coloidală. Acestea pot fi identificate la intrarea în sistem (e.g. debitul de substrat proaspăt) și la ieșire (e.g. debitul de apă tratată care părăsește bazinul aerob),
- fluxuri difuzive – sunt debitele de gaz care sunt barbotate în cultura de microorganisme. Un flux difuziv de intrare are două componente: debitul propriu zis de gaz și rata de transfer

interfazic. Se pot barbota amestecuri de gaze (e.g. aerul) care au rate de transfer diferite (azotul este inert pe când oxigenul se dizolvă ușor). Debitele difuzive de ieșire sunt ignorate deseori, deoarece nu prezintă interes, de exemplu este important să se cunoască debitul de aer și rata de transfer masic gaz-lichid a oxigenului barbotat într-un proces aerob, dar este inutilă informația despre gazul epuizat care părăsește bazinul.

Termenii asociați fluxurilor de intrare sunt pozitivi, iar termenii asociați fluxurilor de ieșire sunt negativi.

*C. Exprimarea sub formă matematică*

În această ultimă etapă fiecare termen al ecuației generalizate a bilanțului de masă (ecuația 2.6) va fi exprimat sub formă matematică. Ecuațiile rezultate trebuie să conțină mărimi care pot fi măsurate sau estimate. Pentru simplificare vom folosi următoarea relație:

$$(\text{Acumulare}) = (\text{Intrare}) - (\text{Ieșire}) + (\text{Producție}) \tag{2.7}$$

Avantajul unei ecuații de bilanț este că are o bază logică, iar expresiile matematice vor păstra această bază.

*Termenul asociat ratei de acumulare*. Masa unui sistem deschis sau a unui component din sistem se poate modifica în timp, iar rata de acumulare a acestora este exprimată matematic ca fiind derivata masei în raport cu timpul:

$$\begin{pmatrix} \text{Rata masică de acumulare} \\ \text{a unui component } i \text{ în sistem} \end{pmatrix} = \left(\frac{dM_i}{dt}\right) \tag{2.8}$$

unde $M_i$ este masa componentului $i$ în sistem și poate fi exprimată în kg sau unități molare, iar timpul poate fi exprimat în orice unitate (secunde, ore etc.).

Așa cum a fost menționat anterior, în științele inginerești se preferă folosirea concentrației unui component în detrimentul masei. Pentru a transforma masa în concentrație este necesară cunoașterea unui volum de referință:

$$\frac{dM_i}{dt} = \frac{d(V\ C_i)}{dt} \tag{2.9}$$

unde $C_i$ este concentrația componentului $i$ exprimată în unități de masă pe unități de volum (e.g. kg/L, kmol/L etc.).

Pentru un component $i$ din sistem, care este prezent în formă gazoasă, se folosește legea gazului ideal care exprimă concentrația în funcție de presiunea parțială și de fracția molară:

$$p_i V = n_i RT \tag{2.10}$$

unde $p_i$ este presiunea parțială a componentului $i$ în faza gazoasă, $R$ este constanta ideală a gazelor, $n$ este numărul de moli și $T$ este temperatura. Concentrația molară componentului gazos poate fi exprimată astfel:

$$C_i^m = \frac{n_i}{V} = \frac{p_i}{RT} = \frac{y_i P}{RT} \tag{2.11}$$

unde $y_i$ este fracția molară a componentului $i$ în faza gazoasa, iar $P$ este presiunea totală a sistemului. Fracția molară este adimensională, iar suma fracțiilor molare ale unui amestec de gaze este egală cu 1. Astfel, termenul asociat ratei de acumulare pentru un component gazos poate fi exprimat în concentrație molară sau fracție molară:

$$\frac{dn_i}{dt} = \frac{d(V\, C_i^m)}{dt} = \frac{d\left(\frac{p_i V}{RT}\right)}{dt} = \frac{d\left(\frac{y_i PV}{RT}\right)}{dt} \tag{2.12}$$

*Termenii asociați fluxurilor convective de transport.* Debitele masice sunt produsul dintre debitele volumetrice și densitate:

$$(\text{Debit masic convectiv}) = (\text{Debit volumetric})\left(\frac{\text{Masă}}{\text{Volum}}\right) \tag{2.13}$$

Dacă exprimăm masa unui component ca $M_i = VC_i$, substituind în relația 2.13, rezultă că debitul masic al unui component $i$ este:

$$M_i = FC_i \tag{2.14}$$

unde $F$ este debitul volumetric care se măsoară în unități de masă pe volum (e.g. mL/min, L/h etc.).

Modelarea fluxurilor difuzive este ceva mai complexă. Ecuațiile de bilanț de masă scrise pentru componenți gazoși dizolvați în lichid au un termen în plus, care descrie transferul de masă gaz-lichid. Uneori acești termeni includ, explicit, și debitul molar de intrare. Pentru oxigen acest termen poate avea următoarea formă:

$$N_{O_2} = k_L a\left(C_{O_2}^* - C_{O_2}\right) \tag{2.15}$$

unde $N_{O_2}$ este rata de transfer masic gaz-lichid, $k_L a$ este coeficientul de transfer masic gaz-lichid, $C_{O_2}^*$ este concentrația de oxigen la saturație, iar $C_{O_2}$ este concentrația actuală de oxigen. $k_L a$ poate fi modelat, astfel încât să conțină explicit debitul molar de oxigen (flux difuziv de intrare).

Fluxurile difuzive de ieșire sunt rareori măsurate, dar rezultă din diferența dintre fluxul difuziv de intrare și rata de transfer masic gaz-lichid.

*Termenul asociat ratei de producere* permite exprimarea producerii sau consumului unui component printr-o reacție chimică sau biologică (*e.g.* rata de creștere a microorganismelor, rata de consum a substratului, rata de formare a unui produs de reacție). În procesele biotehnologice rata de producere a unui component este exprimată în unități de masă pe unități de volum pe timp (e.g. g/L/h):

$$\begin{pmatrix}\text{Rata masică de producere}\\ \text{a componentului } A\end{pmatrix} = \begin{pmatrix}\text{Rata volumetrica}\\ \text{de producere}\end{pmatrix}\begin{pmatrix}\text{Volumul}\\ \text{sistemului}\end{pmatrix} \tag{2.16}$$

$$R_i = r_i V \tag{2.17}$$

unde $R_i$ este rata masică de reacție, iar $r_i$ este rata volumetrică de reacție. Pot fi, de asemenea, folosite cantități molare echivalente (e.g. rata molară a carbonul anorganic total). Rata volumetrică de reacție este pozitivă pentru producere și negativă pentru consum.

**Modelul dinamic general.**

Rata acumulării de masă a oricărui component $i$ din cadrul unui sistem (relația 2.6), exprimată luând în considerare bilanțul de masă, regrupează două tipuri de termeni:

- termeni de conversie (ce descriu cineticile reacțiilor chimice și biologice și randamentele de conversie),
- termenii de transport (ce descriu tranziția masei prin proces, în stare lichidă și/sau gazoasă, și fenomenele transferului de fază) (Sablani *et al.*, 2006; Dochain, 2008).

Modificarea în timp a ratei de acumulare de masă a unui component $i$ într-un sistem este, cel mai adesea, exprimată folosind noțiunea de concentrație (ecuația 2.9).

O serie de ipoteze pot fi formulate pentru un proces biotehnologic:

- procesul are loc într-un reactor cu agitare completă (notă: un bioreactor cu gradient de concentrație este împărțit în $n$ reactoare cu amestecare completă),
- există doar fluxuri convective nu și difuzive (nu se barbotează gaze în reactor),
- există $n$ fluxuri convective de intrare.

În aceste condiții, o ecuație dinamică generală de bilanț de masă poate fi definită după cum urmează:

$$\frac{dVC_i}{dt} = r_i V + \sum_{j=1}^{n} F_{in,j} C_{i,0} - F_{out} C_i \tag{2.18}$$

unde $C_{i,0}$ este concentrația componentului $i$ în fluxul de alimentare, iar $r_i$ este rata volumetrică de producție sau de consum a componentului $i$. Termenul de conversie, $r_i V$, descrie producția

sau consumul unui component prin reacții chimice ori biologice (*e.g.* ratele de creștere a biomasei, asimilare a substratului, formare de produs etc.) în timp ce ceilalți doi termeni descriu fluxurile de transport convectiv a componentului $i$ peste limitele sistemului. Modelul dinamic general (2.18) poate fi simplificat din moment ce foarte rar poate fi identificată existența unui component în mai mult de un singur flux de intrare, $F_{in}$, ceea ce permite înlocuirea termenului de intrare cu $F_{in}C_{i,0}$. Fluxul de ieșire, $F_{out}$, al unui reactor cu amestecare completă are proprietăți identice cu cele ale fluidului din interiorul reactorului.

## 3. Studiu de caz. Modelarea procesului de tartare biologică aerobă a apelor uzate.

Tratarea biologică a apelor uzate este mai mult decât o necesitate, este o responsabilitate a fiecărui producător, care trebuie să îmbunătățească continuu procesul. Există mai multe procese biologice de tratare a apelor uzate, și de aici numeroase modele matematice. Procesele de tratare biologică a apelor uzate sunt foarte complexe, puternic neliniare și caracterizate de incertitudini parametrice.

Vom dezvolta în continuare un model simplificat pentru tratarea biologică aerobă a apelor uzate, care să ne ajute să înțelegem și să simulăm pe calculator acest proces.

### Descrierea procesului de tratare aerobă a apelor uzate

Procesul de tratare biologică aerobă a apelor uzate are ca obiectiv transformarea substanțelor organice poluante, prezente în apa uzată, în produși oxidați stabili și biomasă nouă (nămol activat). Tratarea aerobă a apelor uzate este un proces în care substanțele organice poluante prezente în apa uzată sunt oxidate la $CO_2$, $H_2O$, $NH_4^+$ și celule noi.

O instalație pentru tratarea biologică a apelor uzate reprezintă o asociere de procese sau de unități de tratare distincte care generează un efluent de o anumită calitate dintr-un influent cu debit și compoziție cunoscute.

Tratarea biologică aerobă a apelor uzate, numită și tratament secundar, se folosește de microorganismele prezente în abundență în mediul natural pentru conversia substanțelor organice poluante în masă microbiană densă (nămol activat) care poate fi separată ușor de apa epurată prin procese convenționale de sedimentare. Procesul de tratare aerobă a apelor uzate este realizat în principal de microorganisme heterotrofe care au capacitatea de a descompune diferitele substanțe organice poluante prin două procese diferite, oxidarea și biosinteza, ambele ducând la eliminarea lor din influent (apa uzată).

Oxidarea sau respirația (ecuația 3.1) duce la produși finali anorganici, în timp ce biosinteza (ecuația 3.2) transformă substanțele organice solubile și coloidale în biomasă, care poate fi eliminată prin sedimentare. Atunci când concentrația de substanțe organice poluante, care reprezintă sursa de hrană a microorganismelor care formează nămolul activat, devine insuficientă, celulele vor intra într-un proces de respirație endogenă pentru a obține energia necesară întreținerii (auto-oxidare, ecuația 3.3). Toate cele 3 procese au loc simultan în bazinul aerob și pot fi exprimate stoichiometric după cum urmează (Gray, 2005):

Oxidare

$$COHNS + O_2 + \text{microorganisme} \rightarrow CO_2 + NH_3 + \text{alţi produşi finali} + \text{energie} \quad (3.1)$$

Biosinteză

$$COHNS + O_2 + \text{microorganisme} \rightarrow C_5H_7NO_2 \quad (3.2)$$

Auto-oxidare

$$C_5H_7NO_2 + 5O_2 \rightarrow 5CO_2 + NH_3 + 2H_2O + \text{energie} \quad (3.3)$$

unde $COHNS$ și $C_5H_7NO_2$ sunt expresiile generale pentru substanța organică și pentru microorganisme.

Pentru a asigura mineralizarea substanțelor organice poluante, amestecul lichid din bazinul aerob este îmbogățit cu oxigen dizolvat care este asigurat prin barbotare de aer cu ajutorul unor turbosuflante. Microorganismele care se hrănesc cu aceste substanțe organice poluante se aglomerează între ele formând flocoane de nămol activat care au capacitatea de a sedimenta.

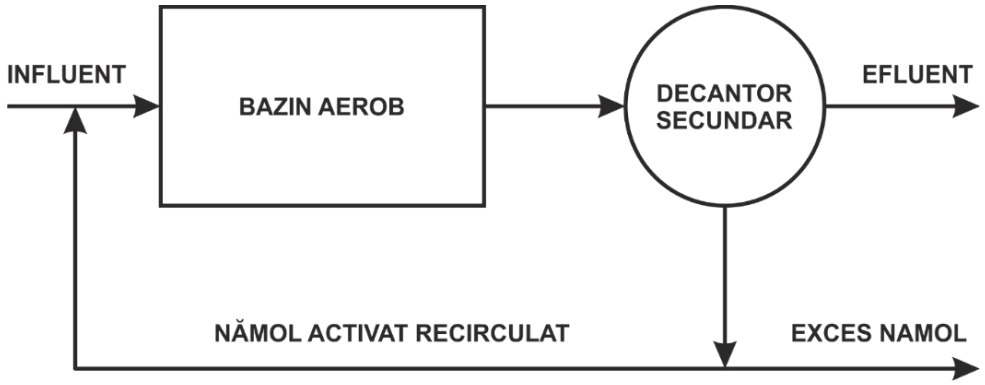


Figura 3.1. Sistem convențional de tratare aerobă a apelor uzate

Astfel, un proces clasic de tratare aerobă a apelor uzate constă în 3 elemente fundamentale (figura 3.1):

- **Bazin aerob** în care are loc oxidarea substanțelor organice poluante. Influentul bazinului aerob (apa uzată) este amestecat cu nămolul activat recirculat care a sedimentat în decantor, făcând ca timpul de retenție al nămolului activat în instalație să fie mai mare decât timpul de retenție hidraulic. Acest aspect permite menținerea unui număr mare de microorganisme în bazinul aerob, capabile să oxideze eficient materia organică poluantă într-o perioadă scurtă de timp.

- **Decantor** folosit pentru sedimentarea flocoanelor de nămol activat produse în bazinul aerob. O parte din nămolul activat sedimentat este recirculat, iar excesul este eliminat.
- Sistemul de **recirculare** este vital pentru stațiile de tratare a apelor uzate. În afară de rolul său descris anterior, previne fenomenul de "wash-out" care poate a avea loc atunci când debitul de influent este foarte mare (e.g. în timpul unei perioade ploioase).

**Modelarea matematică a bazinului aerob**

Așa cum a fost menționat anterior, creșterea microorganismelor heterotrofe care populează bazinul aerob necesită oxigen dizolvat. Totuși, creșterea microorganismelor heterotrofe se datorează substratului solubil (substanțele organice poluante din apa uzată) care reprezintă sursa de carbon și energie a acestora. Ele sunt, deci, limitate nu doar de oxigen ci și de disponibilitatea carbonului organic, macro-nutrimentelor (i.e. azot, fosfor etc.) și a altor factori de creștere. Creșterea biomasei este considerată ca fiind procesul principal, în timp ce celelalte procese sunt cuplate prin parametri stoichiometrici, adică celelalte procese sunt proporționale cu creșterea nămolului activat. Cel mai simplu model matematic care să descrie bazinul aerob este prezentat în tabelul 3.1. În concordanță cu nomenclatura IWA (International Water Association), cu $X$ sunt notate componentele insolubile, iar cu $S$ cele solubile. Indicii acestora se referă la componenții individuali: biomasa, B, substrat, S și oxigen dizolvat, O.

$S_S$ reprezintă sursa de carbon (substratul organic cu carbon) care se exprimă de cele mai multe ori ca CCO (Consumul Chimic de Oxigen) și se măsoară în mg $O_2 \cdot L^{-1}$.

Macro-nutrimentele (i.e. azot, fosfor etc.) nu sunt modelate aici, ci sunt considerate a fi prezente în concentrații suficiente încât să nu limiteze creșterea microorganismelor heterotrofe.

Tabelul 3.1. Termenii cinetici și parametrii stoichiometrici ai procesului de tratare biologică aerobă a apelor uzate

<table>
<tr><td colspan="2">Component → $i$<br>$j$ Proces ↓</td><td>1<br>$X_B$</td><td>2<br>$S_S$</td><td>3<br>$S_O$</td><td>Rata procesului, $\rho_j$<br>[g·L$^{-1}$·h$^{-1}$]</td></tr>
<tr><td>1</td><td>Creștere</td><td>1</td><td>$-\frac{1}{Y}$</td><td>$-\frac{1-Y}{Y}$</td><td>$\mu_{max}\left(\frac{S_S}{K_S+S_S}\right)\left(\frac{S_O}{K_O+S_S}\right)X_B$</td></tr>
<tr><td>2</td><td>Declin</td><td>$-1$</td><td></td><td></td><td>$\mu_d X_B$</td></tr>
<tr><td colspan="2">Ratele de conversie observate [g·L$^{-1}$·h$^{-1}$]</td><td colspan="3">$r_i = \sum_j r_{ij} = \sum_j \nu_{ij}\rho_j$</td><td rowspan="2">*Parametri cinetici*:<br>Viteza specifică maximă de creștere: $\mu_{max}$<br>Constanta de semi-saturație: $K_S$<br>Viteza specifică de declin: $\mu_d$<br>Constanta de saturație pentru oxigenul dizolvat: $K_O$</td></tr>
<tr><td colspan="2">*Parametri stoichiometrici*, $\nu_{ij}$:<br>Randament de creștere: $Y$</td><td>Biomasă<br>[g·L$^{-1}$]</td><td>Substrat<br>[g·L$^{-1}$]</td><td>Oxigen dizolvat<br>[g·L$^{-1}$]</td></tr>
</table>

Ele pot fi modelate prin ecuații stoichiometrice simple, dar sunt deseori neglijate deoarece raportul C:N:P uzual al apelor uzate municipale este de aproximativ 100:5:1, raport ce asigură o bună creștere a nămolului activat și deci o bună capacitate de epurare. Dacă macro-nutrimentele depășesc raportul recomandat, nu pot fi eliminate prin tratament biologic aerob. În acest caz se adaugă alte bazine în instalația de tratare biologică, capabile să întrețină procese

precum eliminarea biologică a azotului prin nitrificare și denitrificare sau eliminarea biologică a fosforului.

Creșterea biomasei este exprimată printr-un model dublu de limitare de substrat și oxigen, iar rata de declin este de ordinul întâi în raport cu concentrația de biomasă. Ratele volumetrice de reacție care rezultă din tabelul 3.1 pot fi scrise astfel:

$$r_{X_\mathrm{B}} = \mu_{max}\left(\frac{S_\mathrm{S}}{K_\mathrm{S}+S_\mathrm{S}}\right)\left(\frac{S_\mathrm{O}}{K_\mathrm{O}+S_\mathrm{S}}\right)X_\mathrm{B} - \mu_d X_\mathrm{B} \tag{3.4}$$

$$r_{S_\mathrm{S}} = -\frac{1}{Y}\left(\mu_{max}\left(\frac{S_\mathrm{S}}{K_\mathrm{S}+S_\mathrm{S}}\right)\left(\frac{S_\mathrm{O}}{K_\mathrm{O}+S_\mathrm{S}}\right)X_\mathrm{B}\right) \tag{3.5}$$

$$r_{S_\mathrm{O}} = -\frac{1-Y}{Y}\left(\mu_{max}\left(\frac{S_\mathrm{S}}{K_\mathrm{S}+S_\mathrm{S}}\right)\left(\frac{S_\mathrm{O}}{K_\mathrm{O}+S_\mathrm{S}}\right)X_\mathrm{B}\right) \tag{3.6}$$

Creșterea microorganismelor heterotrofelor are loc doar prin consumul substanțelor organice cu carbon, a căror mineralizare completă nu are loc decât în prezența oxigenului dizolvat. De aceea, al doilea factor de tip limitare (i.e. $S_\mathrm{O}/(K_\mathrm{O}+S_\mathrm{S})$) este ales pe criterii matematice și nu mai are aceeași semnificație ca modelul Monod original. Acest factor are rolul unei funcții matematice continue care are valori cuprinse între 0 și 1. Constanta de saturație pentru oxigen, $K_\mathrm{O}$, are de obicei o valoare mică, ceea ce denotă că modelul de limitare pentru oxigen este aproape de 1 la valori moderate de oxigen dizolvat, dar tinde către 0 când concentrația de oxigen dizolvat scade către 0.

Rata volumetrică de eliminare a substratului și rata volumetrică de consum a oxigenului dizolvat sunt proporționale cu rata volumetrică de creștere a biomasei (ecuația 3.5 și 3.6).

Deoarece concentrația de oxigen dizolvat este menținută în bazinul aerob prin barbotare cu aer, trebuie luat în calcul transferul masic gaz-lichid:

$$N_{\mathrm{O}_2} = (K_L a)_{\mathrm{O}_2}\left(S_{\mathrm{O,sat}} - S_\mathrm{O}\right) \tag{3.7}$$

unde $(K_L a)_{\mathrm{O}_2}$ este coeficientul global de transfer masic, iar $S_{\mathrm{O,sat}}$ este concentrația la saturație de oxigen dizolvat.

Rata de transfer a oxigenului gaz-lichid poate fi exprimată în funcție de debitul de aerare după cum urmează:

$$(K_L a)_{\mathrm{O}_2} = \alpha W \tag{3.8}$$

unde $\alpha$ este coeficientul de difuzie care se referă la rata de transfer a oxigenului și deci la eficiența difuzoarelor de aer, iar $W$ este rata de aerare.

**Modelarea matematică a decantorului (Ifrim, 2012)**

Flocoanele de nămol activat formate în bazinul aerob sunt lăsate să sedimenteze într-un bazin separat, limpezind astfel apa epurată. Amestecul lichid care intră în decantor se împarte în două fluxuri (figura 3.3), un flux care duce spre partea superioară a decantorului (flux de deversare) și un flux care este extras prin partea inferioară (flux inferior). Peste aceste două fluxuri se suprapune sedimentarea gravitațională a flocoanelor de nămol activat care duce la apariția unui gradient de concentrație pe adâncimea decantorului. Concentrația de nămol activat va fi foarte scăzută în fluxul de deversare și foarte concentrată în fluxul inferior. Acest gradient de concentrație pe adâncime și timp creează un sistem cu parametri distribuiți, a cărui modelare matematică este destul de complexă.

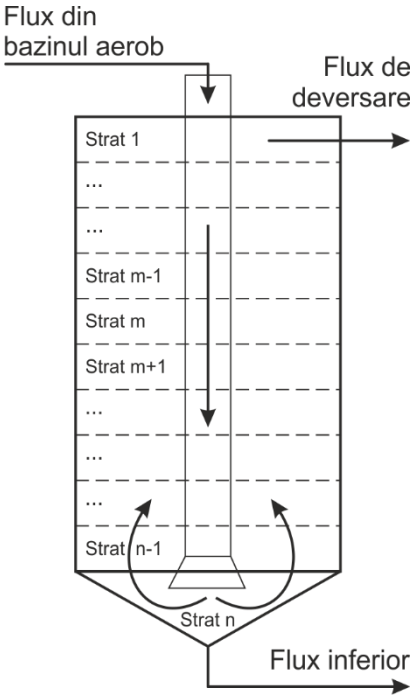


Figura 3.3. Modelul multistrat al unui decantor vertical

Dificultatea modelării vine din faptul că viteza de sedimentare a flocoanelor de nămol activat depinde de foarte multe variabile, precum densitatea nămolului, concentrația de solide în suspensie și debitul din bazinul aerob, dimensiunea flocoanelor de nămol, timpul de retenție etc. O concentrație mare de nămol activat poate duce la compactarea acestuia, un fenomen care are loc la baza decantorului. Alteori, o concentrație scăzută de flocoane de nămol poate duce la o sedimentare ineficientă. Aceste fenomene sunt greu de prezis și de aceea putem adopta un model empiric care să genereze un răspuns satisfăcător.

Figura 3.3 ilustrează un modelul multistrat al unui decantor vertical. Decantorul este împărțit în $n$ straturi, fiecare dintre ele fiind cvasi-omogen. Cu alte cuvinte, decantorul (un bazin de tip piston) este împărțit în $n$ bazine cu amestecare completă pentru a evita modelele cu derivate parțiale (de adâncime și timp). Obiectivul acestui model este de a oferi o predicție satisfăcătoare a concentrației de nămol activat sedimentat (care este diferită de concentrația de nămol activat din bazinul aerob) care este recirculat între decantor și bazinul aerob. Nămolul recirculat influențează toate ratele procesului de tratare aerobă: rata de creștere a nămolului activat, rata

de consum a substratului și rata de consum a oxigenului dizolvat. În acest scop se va modela doar stratul $n$, celelalte neprezentând interes (Figura 3.4).

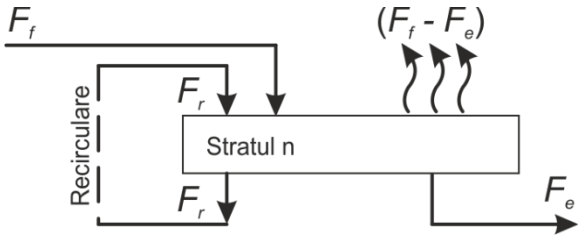


Figura 3.4. Bilanțul de masă la baza decantorului (stratul $n$)

Deoarece se presupune că ratele cinetice sunt nule în decantor (rata de creștere a nămolului activat, rata de consum a substratului și rata de consum a oxigenului dizolvat), trebuie doar identificate fluxurile de biomasă care trec peste limitele zonei de bilanț (figura 3.4). Se poate scrie o ecuație de bilanț luând în calcul biomasa transportată prin fluxul de alimentare din bazinul aerob (fiind un proces continuu, acesta este egal cu fluxul de alimentare cu influent), $F_f$, fluxul de recirculare, $F_r$, și fluxul de nămol în exces, $F_e$. Ecuația de bilanț pentru nămolul activat recirculat din stratul $n$ are următoarea formă:

$$\frac{dX_{\mathrm{R}}}{dt} = \frac{F_f}{V_s}X_{\mathrm{B}} + \frac{F_r}{V_s}X_{\mathrm{B}} - \frac{F_e}{V_s}X_{\mathrm{R}} - \frac{F_r}{V_s}X_{\mathrm{R}} - \eta\frac{\left(F_f - F_e\right)}{V_s}X_{\mathrm{R}} \tag{3.9}$$

unde $X_{\mathrm{R}}$ este concentrația de nămol activat recirculat, $V_s$ este volumul stratului $n$, iar $\eta$ este un parametru subunitar care cuantifică fracția de biomasă care migrează în stratul $n-1$ odată cu fluxul de deversare.

**Aplicație 1. Modelarea și controlul nivelului unui tanc de stocare**

**Aspecte teoretice**

În figura de mai jos este reprezentat un tanc cilindric de stocare pentru lichid. Aria secțiunii transversale este notată $A$, iar nivelul acestuia $h$. Volumul util al tancului este deci $V = Ah$. Debitul volumetric de intrare este $F_{in}$, de unde rezultă că debitul masic de intrare este $M_{in} = \rho F_{in}$. Debitul volumetric de ieșire este $F_{out}$ și deci debitul masic de ieșire este $M_{out} = \rho F_{out}$, unde $\rho$ este densitatea lichidului.

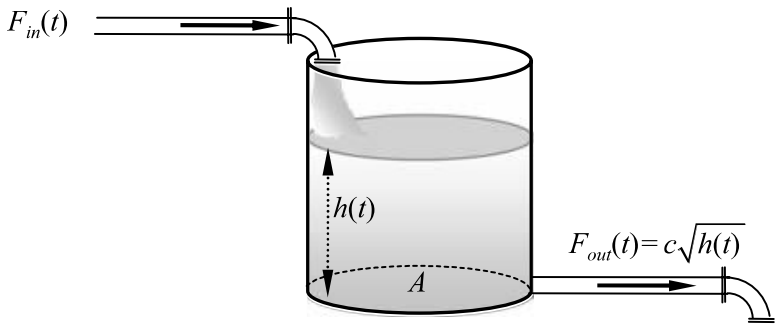


Figura 1. Tanc de stocare pentru lichid (Kravaris & Kookos, 2021)

Masa lichidului din tancul de stocare este deci:

$$\frac{dM(t)}{dt} = M_{in}(t) - M_{out}(t) \tag{1}$$

sau

$$\frac{d(\rho V(t))}{dt} = \rho F_{in}(t) - \rho F_{out}(t) \tag{2}$$

Pentru un component pur la temperatura constanta densitatea $\rho$ este constanta, iar bilanțul de masă poate fi simplificat astfel:

$$\frac{dV(t)}{dt} = F_{in}(t) - F_{out}(t) \tag{3}$$

Debitul volumetric de ieșire $F_{out}$ este o funcție de nivelul de lichid $h$ și este de obicei exprimat sub forma $F_{out} = f(h)$. În multe aplicații $F_{out}$ este considerat proporțional cu rădăcina pătrată a nivelului de lichid din tanc $h$:

$$F_{out}(t) = c\sqrt{h(t)} \tag{4}$$

unde $c$ este o constantă. Substituind ecuația 4 în 3 și exprimând volumul ca $V = Ah$, vom obține:

$$\frac{d(Ah(t))}{dt} = F_{in}(t) - c\sqrt{h(t)} \quad (5)$$

Ecuația 5 poate fi rezolvată numeric pentru orice $F_{in}(t)$, obținând evoluția nivelului de lichid în tancul de stocare.

Debitului volumetric de intrare $F_{in}(t)$ este o variabilă independentă (din exteriorul tancului de stocare) care este *cauza* modificărilor din interiorul tancului. $F_{in}$ se numește ***variabilă de intrare***.

Nivelul lichidului din tancul de stocare $h(t)$ descrie *efectul* variabilei de intrare asupra tancului. $h$ se numește ***variabilă de ieșire***. O variabilă de ieșire este o mărime dependentă.

În același timp $h(t)$ este soluția modelului matematic (ecuația 5) care oferă informații complete asupra stării nivelului tancului de stocare în orice moment. $h$ este totodată ***variabilă de stare***.

$A$ – aria secțiunii transversale a tancului de stocare și constanta $c$ sunt *parametrii* modelului.

**Cerințe**

Fie un tanc de stocare cu aria secțiunii transversale $A$ = 20 m$^2$, alimentat cu un debit de intrare $F_{in}$ = 2 m$^3$/h. Tancul are un volum util de 100 m$^3$, de unde rezultă că nivelul maxim este $h_{max}$ = 5 m. Constanta $c$ = 0.8 m$^2$/h.

a. Să se construiască un model Simulink și să se simuleze nivelul tancului de stocare $h$, pe o perioadă de 300 de ore ($t_f$ = 300). Se presupune că tancul este gol în momentul începerii alimentării: $h(t_0) = 0$.
   - Să construiască un fișier script care să apeleze modelul Simulink și să se reprezinte grafic nivelul tancului $h$ și debitul de ieșire $F_{out}$.
   - Constanta $c$ poate avea semnificația unui robinet manual. Modificați valoarea acesteia în intervalul [0 1] și observați efectul acesteia asupra $h$ și $F_{out}$.

b. Să se implementeze un regulator ON/OFF pentru nivelul tancului de stocare la $h$ = 5 m, deoarece alimentarea se face printr-o valvă ON/OFF.
   - Să se adauge $F_{in}$ la graficul anterior.

c. Să se implementeze un regulator ON/OFF cu histerezis pentru nivelul tancului de stocare pe intervalul 4.9 si 5.1 m.

**Implementare în Matlab/Simulink**

Vom implementa în Simulink următoarea ecuație diferențială:

$$\frac{dh(t)}{dt} = \frac{F_{in}(t) - c\sqrt{h(t)}}{A}$$

Putem începe prin tragerea unui bloc *Integrator* la a cărui ieșire legăm un bloc *Output port*. Blocul *Output port* face ca datele să fie disponibile în *Workspace*. Condițiile inițiale $h(t_0) = 0$ și timpul final de simulare $t_f = 300$ se introduc așa cum poate fi văzut în Figura 2.

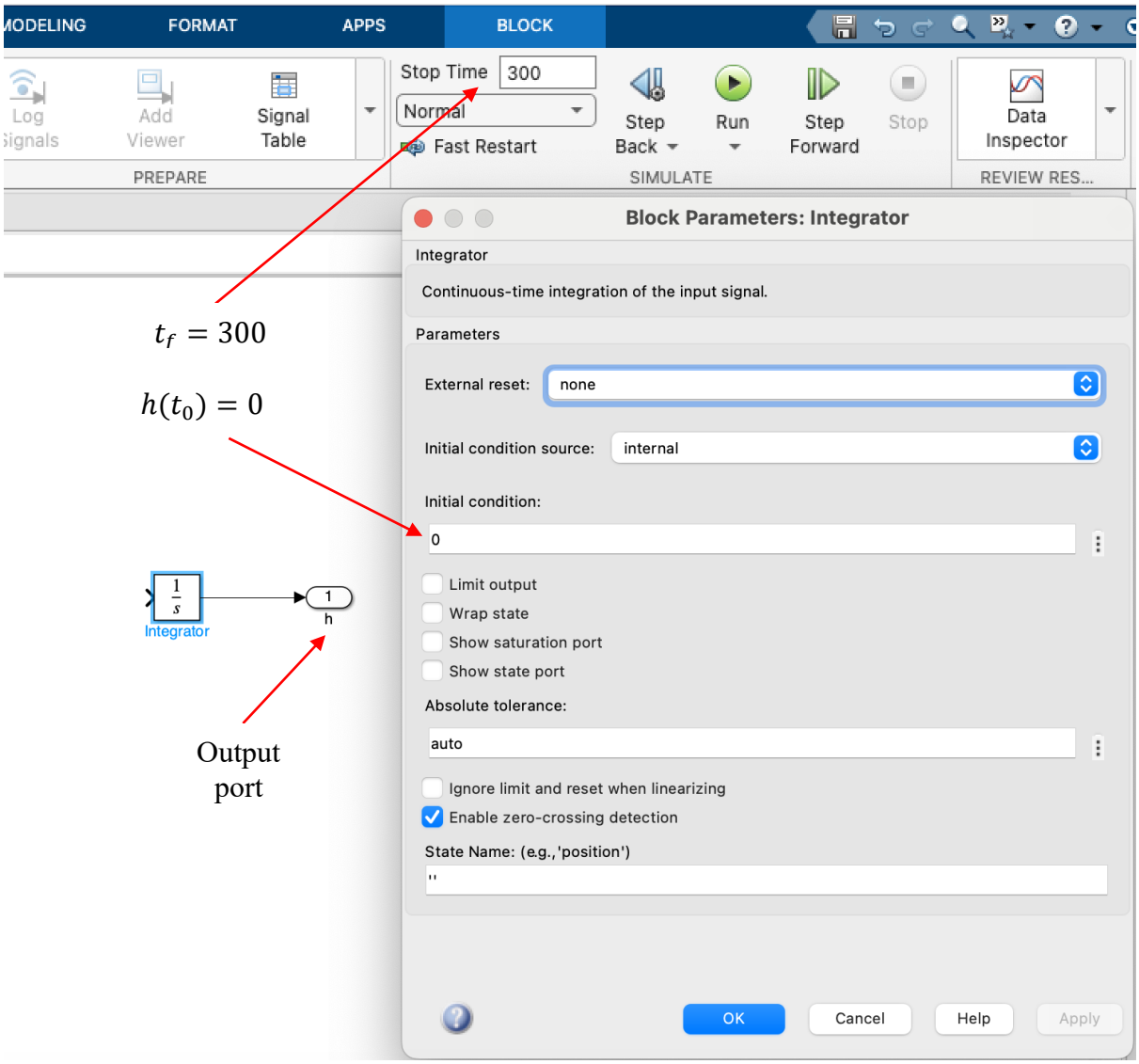


Figura 2. Setarea condițiilor inițiale și a timpului de simulare

În Figura 3 găsim un exemplu de dezvoltare a modelului matematic prin asocierea blocurilor *Divide*, *Sum*, *Gain*, *Constant*, *Sqrt* și *To Workspace*.

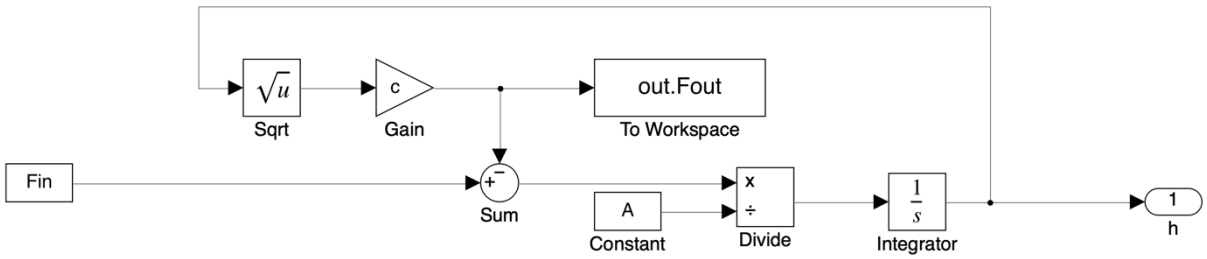


Figura 3. Modelul Simulink al nivelului tancului de stocare

Acest model solicită valorile parametrilor modelului. Aceștia vor fi încărcați în *Workspace* cu ajutorul unui fișier script:

```
clear; close

%% Parametrii modelului matematic

Fin = 2;    % Debit de intrare (m3/h)
c = 0.8;    % constanta (m2/h)
A = 20;     % Aria sectiunii transversale (m2)
```

Modelul Simulink poate fi apelat astfel:

```
%% Apelare model Simulink

StopTime = 300;

SimIn = Simulink.SimulationInput('TancStocare');
SimIn = SimIn.setModelParameter("StopTime",num2str(StopTime));

SimOut = sim(SimIn);
```

TancStocare este numele fișierului Simulink. Pentru apelarea modelului s-a folosit funcția sim. Aceasta poate fi folosită apelând numele modelului (i.e. SimOut = sim('TancStocare')). Totuși, definirea unui obiect SimulationInput vine cu avantajul că modelul Simulink poate să fie modificat fără a fi necesară deschiderea acestuia. În exemplul de mai sus s-a modificat timpul de simulare $t_f = 300$.

Obiectul SimOut care poate fi găsit în *Workspace* după simulare, este o structură care conține timpul de simulare (i.e. SimOut.tout) și nivelul tancului de stocare (i.e. SimOut.yout{1}.Values.Data).

Pentru reprezentarea grafică a debitului volumetric de ieșire s-a adăugat un bloc *To Workspace* (Figura 3), care face ca datele să fie disponibile în *Workspace*, în aceeași structură SimOut (i.e. SimOut.Fout.Data).

Se poate realiza astfel reprezentarea grafică a nivelului tancului de stocare și a debitul de ieșire, folosind următoarea instrucțiune:

```
%% Reprezentare grafica

figure(1)
subplot(211)
plot(SimOut.tout,SimOut.yout{1}.Values.Data); hold on
plot([0 StopTime],[5 5],'r--'); ylim([0 6])
xlabel('Timp (h)'); ylabel('Nivel (m)'); grid; grid minor
legend('Nivel (m)','Nivel maxim (m)','Location','southeast')
subplot(212)
plot(SimOut.tout,SimOut.Fout.Data); hold on
xlabel('Timp (h)'); ylabel('Debit iesire (m3/h)'); grid; grid minor
```

Pentru a simplifica reprezentarea modelului se vor selecta blocurile așa cum se poate vedea în Figura 4, click dreapta pe selecție apoi se alege *Create Subsystem from Selection*. După crearea subsistemului va fi necesară redenumirea acestuia și a blocurilor *Input port* și *Output port* din interiorul acestuia.

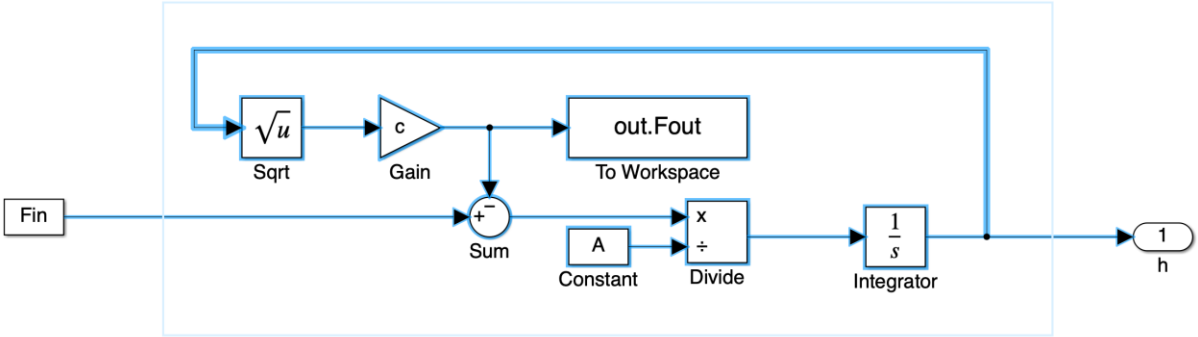


Figura 4. Crearea unui subsistem pentru tancul de stocare

Rezultatul va fi un subsistem compact care conține modelul matematic care dă nivelul tancului de stocare.

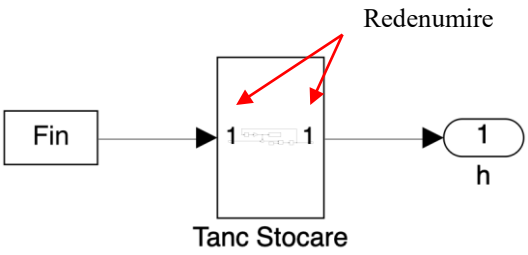


Figura 5a. Subsistemul tancului de stocare

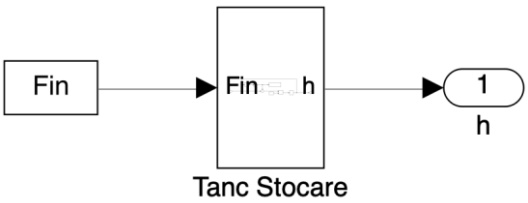


Figura 5b. Subsistemul tancului de stocare după redenumirea intrării și ieșirii

După simularea modelului s-au obținut următoarele rezultatele grafice din Figura 6. La acest debit de intrare, $F_{in}$ = 2 m$^3$/h, nivelul tancului va depăși valoarea maximă admisă $h_{max}$ = 5 m. Se impune deci controlul în buclă închisă a nivelului tancului de stocare pentru a evita deversarea.

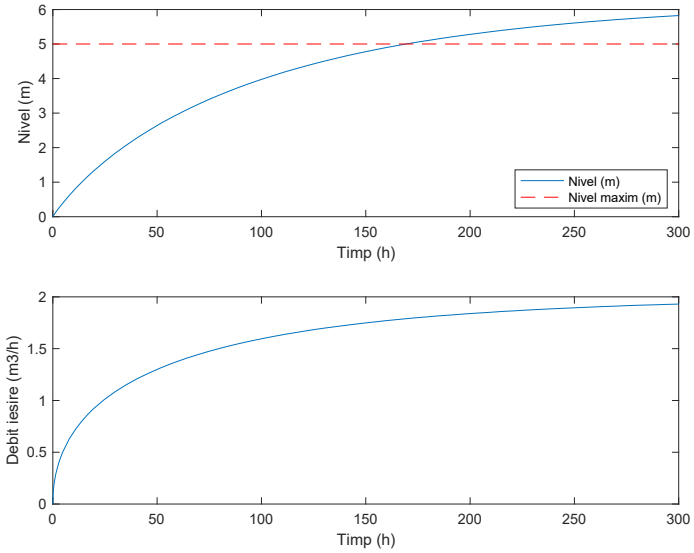


Figura 6. Evoluția în timp a nivelului și a debitului de ieșire

Implementarea regulatorului de nivel s-a realizat așa cum poate fi văzut în Figura 7. S-a adăugat un bloc *Manual Switch* pentru a compara ușor simulările cu și fără regulator. Pentru a acționa blocul *Manual Switch* din fișierul script se va adăuga următoarea instrucțiune la definirea obiectului SimulationInput (1 – Automat, 0 – Manual).

```
Regulator = 0;      % 1 - cu regulator;     0 - fara regulator
SimIn = SimIn.setBlockParameter("TancStocare/Manual Switch","sw",num2str(Regulator));
```

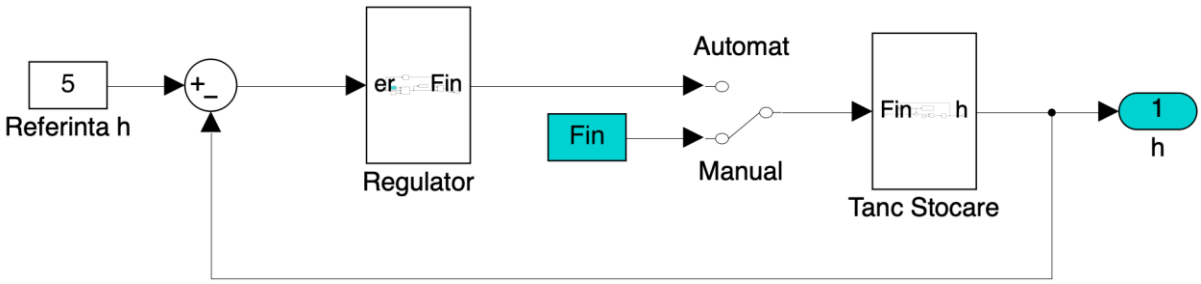


Figura 7. Regulatorul de nivel al tancului de stocare

Regulatorul de nivel ON/OFF constă în adăugarea unui bloc *Switch* care lasă să treacă valoarea $F_{in} = 2$ dacă eroarea este $> 0$ și $F_{in} = 0$ în caz contrar. Dacă apar mesaje de eroare trebuie accesate setările solverului și selectat *Adaptive* la *Signal Threshold*.

Se poate selecta o metodă de integrare cu pas fix și se poate specifica pasul de integrare.

Regulatorul de nivel cu histerezis poate fi implementat printr-un singur bloc *Relay*.

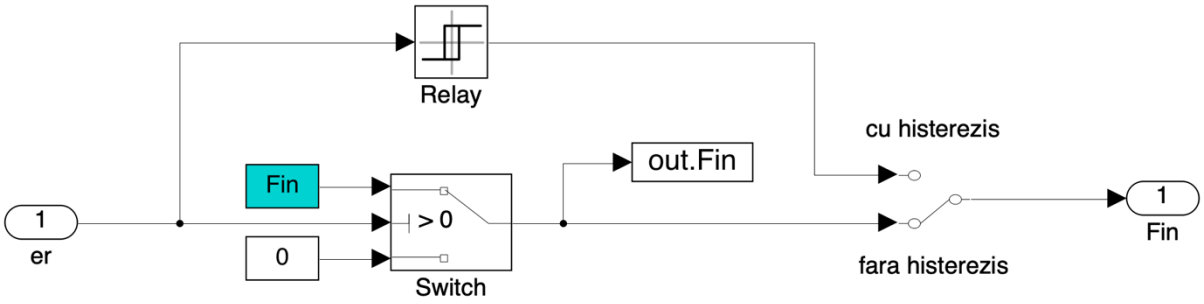


Figura 8. Subsistemul *Regulator*

Pentru a face comparația între cele doua regulatoare s-a adăugat un bloc *Manual Switch* iar pentru a putea reprezenta grafic și debitul de intrare s-a folosit un bloc *To Workspace*.

```
Histerezis = 0;     % 1 - cu histerezis;    0 - fara histerezis
SimIn = SimIn.setBlockParameter("TancStocare/Regulator/Manual Switch","sw",num2str(Histerezis));
```

Pentru reprezentarea grafică a debitului de intrare $F_{in}$ s-au adăugat următoarele instrucțiuni:

```
if Regulator == 0
   plot(SimOut.tout,Fin*ones(1,length(SimOut.tout)),'o-')
else
   stairs(SimOut.tout,SimOut.Fin.Data,'o-')
end

legend('Debit iesire (m^3/h)','Debit intrare (m^3/h)','Location','southwest')
```

**Aplicația 2: Dezvoltarea unui model dinamic de bilanț pentru un proces de tratare biologică aerobă a apelor uzate**

**Aspecte teoretice**

*Modelul dinamic general pentru procesele continue*. Un proces continuu este acel proces în care mediul proaspăt de cultură este adăugat în mod continuu în reactor și astfel, $F_{in} = F_{out} = F$ (figura 1). Într-un sistem continuu, volumul este constant și, prin substituirea lui $D = F/V$ și împărțirea la $V$ în ambele părți ale modelului general (2.18), se obține:

$$\frac{dc_i}{dt} = r_i + Dc_{i,0} - Dc_i \tag{1}$$

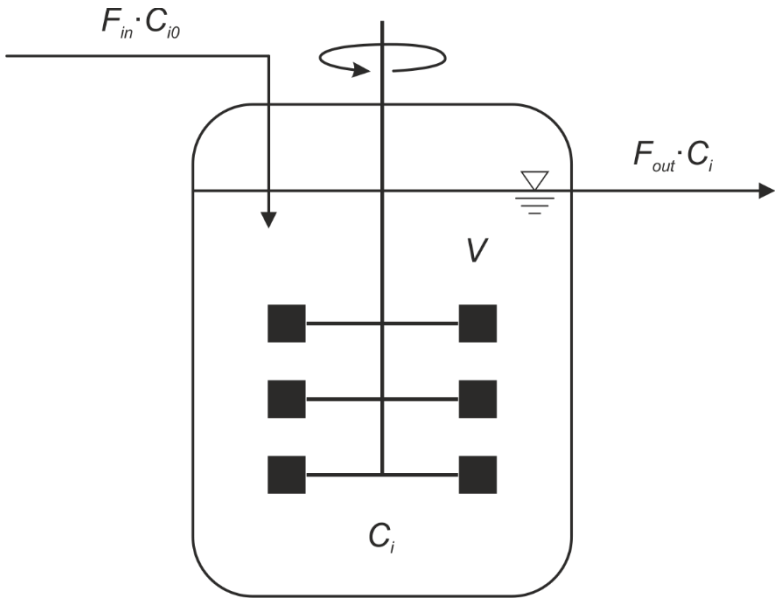


Figura 1. Bioreactor continuu cu amestecare completă

Modelul dinamic general 2.18 descrie un proces care nu conține fluxuri difuzive. Atunci când vine vorba de procese biotehnologice, prezența unui flux difuziv se justifică doar pentru gaze care se pot dizolva în apă și care participă la proces. Opțiunile sunt destul de limitate, cel mai des modelându-se dinamica oxigenului dizolvat, iar uneori cea a dioxidului de carbon dizolvat. La modelarea gazelor dizolvate, modelul dinamic general este îmbogățit cu un termen care descrie rata de transfer masic gaz-lichid (ecuația 2.15). Dacă rescriem modelul pentru un proces continuu care se desfășoară într-un reactor cu amestecare completă, cu un singur flux convectiv, ecuația 1 va deveni:

$$\frac{dc_i}{dt} = r_i + Dc_{i,0} - Dc_i + N_i \tag{2}$$

Termenul care descrie rata de transfer masic gaz-lichid, $N_i$, va fi mereu pozitiv (sau 0) pentru că se transferă masă dinspre gaz spre lichid.

**Modelul dinamic de bilanț de masă**

Modelul de bilanț de masă regrupează termenii care descriu ratele cinetice (de creștere a nămolului activat și de consum a substratului și oxigenului dizolvat) și termenii de transport și are următoarea formă (Ifrim, 2012):

$$\frac{dX_{\mathrm{B}}}{dt} = r_{X_{\mathrm{B}}} - DX_{\mathrm{B}} + rDX_{\mathrm{R}} - rDX_{\mathrm{B}} \tag{3}$$

$$\frac{dS_{\mathrm{S}}}{dt} = r_{S_{\mathrm{S}}} + DS_{\mathrm{S,in}} - DS_{\mathrm{S}} \tag{4}$$

$$\frac{dS_{\mathrm{O}}}{dt} = r_{S_{\mathrm{O}}} \cdot 10^3 + DS_{\mathrm{O,in}} - DS_{\mathrm{O}} + N_{O_2} \tag{5}$$

$$\frac{dX_{\mathrm{R}}}{dt} = D_s X_{\mathrm{B}} + rD_s X_{\mathrm{B}} - rD_s X_{\mathrm{R}} - \beta D_s X_{\mathrm{R}} - \eta D_s (1-\beta) X_{\mathrm{R}} \tag{6}$$

unde $D$ – diluția bazinului aerob, $D_s$ – diluția stratului din decantor, $r$ – rata de recirculare și $\beta$ – rata de nămol în exces au următoarea expresie:

$$D = \frac{F_f}{V}; \qquad D_s = \frac{F_f}{V_s} = D\frac{V}{V_s}; \qquad r = \frac{F_r}{F_f}; \qquad \beta = \frac{F_e}{F_f};$$

Ratele volumetrice de creștere a nămolului activat, $r_{X_{\mathrm{B}}}$, de consum a substratului, $r_{S_{\mathrm{S}}}$, și de consum a oxigenului dizolvat, $r_{S_{\mathrm{O}}}$, sunt date de ecuațiile 3.4, 3.5 și 3.6, iar rata de transfer masic gaz-lichid a oxigenului, $N_{O_2}$, este calculată cu ecuațiile 3.7 și 3.8. $r$ este coeficientul de recirculare și reprezintă raportul dintre debitul de recirculare și debitul de alimentare, iar $\beta$ este coeficientul de exces de nămol care reprezintă raportul dintre debitul de exces și debitul de alimentare. $S_{\mathrm{S,in}}$ și $S_{\mathrm{O,in}}$ reprezintă concentrațiile de substanțe organice poluante (substrat) și de oxigen dizolvat din influent.

**Cerințe:**

Să se construiască un model Simulink și să simuleze evoluția tuturor variabilelor de stare ale procesului pe o perioadă de 300 de ore ($t_f = 300$). Condițiile inițiale sunt: $x(t_0) = [X_B(t_0) \quad S_S(t_0) \quad S_O(t_0) \quad X_R(t_0)] = [0.5 \quad 0.8 \quad 2 \quad 0]$.

- Să construiască un fișier script care să apeleze modelul Simulink și să se reprezinte grafic toate cele 4 variabile de stare ale procesului.

**Implementare în Matlab/Simulink**

Vom începe prin construirea subsistemului pentru viteza specifică de creștere. Un exemplu de implementare poate fi găsit în figura 1.

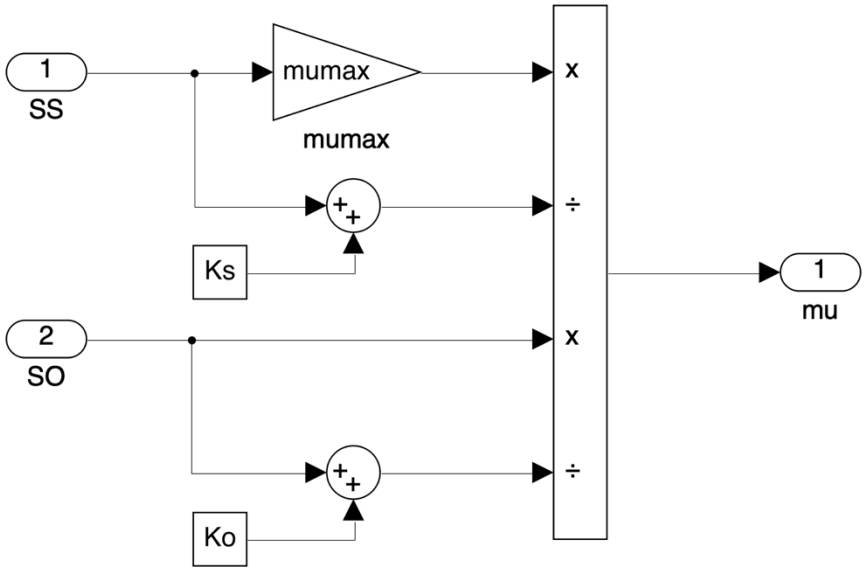


Figura 1. Subsistemul vitezei specifice de creștere

Acest subsistem va fi folosit pentru construirea modelului pentru bazinul aerob. În figura 2 găsim un exemplu de implementare a celor 3 variabile de stare care caracterizează bazinul aerob: $X_B$, $S_S$ și $S_O$.

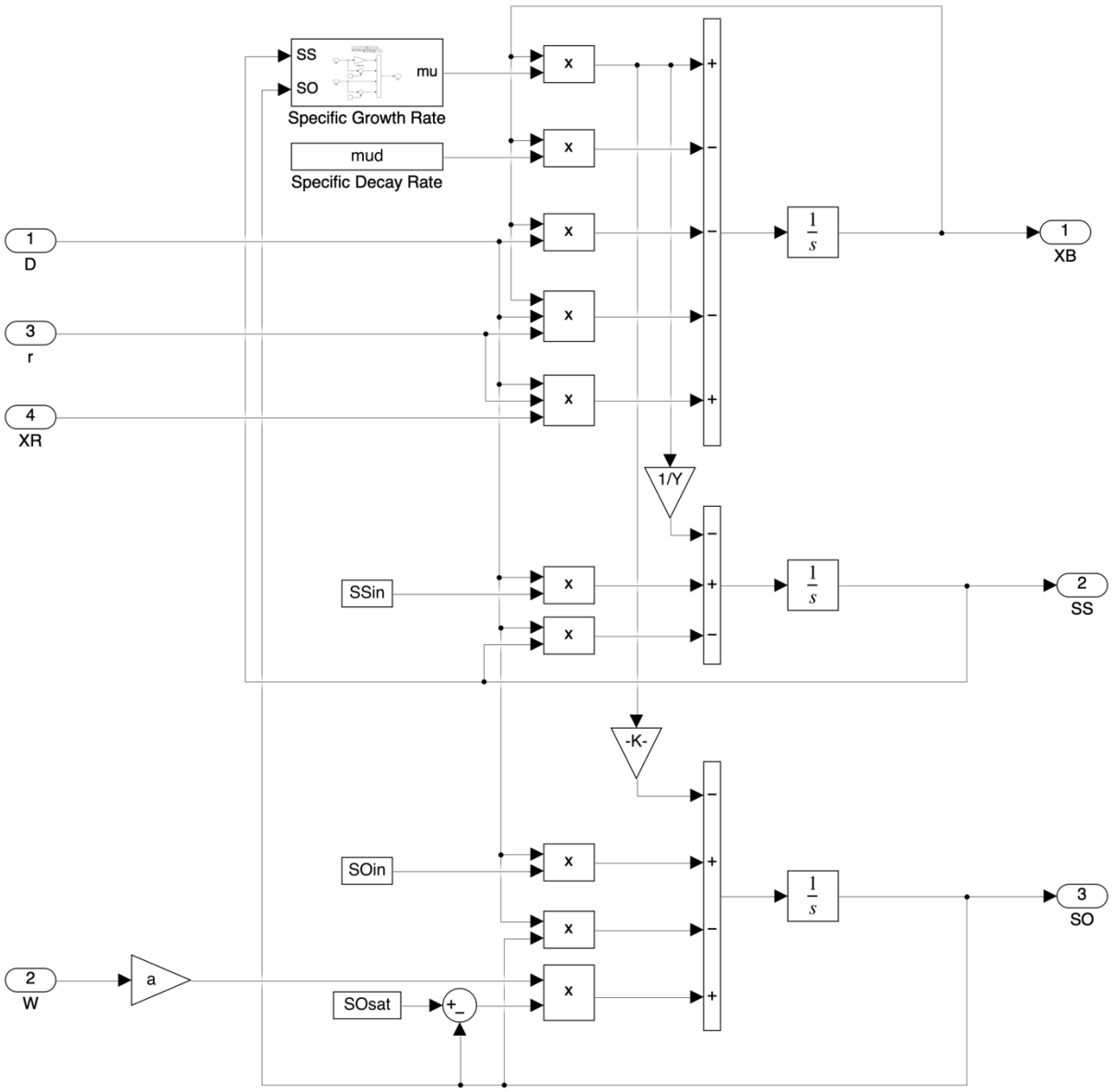


Figura 2. Subsistemul bazinului aerob

Decantorul se va implementa într-un subsistem separat așa cum poate fi văzut în figura 3.

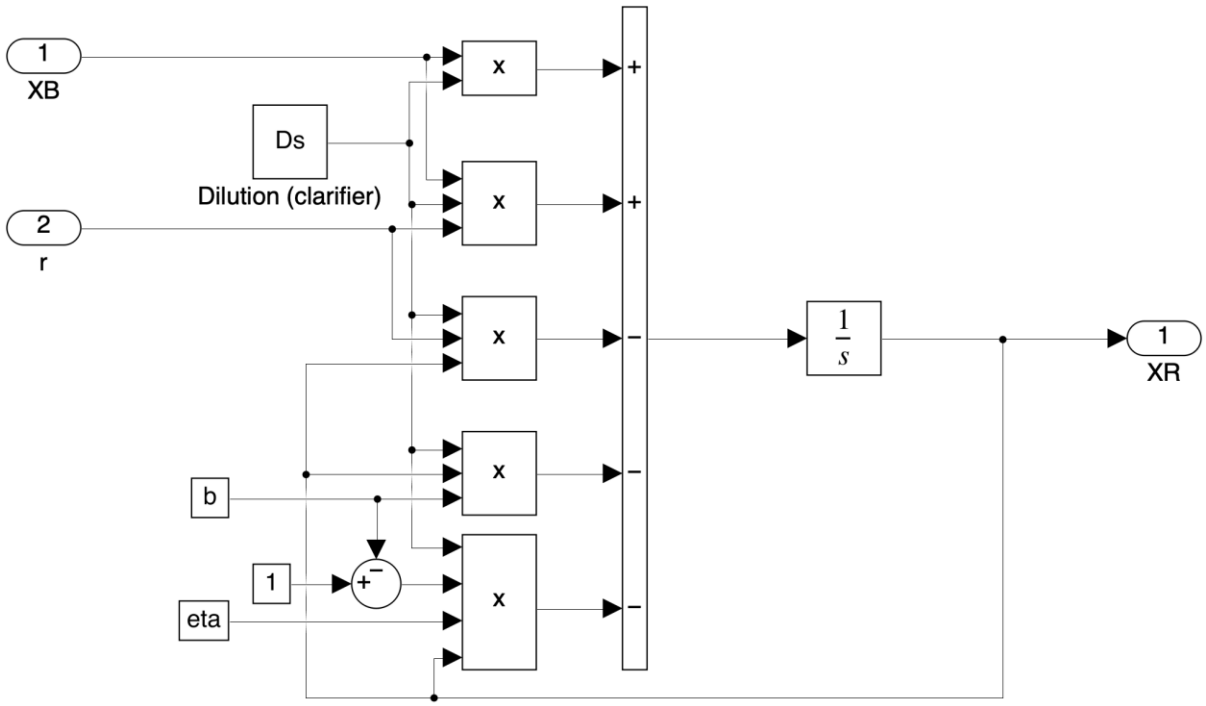


Figura 3. Subsistemul decantorului

Cele două subsisteme vor putea fi cuplate așa cum poate fi văzut în figura 4.

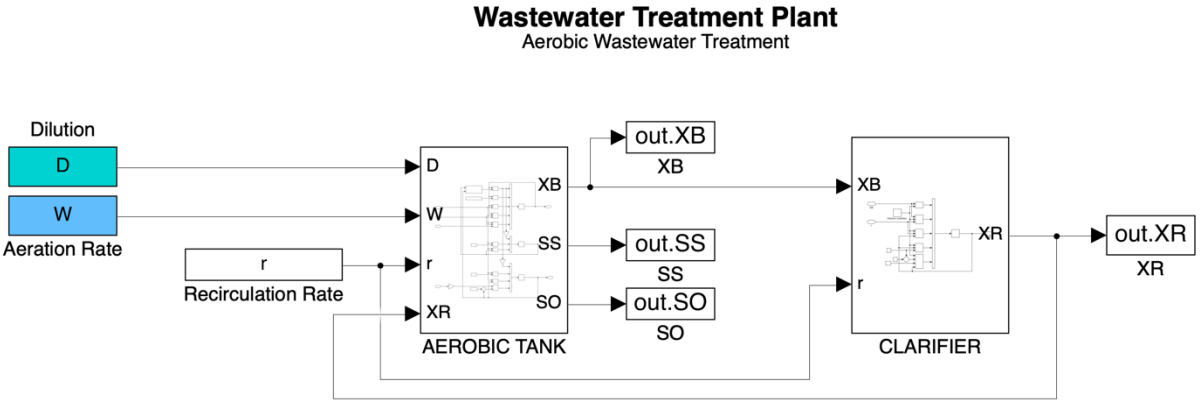


Figura 4. Modelul matematic al unui proces de tratare aerobă a apelor uzate

**Parametrii modelului și variabilele de intrare** vor fi scrise într-un fișier script

```
%% Parametri model

mumax = 0.21;   % Viteza specifica max. de crestere [h-1]
mud = 0.02;     % Viteza specifica de declin [h-1]
Ks = 0.18;      % Const. de semisaturatie [g.L-1]
Ko = 0.2;       % Const. de saturatie pentru OD [mg.L-1]
Y = 0.67;       % Randament de conversie substrat [-]

a = 0.0033;     % coeficient de difuzie
SOsat = 8;      % Conc. de OD la saturatie [mg.L-1]

r = 1;          % Raport dintre Fr si Ff [-]
```

```
b = 0.2;        % Raport dintre Fe si Ff [-]
V = 35;         % Volumul bazinului aerob [m3]
Vs = 6;         % Volumul stratului de namol [m3]
eta = 0.25;

%% Variabile intrare

D = 0.03;       % Dilutia bazinului aerob [h-1]
Ds = D*V/Vs;    % Dilutia stratului de namol [h-1]

W = 2100;         % Rata de aerare [l.min-1]

SSin = 4;       % Conc. substratului in influent [g.L-1]
SOin = 2;       % Concentratia OD in influent [g.L-1]
```

**Apelarea modelului Simulink** va putea fi făcută cu funcția Matlab sim

```
%% Apelare model Simulink

StopTime = 300;

SimIn = Simulink.SimulationInput('Apauzata');
SimIn = SimIn.setModelParameter("StopTime",num2str(StopTime));

SimOut = sim(SimIn);                                    % Simulates the model
```

**Reprezentarea grafică**

```
%% Reprezentare grafica

figure(1)
subplot(221)
plot(SimOut.tout,SimOut.XB.Data)
xlabel('Time (Hours)'); ylabel('Namol activat (g/L)')
grid; grid minor; hold on
subplot(222)
plot(SimOut.tout,SimOut.SS.Data)
xlabel('Time (Hours)'); ylabel('Substrat (g/L)')
grid; grid minor; hold on
subplot(223)
plot(SimOut.tout,SimOut.SO.Data)
xlabel('Time (Hours)'); ylabel('Oxigen dizolvat (mg/L)')
grid; grid minor; hold on
subplot(224)
plot(SimOut.tout,SimOut.XR.Data)
xlabel('Time (Hours)'); ylabel('Namol recirculat (g/L)')
grid; grid minor; hold on
```

În urma simulării modelului matematic s-a obținut următoarele rezultate:

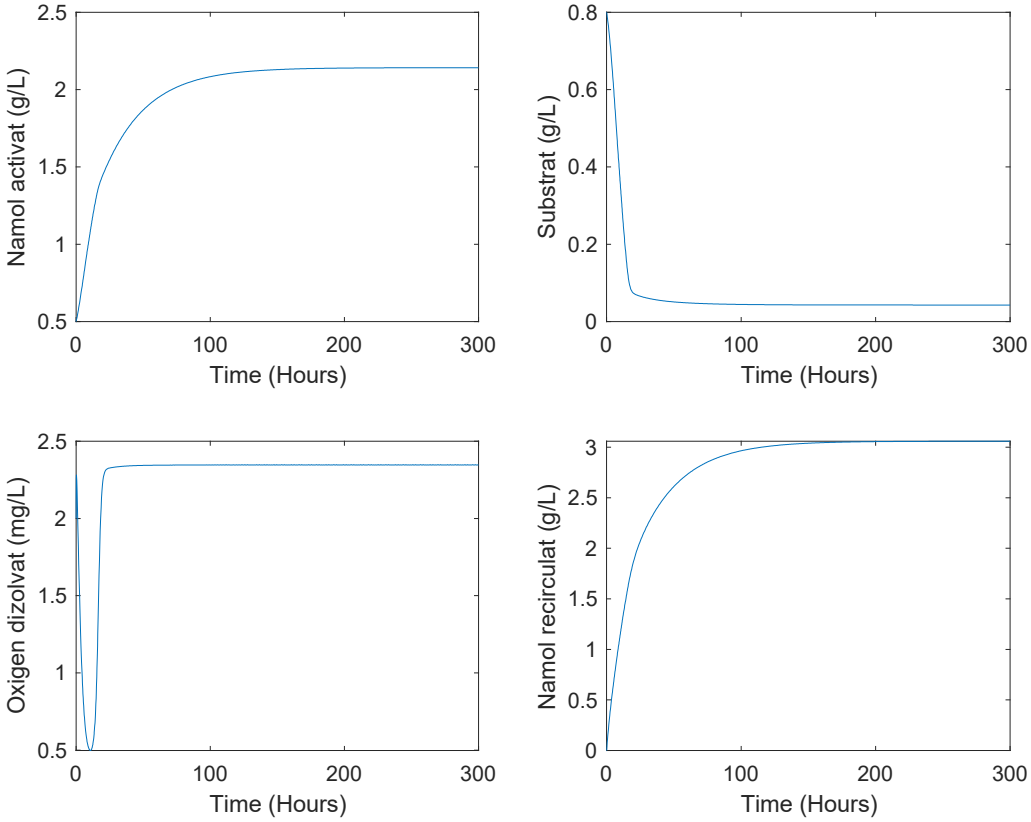


Figura 5. Rezultatele obținute în simulare

**Aplicația 3: Controlul oxigenului dizolvat într-un proces de tratare biologică aerobă a apelor uzate**

Controlul oxigenului dizolvat este esențial pentru orice proces aerob, și cu atât mai mult pentru tratarea biologică aerobă a apelor uzate. Oxigenul dizolvat $S_O$ se măsoară cu senzori electrochimici sau optici și se controlează cu ajutorul ratei de aerare, $W$. Bucla de oxigen dizolvat trebuie să existe indiferent de ce alt tip de control este implementat în instalație. Un bun control al oxigenului dizolvat poate duce la o eficiență a eliminării substanțelor organice poluante cu până la 10%.

**Cerințe:**

Să se implementeze un sistem de control în buclă închisă pentru oxigenul dizolvat folosind modelul matematic al procesului de tratare biologică a apelor uzate dezvoltat la aplicația anterioară.

- Să se implementeze un regulator PI care să fie acordat prin metoda "trial and error". Valoarea de referință pentru oxigenul dizolvat este $S_O = 2$ mg/L.
- Să se reprezinte grafic rata de aerare și oxigenul dizolvat.

**Implementare în Matlab/Simulink**

Primul pas va fi înglobarea întregului model al stației de tratare aerobe într-un subsistem care are la intrare rata de aerare $W$ și la ieșire oxigenul dizolvat $S_O$.

În jurul acestui subsistem se va construi schema de control în buclă închisă așa cum poate fi văzut in figura 1.

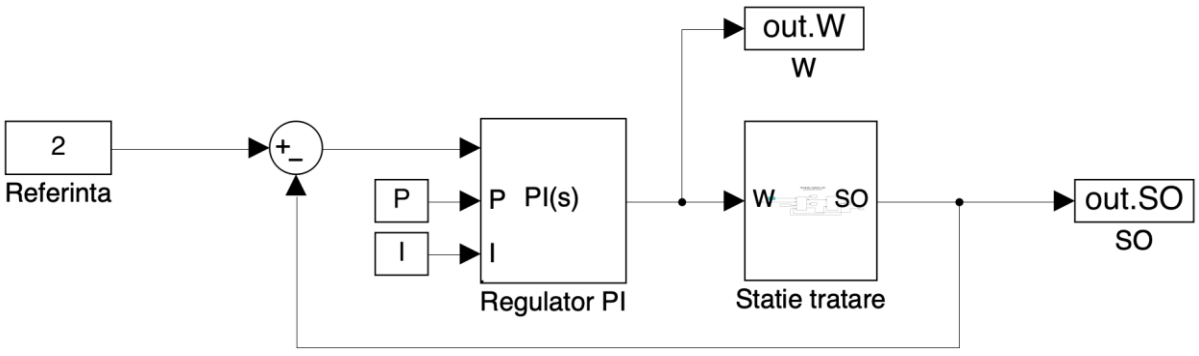


Figura 1. Schema de control în buclă închisă a oxigenului dizolvat

Fișierului script i-au fost adăugați parametrii regulatorului PI

```
%% Parametri regulator

P = 1;
I = 1;
```

Pentru reprezentarea grafică a variabilei de intrare $W$, s-a adăugat un bloc *To Workspace*, iar fișierului script următoarele instrucțiuni

```
figure(2)
subplot(211)
plot(SimOut.tout,SimOut.W.Data)
xlabel('Time (Hours)'); ylabel('Rata de aerare (L/h)')
grid; grid minor; hold on
subplot(212)
plot(SimOut.tout,SimOut.SO.Data)
xlabel('Time (Hours)'); ylabel('Oxigen dizolvat (mg/L)')
grid; grid minor; hold on
```

## Bibliografie

Dochain D. Automatic Control of Bioprocesses. John Wiley & Sons, Inc. Great Britain, 2008.

Doran MP. Bioprocess Engineering Principles. Academic Press, 1995.

Gray NF. Water Technology. An Introduction for Environmental Scientists and Engineers. Second Edition. Elsevier, Oxford. 2005.

Ifrim GA. Comanda proceselor de interes pentru mediu (tratarea biologică a apelor uzate și creșterea microalgelor în fotobioreactor). Teza de doctorat. Universitatea „Dunărea de Jos" din Galați, 2012.

Kravaris, C., & Kookos, I. (2021). Understanding Process Dynamics and Control (Cambridge Series in Chemical Engineering). Cambridge: Cambridge University Press. doi:10.1017/9781139565080

Sablani SS, Shafiur R, Datta AK and Majumdar AS. Handbook of Food and Bioprocess Modeling Techniques. Taylor & Francis, USA, 2006.

Snape JB, Dunn IJ, Ingham J, Prenosil JE. Dynamics of Environmental Bioprocesses. Modeling and Simulation. VCH, 1995.

# SECURITATEA CIBERNETICĂ

**- note de curs –**

## I. Introducere

Din moment ce rețeaua globală de comunicații digitale a devenit parte integrantă din viața noastră pe toate palierele, securitatea cibernetică este esențială pentru protejarea datelor, a societății, a statului și pentru asigurarea stabilității economice.

Securitatea cibernetică se referă la abilitatea de a proteja sau apăra spațiul cibernetic de atacurile cibernetice[1]. Cu alte cuvinte, aceasta privește efortul de protejare continuă a indivizilor, organizațiilor și guvernelor de atacurile digitale care vizează utilizarea neautorizată sau distrugerea sistemelor critice, a rețelelor, a aplicațiilor software și datelor sensibile ale acestora. [2]

Trebuie accentuat faptul că protecția identității, datelor și dispozitivele electronice este importantă atât la nivel personal, cât și profesional. La nivel organizațional, este responsabilitatea tuturor angajaților și partenerilor să asigure păstrarea reputației, să protejeze datele și clienții companiei pentru care lucrează. La nivel de stat, se află în joc securitatea națională, siguranța și bunăstarea tuturor cetățenilor.

E un fapt cunoscut că amenințările de securitate cibernetică sunt în creștere iar infractorii cibernetici caută să exploateze orice vulnerabilitate pe care o pot găsi pentru a fura informații sau bani. Problemele legate de securitate cibernetică și atacurile asupra infrastructurii critice se află în **top 5 riscuri globale** conform *World Economic Forum Global Risk Report* (2023)[3].

Atacurile cibernetice cresc cu fiecare nouă conexiune digitală realizată în lume. Prin urmare, profesioniștii din domeniul securității cibernetice ce pot proteja și apăra rețeaua unei organizații sunt la mare căutare în acest moment. În topul celor mai căutate și mai bine plătite locuri de muncă în domeniul securității cibernetice se pot enumera[4]: Director de Securitate a Informațiilor (*Chief Information Security Officer* – CISO: 108.000 – 233,000$), Manager de Securitate a Informațiilor (*Information Security Manager*: 82.000 – 156.000$), Arhitect de securitate (*Security* Architect: 87.000 – 158.000$), Inginer de Securitate a Rețelei (*Network Security Engineer*: 63.000 – 140.000$) ș.a.

## II. Scopurile securității cibernetice

Scopurile securității cibernetice pot fi rezumate în următoarele principii cheie, cuprinse în ghidul pentru securitatea informațiilor numit **Triada CIA** (*Confidentiality, Integrity, Availability*):

- **Confidențialitatea** – previne dezvăluirea informațiilor către utilizatori sau procese neautorizate, incluzând mijloacele de protejare a informațiilor confidențiale și a celor proprietare. Există mai multe metode și tehnologii ce pot fi utlizate pentru a asigura confidențialitatea datelor și pentru a îmbunătăți intimitatea: criptarea datelor, controlul accesului, anonimizarea, *token*-izarea,
- **Integritatea** – previne modificarea neautorizată sau distrugerea informațiilor și se referă la autenticitate și exactitate, consistență și încredere în dispozitive, rețele, aplicații, informații și

[1] Cyber Security - Glossary | CSRC (nist.gov) (Aprilie, 2023)
[2] Cisco Skills For All - Introduction to Cybersecurity (Aprilie, 2023)
[3] WEF_Global_Risks_Report_2023.pdf (weforum.org) (Septembrie, 2023)
[4] Top 10 Cybersecurity Jobs Boost Your Career In 2023 (thecyberexpress.com) (Septembrie, 2023)

date, pe întreg ciclul lor de viață. Metodele utilizate pentru asigurarea integrității datelor includ: utilizarea *hashing*, verificarea validării datelor, verificări de consistențe a datelor și controlul accesului. Sistemele de integritate a datelor pot include una sau mai multe din aceste metode.

- **Disponibilitate** – se asigură că sistemele, rețelele, informațiile și datele sunt disponibile atunci când sunt necesare utilizatorilor sau proceselor autorizate. Metodele utilizate pentru a asigura disponibilitatea includ: redundanța sistemelor, backup-urile sistemelor, creșterea rezistenței sistemelor, mentenanța echipamentelor, actualizările sistemelor de operare și a aplicațiilor, planuri proactive de revenire după dezastre neprevăzute.

Protecția datelor trebuie asigurată pe întreg ciclul lor de viață:

- **în repaus**, atunci când sunt stocate și niciun utilizator sau proces nu le accesează, solicită sau modifică;
- **în tranzit**, atunci când datele sunt transmise între dispozitivele conectate în rețea;
- **în timpul procesării**, atunci când datele sunt introduse, modificate manual sau automat, prin procesare sau atunci afișate.

## III. Categorii de infractori cibernetici

Amenințările de securitate sunt multiple și pot fi încadrate în diverse categorii. Această clasificare permite organizațiilor să le evalueze și să estimeze impactul pe care îl pot avea, pentru a le prioritiza și a putea organiza apărarea împotriva lor.

În lista amenințărilor de securitate sunt incluse: atacurile software cum ar fi atacurile DDoS sau virusurile pentru computere, erorile (bug-urile) software care permit partajarea ilegală de fișiere sau aplicații funcționând off-line, erorile și defecțiunile echipamentelor hardware, erorile umane la introducerea datelor, manipularea echipamentelor sau dezvăluirea de informații confidențiale, sabotajul, furtul de informații sau echipamente din locațiile nesupravegheate, întreruperi ale furnizării utilităților cum ar fi căderile de tensiune și, nu în ultimul rând, dezastre naturale ca inundațiile, incendiile, cutremurele sau vreme severă (furtună, uragan, tornadă).

Sursa amenințărilor poate fi atât **internă** cât și **externă**, amenințările interne putând determina distrugeri care să le depășească pe cele pricinuite de amenințările externe, din cauza posibilității accesului direct la dispozitivele și rețeaua organizației și, de asemenea, din cauză că atacatorii posedă informații din interior privind resursele și datele confidențiale precum și măsurile de securitate implementate în organizație.

**Amenințările interne** pot proveni de la angajați și de la alte persoane contractuale, atât curenți cât și foști, care utilizează în mod inadecvat resursele și datele organizației, distrugându-le sau alterându-le în mod accidental sau intenționat, configurând în mod eronat echipamentele sau aplicațiile. Aceștia pot afecta operarea serverelor, infrastructura și echipamentele de rețea, pot conecta medii de stocare infectate sau pot accesa email-uri sau site-uri web malițioase.

**Amenințările externe** provin de obicei de la atacatori sau grupuri de atacatori, amatori sau foarte bine instruiți care au capacitatea și instrumentele necesare exploatării vulnerabilităților echipamentelor, rețelei și aplicațiilor software utilizate de organizație sau care pot obține acces la acestea utilizând tehnici de inginerie socială.

Amenințările persistente avansate (APT – *Advanced Persistent Threat*) reprezintă un atac continuu, complex care utilizează tactici de spionaj elaborate, malware sofisticat și implică atacatori multipli pentru a obține acces, a analiza și exploata rețeaua, dispozitivele și datele țintei. APT sunt de obicei bine-orchestrate, bine-finanțate, rămân nedetectate perioade lungi de timp și au consecințe devastatoare.

În figura 1 sunt prezentate principalele tipuri de infractori cibernetici, grupați în categorii după sursa amenințărilor și nivelul de pregătire. Infractorii amatori, numiți și ***script kiddies*** – posedă puține abilități și folosesc instrumente sau instrucțiuni găsite pe internet pentru a exploata vulnerabilitățile unei rețele și a obține câștiguri financiare sau personale. Chiar dacă nivelul atacurilor nu este întotdeauna suficient de sofisticat, rezultatele pot fi devastatoare.

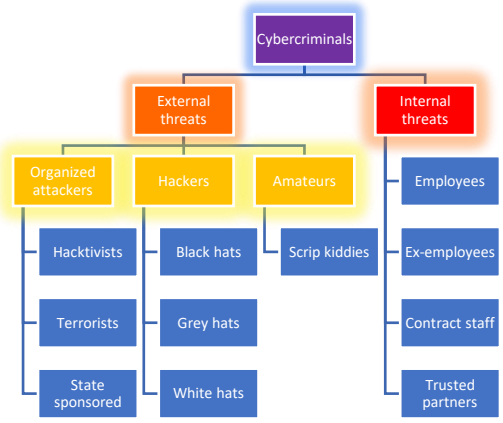


***Fig. 1*** Tipuri de infractori cibernetici

O altă categorie de atacatori – ***hackeri*** – sunt bine instruiți și informați, atacând rețelele sau computerele, exploatându-le vulnerabilitățile sau folosind tehnici de inginerie socială, acționând individual sau organizați în grupuri și având diferite obiective:

- hackeri ***white hat*** – penetrează, cu permisiunea proprietarului, rețelele și sistemele pentru a îmbunătăți securitatea acestora; se mai numesc hackeri etici;
- hackeri ***gray hat*** – se comportă ca hackeri *white hat* sau *black hat*, în funcție de interesul lor; fie raportează proprietarilor sistemelor vulnerabilitățile identificate, fie le publică pe internet pentru a fi exploatate de alți atacatori;
- hackeri ***black hat*** – profită de vulnerabilități pentru câștiguri ilegale personale, financiare sau politice
- hackeri organizați
    - **infractori cibernetici** – sunt grupuri de infractori profesioniști car are ca scop obținerea de control, putere și bani
    - ***hacktiviști*** – au ca scop sensibilizarea publicului în legătură cu anumite probleme considerate importante
    - **teroriști** – infractori care folosesc tehnologia pentru a realiza atacuri premeditate având scopul de a răspândi frică și a produce perturbări fizice, sociale, politice sau economice;

- grupuri sponsorizate la nivel statal – efectuează atacuri cibernetice în beneficiul guvernelor care-i susțin financiar și logistic.

Internetul a devenit și o zonă în care se desfășoară conflictele între națiuni, așa numitul război cibernetic (***cyberwarfare***). Luptele care au loc în spațiul cibernetic presupun penetrarea sistemelor informatice și a rețelelor altor națiuni pentru a sabota și provoca daune, întreruperea unor servicii și pentru a spiona la nivel industrial sau militar. Războiul cibernetic poate destabiliza națiunile, poate afecta relațiile comerciale, poate diminua încrederea populației în guvern și autoritățile statului, fără a fi nevoie de invazie militară.

## IV. Categorii de atacuri cibernetice

Majoritatea atacurilor cibernetice se desfășoară pe mai multe planuri, combinând mai multe tehnici pentru a putea compromite și a exploata un sistem, o rețea sau o aplicație. Atacatorii exploatează vulnerabilitățile software și hardware, utilizează diverse tipuri de software malițios (malware) și se bazează pe diverse metode de infiltrare, înșelăciune, manipulare și fraudare bazate pe ingineria socială.

Printre cele mai întâlnite tipuri de atacuri cibernetice se regăsesc:

- utilizarea de ***malware***, software malițios care poate fi utilizat pentru a fura sau distruge date, pentru a ocoli măsurile de control al accesului sau pentru a compromite sau vătămă un dispozitiv, o rețea sua o aplicație:
- atacuri ***man-in-the-middle***, pentru a intercepta, modifica și transmite informații false între două dispozitive, cu scopul de a fura date sau de a uzurpa identitatea unuia din dispozitive;
- atacuri *zero-day* care exploatează vulnerabilitățile software ale aplicațiilor înainte de a deveni cunoscute publicului și a fi corectate;
- atacuri de *spoofing* prin falsificarea adreselor MAC sau IP pentru a deghiza dispozitivul atacatorului ca fiind unul valid;
- atacuri de *flooding* pentru compromiterea echipamentelor intermediare din rețea prin inundarea cu adresa MAC false;
- atacuri *denial of service* prin care se trimit volume mari de date la o rată pe care sistemul sau rețeaua țintă nu le poate gestiona, rezultând încetinirea sau blocarea acestora;

tehnici de înșelăciune, manipulare sau păcălire a utilizatorilor pentru a-i determina să efectueze anumite acțiuni sau să divulge informații sensibile.

### 4.1. Tipuri de malware

Principalele tipuri de malware utilizate pentru a fura datele și a ocoli mecanismele de control al accesului, pentru a provoca daune sau pentru a compromite sistemele și rețelele sunt următoarele:

- **virus** – cod executabil care se replică și se atașează de alte fișiere executabile pentru a modifica, fura sau șterge date; se răspândesc prin intermediul dispozitivelor de stocare (USB, CD, DVD), partajărilor de resurse în rețea sau prin email.
- **vierme** – cod care nu are nevoie de un fișier gazdă pentru a fi executat, care exploatează vulnerabilitățile rețelei pentru a se replica și răspândi foarte repede conducând la încetinirea sau blocarea rețelei;
- **troian** – malware care execută operații maligne disimulate sub umbrela unor acțiuni valide, atașându-se de fișiere ne-executabile și exploatând privilegiile utilizatorului;

- ***ransomware*** – malware proiectat să blocheze, de obicei pin criptare, funcționarea unui sistem informatic sau datele stocate pe acesta și amenință cu menținerea blocajului sau cu ștergerea datelor până la efectuarea unei plăți către atacator; efectuarea plății nu garantează deblocarea sau decriptarea datelor;
- ***spyware*** – acționează ca parte a unui software legitim sau a unui troian, ocolind sistemul de securitate pentru a urmări și spiona utilizatorul;
- ***adware*** – malware conceput să livreze anunțuri și reclame în mod automat, putând fi însoțit și de un program *spyware*;
- ***scareware*** – software care exploatează frica utilizatorului, determinându-l să efectueze acțiuni care să permită instalarea de alte tipuri de malware.

Indiferent de tipul de malware cu care a fost infectat un sistem informatic sau o rețea, pot fi recunoscute o serie de simptome comune cum ar fi: creșterea utilizării procesoarelor, scăderea vitezei de lucru, blocarea sau oprirea sistemului, modificarea, ștergerea sau criptarea fișierelor, prezența unor fișiere sau aplicații necunoscute, executarea unor procese necunoscute, trimiterea de mesaje fără acordul utilizatorului.

### 4.2. Tipuri de înșelăciuni

Unul din cele mai eficiente și, uneori, mai simple moduri prin care atacatorii pot obține informații despre sistemele informatice și rețeaua unei organizații se bazează pe înșelătorie, manipulând victima astfel încât aceasta să efectueze anumite acțiuni (inginerie socială), urmărind acțiunile victimei în timp ce introduce codurile PIN sau parolele, din apropiere sau cu ajutorul diverselor dispozitive de vedere la distanță (*shoulder surfing*), căutând informații în documentele aruncate, fără a fi distruse, în coșul de gunoi (*dumpster diving*) pretinzând a fi o altă persoană (*impersonating*) sau inducând victima în eroare, păcălind-o pentru a crea frică și comportament irațional (*hoax*).

Indiferent de cât de bune sunt măsurile de securitate, de cât de corect sunt configurate echipamentele, serverele și sistemele din rețea și cât de restrictive sunt tehnicile de control al accesului, tehnicile de înșelăciune nu pot fi prevenite fără o conștientizare și o instruire adecvată a utilizatorilor.

### 4.3. Tipuri și tactici de inginerie socială

Ingineria socială exploatează natura umană, se bazează pe sociabilitatea oamenilor, pe dorința acestora de a fi de ajutor, pe încrederea implicită pe care o au oamenii unii în alții sau profită de slăbiciunile acestora pentru a-i manipula și determina să efectueze anumite acțiuni sau să divulge informații sensibile, confidențiale.

Există mai multe tipuri de atacuri bazate pe inginerie socială, dintre care se pot menționa:

- pretextul (*pretexting*) – în care atacatorul minte, pretinzând că are nevoie de date personale, medicale sau financiare pentru a confirma identitatea țintei;
- schimbul (*quid pro quo*) – în care informațiile solicitate se solicită în schimbul a altceva, cum ar fi un cadou, o vacanță gratuită etc. ;
- frauda de identitate – în care se utilizează identitatea furată a unei persoane pentru a obține bunuri sau servicii.

Tacticile utilizate de atacatori pentru a obține acces la informații sensibile includ utilizarea autorității, intimidării, a consensului în acțiuni, evidențierea unei limitări în disponibilitate, impunerea unui timp limită pentru a accentua o urgență, construirea unui raport de familiaritate sau de încredere cu victima.

## V. Ciclul de viață al unui atac cibernetic

Un atac informatic se desfășoară în mai multe etape. Lockheed Martin au dezvoltat un model cadru pentru identificarea și prevenirea atacurilor din spațiul cibernetic, numit ***Cyber Kill Chain***[5] (sau ciclul de viață al unui atac cibernetic). Acesta cuprinde următorii șapte pași, grupați în trei etape:

- **Înainte de compromitere**
  1. Recunoaşterea – se caută, se identifică și se selectează ținta, iar apoi se culeg cât mai multe informații despre aceasta din spațiul public și din social media (de ex. adrese de email ale angajaților, funcțiile acestora, hobby-uri etc.)sau prin utilizarea diverselor instrumente pentru scanarea vulnerabilităților rețelei, dispozitivelor, serviciilor și aplicațiilor: analizoare de rețea (scanere de pachete sau de protocoale, de ex. *tcpdump* și *Wireshark*), scanere de proturi (de ex. *Nmap*) scanere de vulnerabilități (de ex. *Nessus*) spărgătoare de parole (de ex. *John the Ripper*) etc.
  2. Pregătirea atacului (înarmarea) – se determină ce metodă se va utiliza pentru comăpromiterea țintei și se identifică sau se creează exploit-uri care se cuplează într-o încărcătură malițioasă (*payload*)
  3. Livrarea – se trimite pachetul malițios către victimă prin email, redirectare web, USB, partajare de fișiere infectate sau prin alte metode.
- **În timpul compromiterii**
  4. Exploatare – se exploatează o vulnerabilitate pentru a se executa cod pe sistemul victimei fie prin declanșarea involuntară a exploit-ului de către utilizator (de ex. click pe un link sau deschidere unui fișier infectat, atașat la email), fie prin declanșarea de la distanță de către atacator a unui exploit al unei vulnerabilități cunoscute.
  5. Instalare – se instalează programe *malware* și *backdoor* pe sistemul compromis.
- **După compromitere**
  6. Comandă și control – se controlează de la distanță dispozitivul victimei printr-un canal criptat de comandă și control sau printr-un server C2 (*command & control*); pentru a se păstra nedetectat traficul C2 se utilizează: criptarea, evitarea prin intermediari (*proxy*) sau instrumente de acces la distanță, ocolirea porturilor (*port hoping*), tunelarea etc.
  7. Acțiuni asupra obiectivului – atacatorul fură informații, distruge sau modifică sisteme critice, rețele sau date, atacă alte echipamente din rețea sau utilizează ținta compromisă ca platformă pentru a sprijini atacul asupra unei alte victime.

De asemenea, se poate observa că primii patru pași constituie **faza accesului neautorizat**, în timp de ultimii patru pași constituie **faza utilizării neautorizate**.

## VI. Contramăsuri de securitate

Contramăsurile de securitate pe care o organizație le are la dispoziție se încadrează în două mari categorii:

- **contramăsuri bazate pe tehnologie** – Aceste contramăsuri includ utilizarea unor dispozitive de protecție (firewall) hardware și software, echipamente de detecție și prevenție a intruziunilor (IDS și IPS) și diverse tipuri de scanere de rețea, de porturi și de vulnerabilități.
- **contramăsuri bazate pe factorul uman** – În această categorie sunt incluse educarea și instruirea periodică a personalului pentru conștientizarea problemelor de securitate cibernetică și implementarea unor politici, proceduri, ghiduri și bune practici de securitate.

[5] Cyber Kill Chain® | Lockheed Martin (Septembrie, 2023)

Pentru asigurarea protecția datelor stocate, măsurile de securitate includ, printre altele, criptarea datelor sensibile pentru confidențialitate, realizarea de copii de siguranță și recuperarea datelor alterate pentru implementarea disponibilității. Datele aflate în tranzit sunt protejate utilizând măsuri de securitate cum ar fi criptarea pentru garantarea confidențialității, hashing pentru asigurarea autenticității și integrității, respectiv asigurarea redundanței pentru susținerea disponibilității. Pentru datele care nu sunt stocate și care nici nu sunt în tranzit, măsurile de securitate sunt îndreptate în special către o mai bună proiectare a aplicațiilor și sistemelor software și hardware astfel încât datele de intrare să fie validate, erorile software (bug-urilor) să fie diminuate și dispozitivele de ieșire să fie corect configurate.

### 6.1. Controlul accesului

Controlul accesului este procesul prin care sunt restricționate drepturile de acces pentru a controla utilizarea rețelei, a echipamentelor, a datelor, serviciilor și aplicațiilor. Acesta include activități pornind de la gestionarea accesului fizic într-o locație sau la o resursă, până la impune cine are acces și ce poate face cu aceasta. Multe vulnerabilități de securitate sunt create de utilizarea necorespunzătoare a proceselor de control al accesului. Există mai multe tipuri diferite de control al accesului: control fizic, control logic și control administrativ.

**Controlul fizic** al accesului reprezintă barierele ridicate, în afara sau în interiorul perimetrului protejat, pentru a preveni contactul fizic direct cu echipamentele și cu rețeaua. Scopul este acela de a preveni utilizatorii neautorizați să obțină acces fizic la facilități, echipamente și alte bunuri ale organizației. Controlul fizic determină **cine** poate intra (sau ieși), **unde** pot intra (sau ieși) și **când** pot intra (sau ieși).

Exemple de control fizic al accesului pot fi următoarele: pază, garduri, porți, detectoare de mișcare, încuietori pentru echipamente, încuietori pentru uși, carduri pentru acces, camere video, sisteme de intrare specializate (de ex. *mantrap*), alarme pentru a detecta intruziunile etc.

**Controlul logic** al accesului se referă la soluții hardware și software utilizate pentru a gestiona accesul la resurse și sisteme. Aceste soluții bazate pe tehnologii includ instrumentele și protocoalele pe care sistemele de calcul le utilizează pentru identificare, autentificare, autorizare și auditare.

Pentru controlul logic al accesului se pot utiliza: criptare, carduri smart, parole, identificatori biometrici, , liste de control al accesului (ACL), protocoale, firewall-uri, ruter-e, sisteme de detecție a intruziunilor (IDS) etc.

**Controlul administrativ** al accesului este reprezentat de politicile și procedurile definite pentru a implementa și impune toate aspectele de control al accesului neautorizat.

Lista exemplelor de control administrativ include: politicile (obiective, reguli și cerințe), procedurile (pași detaliați necesari pentru a efectua o activitate), practicile de angajare (pașii pe care îi face o organizație pentru a găsi angajați calificați), verificările antecedentelor (informații de la foști angajatori, istoricul creditului, istoricul infracțional etc.), clasificarea datelor (pe baza sensibilității), instruirea (educarea angajaților în ceea ce privește politicile de securitate) ș.a.

### 6.2. Servicii de securitate

Conceptul de control administrativ al accesului implică trei servicii esențiale de securitate: autentificarea, autorizarea și contabilizare. Aceste servicii furnizează cadrul principal de control al accesului, prevenind accesul neautorizat la sisteme, rețele, date sau alte resurse.

**Autentificarea** este procesul de confirmare a identității unui utilizator sau sistem, pentru a preveni accesul neautorizat. Aceasta asigură faptul că persoana sau entitatea care încearcă să acceseze resursele sau să efectueze acțiuni este ceea ce pretinde a fi.

**Autorizarea** este procesul de acordare sau refuzare a accesului la anumite resurse. Serviciile de autorizare determină care resurse pot fi accesate de utilizatori, împreună cu operațiile pe care aceștia le pot face cu acele resurse. Unele sisteme realizează acest lucru prin intermediul listelor de control al accesului (ACL) care determină dacă un utilizator poate avea anumite privilegii după ce se autentifică.

Autorizarea poate controla și când un utilizator are acces la o anumită resursă. De exemplu, angajații pot avea acces la baza de date în timpul orelor de lucru, dar accesul nu este permis după încheierea programului.

**Auditarea** este procesul de înregistrare și monitorizare a activităților și evenimentelor care au loc într-o rețea sau pe un sistem de calcul. Sunt păstrate operațiile realizate de utilizatori: ce resurse accesează, cât timp și ce modificări au făcut. Scopul auditării este să asigure conformarea cu politicile de securitate și să faciliteze identificarea eventualelor amenințări și abuzuri, pentru a preveni și detecta incidentele de securitate.

### 6.3. Autentificarea multi-factor

Un identificator unic asigură asocierea potrivită între activitățile permise și subiecți. Pentru autentificare, utilizatorii furnizează identitatea lor cu un nume sau ID. Un nume de utilizator este cea mai obișnuita metodă utilizată pentru a identifica un utilizator printr-o combinație de carcatere alfanumerice. Un identificator unic asigură că un sistem poate identifica fiecare utilizator individual, permițând așadar ca un utilizator autorizat să efectueze acțiunile corespunzătoare asupra anumitor resurse.

În plus, utilizatorii trebuie să demonstreze identitatea prin furnizarea a:

- ceva ce cunosc – de ex. o parolă, o frază sau un cod PIN
- ceva ce au – un *token*, un card sau o cheie
- ceva ce sunt - identificatori biometrici – o amprentă sau altă caracteristică fizică (față, mână, retină, ureche) sau comportamentală (gest, voce, mers).

În cazul autentificării cu doi (2FA) sau mai mulți factori (MFA), pentru a verifica identitatea cuiva, este necesară o combinație de cel puțin două din mijloacele de autentificare enumerate de mai sus.

De exemplu, un website al unei bănci poate cere o parolă și un cod PIN pe care utilizatorul îl primește pe telefon. În acest caz, primul factor este parola iar al doilea factor este un cod temporar, deoarece dovedește că aveți acces la ceea ce este înregistrat ca telefon. Retragerea de bani de la bancomat este un alt exemplu simplu de autentificare multi-factor deoarece utilizatorul trebuie să aibă cardul bancar și să cunoască PIN-ul.

Autentificarea multi-factor poate reduce incidentele de furt de identitate deoarece înseamnă că simpla cunoașterea a unei parole nu va oferă infractorului cibernetic acces direct la contul utilizatorului.

### 6.4. Criptografia

Criptografia este știința creării și spargerii codurilor secrete. Criptarea este procesul de transformare a datelor astfel încât persoanele neautorizate nu le pot citi cu ușurință. Acest proces convertește mesajul citibil (text în clar, eng. *plaintext*) în mesaj cifrat (eng. *ciphertext*), care este necitibil, mesaj deghizat. Decriptarea inversează procesul. Prin stocarea și transmiterea datelor criptate, astfel încât doar

destinatarul dorit să poată citi sau procesa datele, se oferă protecția datelor. Asta înseamnă că utilizatorii neautorizați nu pot citi cu ușurință informațiile sensibile.

Criptarea necesită o cheie care joacă un rol critic în criptarea și decriptarea mesajului. Persoana care posedă cheia poate descifra mesajul criptat, adică poate transforma *ciphertext* în *plaintext*.

### 6.4.1. Tipuri de criptografie

Cele mai utilizate tipuri de criptografie utilizează cifruri bloc (***block***) sau cifruri flux (***stream***). Fiecare metodă diferă în modul în grupuri de biți de date pe care îi criptează.

**Cifrurile bloc** transformă un bloc de text în clar de lungime fixă într-un bloc obișnuit de text cifrat de 64 sau 128 de biți. Dimensiunea blocului este volumul de date criptate la orice moment de timp. Pentru a decripta acest text cifrat, se aplică transformarea inversă pentru blocul cifrat, utilizând aceeași cheie secretă.

Cifrurile bloc rezultă de obicei în datele rezultat care este mai mare decât datele de intrare, deoarece textul cifrat trebuie să fie multiplu de dimensiunea blocului. De exemplu, DES (*Data Encryption Algorithm*) este un algoritm simetric care criptează blocuri în bucăți de 64 biți, folosind o cheie de 56 biți. Pentru a realiza asta, algoritmul ia câte un bloc pe rând – de exemplu, 8 biți pe bucată – până când întreg blocul este plin. Dacă există mai puține date de intrare decât un bloc plin, algoritmul adaugă date artificiale, sau spații până umple 64 de biți.

**Cifrurile flux** criptează textul în clar un bit după celălalt, rând pe rând, câte un bit la un moment dat. Gândiți-vă la un cifru flux ca la un cifru bloc de dimensiune 1 bit. Cu un cifru flux, transformările acestor mici unități de text clar variază, depinzând de când au loc în timpul procesului de criptare. Cifrurile flux pot fi mult mai rapide decât cele bloc și, în general, nu cresc dimensiunea mesajului ce este criptat, deoarece pot cripta un număr arbitrar de biți.

De exemplu, A5 este un cifru flux care furnizează intimitatea vocii și criptează comunicațiile telefoanelor mobile. Este de asemenea posibil să se utilizeze DES în modul cifru flux.

Evident, sistemele criptografice complexe pot combina blocuri și fluxuri în același proces.

### 6.4.2. Clase de algoritmi de criptare

Criptografia modernă utilizează algoritmi de securitate pentru a se asigura că infractorii cibernetici și alți actori rău intenționați nu pot compromite cu ușurință informații protejate. Toate metodele de criptare utilizează chei pentru a cripta și decripta mesaje – și securitatea criptării se bazează pe secretul cheilor, nu pe secretul algoritmului. În criptografia modernă, algoritmii sunt publici. Cheile criptografice trebuie să asigure secretul datelor.

Un algoritm de criptare este la fel de bun ca cheia pe care o folosește. Cu cât e implicată mai multă complexitate, cu atât e mai sigur algoritmul. Managementul cheilor este o parte importantă și extrem de dificilă a unui sistem criptografic.

Există două clase de algoritmi de criptare:

- **Criptarea simetrică** – acești algoritmi utilizează aceeași cheie pre-partajate, uneori numită cheie secretă, pentru a cripta și decripta datele. Atât expeditorul cât și destinatarul cunosc cheia pre-partajată înainte de a începe comunicarea criptat. Algoritmii simetrici utilizează aceeași cheie pentru a cripta și decripta mesajul la ambele capete ale procesului. Algoritmii de criptare care utilizează o cheie comună sunt mai simpli și au nevoie de mai puțină putere de calcul deoarece se bazează pe operații matematice simple.

Exemple:

- **3DES** (triple DES) - criptează datele de trei ori folosind DES și utilizează chei diferite pentru cel puțin una din cele trei treceri, rezultând o cheie cumulată de la 112 la 168 de biți.
- **IDEA** (*International Data Encryption Algorithm*) utilizează blocuri de 64 biți și 128 de biți, efectuând 8 treceri de transformare a fiecăruia din cele 16 blocuri care rezultă din împărțirea fiecărui bloc de 64 biți.
- **AES** – (*Advanced Encryption Standard*) utilizează un bloc de dimensiune fixă de 128 de biți cu o cheie de dimensiune 128, 192 sau 256 de biți. NIST (*National Institute of Standards and Technology*) a aprobat algoritmul AES în decembrie 2001, iar guvernul Statelor Unite utilizează AES pentru a proteja informațiile clasificate.
- **Blowfish**

- **Criptare asimetrică** – Algoritmii de criptare asimetrică utilizează două chei pentru a cripta datele. O cheie este publică și alta este privată. În sistemul de criptare cu cheie publică, orice persoană poate cripta mesajul utilizând cheia publică a receptorului și receptorul este singurul care poate decripta mesajul folosind cheia sa privată. Părțile schimbă mesaje sigure fără a fi nevoie de o cheie pre-partajată. Algoritmii asimetrici sunt mai complecși, consumatori de resurse și mai lent de executat întrucât au la bază operații matematice complexe.
  Exemple:
  - **RSA** (*Rivest–Shamir–Adleman*) – utilizează produsul a două numere prime foarte mari cu o lungime între 100 și 200 de cifre. Browser-ele web utilizează RSA pentru a stabili o conexiune sigură.
  - ***Diffie-Hellman*** – oferă o metodă de schimb electronic ce partajează o cheie secretă. Protocoalele sigure cum ar fi SSL (Secure Sockets Layer), TLS (*Transpot Layer Security*), SSH (*Secure Shell*) și IPsec (*Internet Protocol Security*) utilizează *Diffie-Hellman*.
  - ***ElGamal*** – utilizează standardul guvernului Statelor Unite pentru semnături digitale.
  - **ECC** (*Elliptic Curve Cryptography*) – utilizează curbele eliptice ca parte a algoritmului. În Statele Unite, NSA utilizează ECC pentru generarea semnăturilor digitale și schimbul de chei.

## VII. Bune practici și ghiduri de securitate cibernetică

Oamenii sunt prima linie de apărare în securitatea cibernetică și fiecare instituție este la fel de puternică ca și cea mai slabă verigă. Investițiile în tehnologie nu vor va face o diferență semnificativă în lupta cu infractorii cibernetici, dacă oamenii din organizație nu sunt instruiți. Fiecare angajat al unei instituții trebuie să fie conștient de politicile de securitate și să le implementeze în activitățile de zi cu zi.

Conștientizarea securității trebuie să fie un proces continuu deoarece noi amenințări și tehnologii apar continuu. Construirea unei culturi eficiente a securității cibernetice necesită efort continuu și implicarea tuturor membrilor instituției.

### 7.1. Politicile de securitate

O **politică de securitate** stabilește obiective de securitate, reguli de comportament și cerințele sistemelor utilizate. Politicile de securitate informează utilizatorii, angajații și managerii despre cerințele instituției pentru protecția bunurilor tehnologice și informatice. O politică de securitate specifică și mecanismele necesare pentru a îndeplini cerințele de securitate.

Următoarele politici pot fi date ca exemplu: politici de identificare și autentificare, politicile pentru crearea, utilizarea și schimbarea parolelor, politica pentru utilizarea acceptabilă, politicile pentru accesul la distanță, politicile de mentenanță, politicile de gestionare a incidentelor ș.a.

O listă extinsă a politicilor generale sau specifice privind securitatea informațiilor, aplicațiilor, serverelor, sau rețelei precum și a o serie de șabloane care pot fi utilizate pentru crearea unor astfel de politici poate fi găsită la SANS[6].

### 7.2. Standardele de securitate

Standardele ajută angajații să mențină consistența în utilizarea resurselor instituției, să îmbunătățească eficiența, mentenanța și depanarea resurselor informatice. Unul din cele mai importante principii de securitate este consistența. Din acest motiv, este necesar pentru organizații să stabilească standarde. Fiecare organizație dezvoltă standarde care susțin propriul lor mediu unic de operare.

ISO/IEC 27000[7] reprezintă o serie de standarde de securitate a informației care să ajute organizațiile și instituțiile să-și îmbunătățească securitatea. Publicate de ISO și ICO , standardele ISO 27000 stabilesc cerințele unui sistem cuprinzător ISMS (*Information Security Management System*). Un ISMS constă din toate măsurile administrative, tehnice și operaționale care adresează securitatea informației într-o organizație.

Standardul ISO 270000 se referă de 12 domenii independente care furnizează bazele pentru dezvoltarea de standarde de securitate și practici eficiente de management al securității în organizații și ajută la facilitarea comunicației între organizație: evaluarea riscului, politica de securitate, organizarea securității informației, managementul bunurilor, securitatea resurselor umane, securitatea fizică și a mediului, managementul comunicației și operațiilor, achiziția, dezvoltarea și mentenanța sistemelor informatice, controlul accesului, managementul incidentului de securitate, managementul continuității activității, conformitatea cu politicile standardele și regulamentele de securitate.

### 7.3. Ghiduri de securitate și bune practici în securitatea cibernetică

Ghidurile sunt o listă de sugestii privind modul în care se pot realiza lucrurile mai eficient și mai sigur. Sunt similare cu standardele dar sunt mult mai flexibile și nu sunt de obicei obligatorii. Ghidurile definesc modul în care sunt dezvoltate standardele și garantează aderarea la politicile generale de securitate. Există numeroase ghiduri disponibile, cum ar fi: NIST *Computer Resource Center*[8], NSA Security Configuration Guides[9], ENISA Guidelines[10] etc.

Standardele și unele din cele mai utile și ghiduri permit identificarea celor mai bune practici în domeniul securității cibernetice. Acestea pot face referire la:

- **Evaluarea riscurilor** pentru înțelegerea importanței bunurilor ce trebuie protejate și determinarea sumele alocate pentru securitate.
- **Crearea unei politici de securitate** în care sunt stipulate în mod clar regulile instituției, responsabilitățile și așteptările.
- **Implementarea unor măsuri privind securitatea fizică** pentru restricționarea accesului la zonele în care se găsesc resursele importante.

---

[6] Information Security Policy Templates | SANS Institute (Septembrie, 2023)
[7] ISO/IEC 27000:2018 - Information technology — Security techniques — Information security management systems — Overview and vocabulary (Septembrie, 2023)
[8] NIST Computer Security Resource Center | CSRC (Septembrie, 2023)
[9] Cybersecurity Advisories & Guidance (nsa.gov) (Septembrie, 2023)
[10] Guidelines — ENISA (europa.eu) (Septembrie, 2023)

- **Implementarea unor măsuri de securitate privind resursele umane** pentru verificarea angajaților în mod corespunzător.
- **Realizarea de *backup* periodic** și testarea soluțiile de backup.
- **Implementarea de actualizări de securitate** pentru sisteme de operare, servicii și aplicații.
- **Controlul accesului** prin configurarea de roluri și niveluri de privilegii pentru utilizatori
- **Implementarea unor servicii de autentificare** puternice.
- **Testarea periodică a răspunsului la incidente** și a scenariilor de răspuns în caz de urgență.
- **Implementarea unor instrumente de monitorizare și administrare** a rețelei.
- **Implementarea unor echipamente de securitate** în rețea (de ex. ruter, firewall).
- **Implementarea unor soluții de securitate** pentru sistemele de calcul (de ex. programe antimalware și antivirus).
- **Instruirea utilizatorilor** și a angajaților.
- **Criptarea datele** importante.

### 7.4. Proceduri de securitate

Procedurile sunt documente mai lungi și mai detaliate decât standardele și ghidurile. Acestea includ detalii de implementare ce conțin de obicei instrucțiuni pas cu pas și reprezentări grafice pentru a ilustra modul de realizare a unei anumite activități.

Orice organizație sau instituție trebuie să utilizeze o serie de proceduri pentru a menține consistența desfășurării activității într-un mediu sigur. Toți angajații ar trebui să fie educați și informați asupra procedurilor de securitate și a acțiunilor care trebuie întreprinse în cazul unei breșe de securitate.

## VIII. Răspunsul la incident și recuperarea după dezastru

Chiar după implementarea tuturor măsurilor de securitate necesare, este aproape inevitabil să nu apară o breșă de securitate la un moment dat. Existența unui plan de răspuns rapid și eficient poate reprezenta diferența dintre un incident cu consecințe minore și un dezastru major. Cu o abordare bine organizată, se pot limita distrugerile (minimizarea impactului) și se pot reduce timpul și costurile de recuperare.

### 8.1. Răspunsul la incident

Răspunsul la incident este un set de proceduri pe care trebuie să le urmeze o organizație în cazul unui incident de securitate sau la apariția amenințări cibernetice. Acest răspuns cuprinde mai multe etape:

#### 8.1.1. Pregătirea

O organizație trebuie să dezvolte un plan de răspuns la incident și să alcătuiască o echipă CSIRT (*Computer Security Incident Response Team*) care să gestioneze răspunsul. În funcție de mărimea organizației și de complexitatea amenințărilor, rolurile acestei echipe includ: detectarea incidentelor, analiza incidentelor, răspunsul la incidente, comunicarea și coordonarea cu alte echipe, documentarea și raportarea, prevenirea altor incidente viitoare, instruirea și conștientizarea angajaților și monitorizarea continuă a mediului IT, îmbunătățirea securității.

#### 8.1.2. Detecția și analiza

**Detecția** corespunzătoare înseamnă depistarea momentului în care a apărut un incident, ce date și ce sisteme au fost implicate. Detecția, începe atunci când cineva descoperă un incident.

O organizație poate avea cele mai sofisticate sisteme de detecție dar acestea nu valorează nimic dacă administratorii nu urmăresc jurnalele și alertele. Odată ce a fost detectat un incident de securitate, notificările sunt trimise către conducere și responsabili pentru date și sisteme, astfel încât aceștia să poată fi implicați în remediere și reparare.

După detectare, este necesar să se confirme dacă există cu adevărat un incident de securitate cibernetică. Aceasta poate implica investigații suplimentare pentru a determina natura și amploarea incidentului. **Analiza incidentului** ajută la identificarea sursei, extinderii, impactului și detaliilor incidentului, inclusiv breșa de date. Organizația poate decide se cheme în echipă un expert pentru a desfășura investigația criminalistică. O investigație detaliată poate fi esențială pentru înțelegerea completă a incidentului

#### 8.1.3. Izolarea, eradicarea și recuperarea

O parte esențială a răspunsului la un incident de securitate cibernetică este **izolarea** acestuia pentru a preveni răspândirea la alte sisteme sau resurse din cadrul organizației și a reduce impactul și daunele cauzate de acesta. De exemplu, se poate realiza deconectarea sistemului afectat de la rețea pentru a opri scurgerea e informații.

După identificarea și izolarea breșei de securitate urmează etapa de răspuns la incident în care se iau măsuri de **eradicare** (eliminare) a efectelor incidentului (de ex. eliminarea *malware*-ului).

Această etapă este urmată de recuperare, adică remedierea resurselor afectate și restaurarea sistemelor și a datelor la starea lor originală.

#### 8.1.4. Urmarea incidentului

După gestionarea cu succes a incidentului și restaurarea sistemelor la starea lor normală, pasul final al procesului este responsabil cu abordarea tuturor aspectelor legate de incident pentru identificarea cauzei și impactului incidentului, măsurile luate și analiza lecțiilor învățate: ce acțiuni vor preveni reapariția incidentului sau apariția unor incidente similare, ce măsuri preventive trebuie îmbunătățite, cum se poate minimiza impactul unui astfel de incident, etc. Toate aceste lucruri sunt documentate și raportate pentru a îmbunătăți măsurile de securitate ale organizației.

### 8.2. Recuperarea după dezastru

Un dezastru include orice eveniment uman sau natural care cauzează distrugeri de bunuri sau proprietăți și afectează abilitatea unei organizații de a continua să opereze. Există două tipuri de dezastre: naturale și provocate de om.

**Dezastrele naturale** diferă în funcție de locație și sunt dificil de prezis:

- dezastre geologice (de ex. cutremure, alunecări de teren, erupții vulcanice și tsunami),
- dezastre meteorologice (de ex. uragane, tornade, viscol, fulgere și grindină),
- dezastre de sănătate (de ex. fi carantină și pandemii),
- alte dezastre (de ex. incendii, inundații, explozii solare și avalanșe).

**Dezastrele cauzate de om** implică oameni sau organizații și intră în următoarele categorii:

- evenimente de muncă (de ex. greve),
- evenimente socio-politice (de ex. vandalisme, blocade, proteste, sabotaje, terorism și război),
- evenimente materiale – (de ex. deversările de substanțe sau incendiile),

- întreruperi de utilități (alimentarea cu energie electrică), întreruperi de comunicații, combustibil și dezastre radioactive.

Atunci când are loc un dezastru, pentru a se asigura că sistemele critice sunt în funcțiune, o organizație trebuie să pună în acțiune un **plan de recuperare** după dezastru (*Disaster Recovery Plan* – DRP). Acest plan include toate activitățile ce trebuie desfășurate pentru a evalua, salva, repara și restaura facilitățile sau bunurile afectate. De asemenea, un DRP trebuie să identifice care sunt procesele și sistemele critice, pentru a prioritiza restaurarea acestora în timpul procesului de recuperare.

**Măsurile de recuperarea** după dezastru încearcă să minimizeze efectele unui dezastru astfel încât să poată fi reluată funcționarea. Există trei tipuri de măsuri de recuperare după dezastre:

- măsuri preventive – au drept scop identificarea și reducerea riscurilor,
- măsurile de detecție – descoperă evenimentele nedorite și posibile amenințări,
- măsuri de corecție – restaurează sistemul după un dezastru sau un eveniment.

În ciuda celor mai bune eforturi de a preveni dezastrele și pierderea datelor, este imposibil să se țină cont de fiecare scenariu. De aceea este critic să existe un **plan de continuitate** a afacerii (*Business Continuity Plan* – BCP), indiferent de ceea ce se întâmplă.

Un plan de continuitate a afacerii este un plan mai acoperitor decât DRP deoarece poate include mutarea sistemelor critice într-o altă locație în timp ce se repară sediul original. Într-un astfel de scenariu, angajații vor continua să efectueze toate procesele afacerii într-o manieră alternativă până se reiau operațiile normale.

## IX. Bibliografie


[1] **Cisco Skills For All**

Introduction to Cybersecurity

https://skillsforall.com/course/introduction-to-cybersecurity?userLang=en-US

Cybersecurity Essentials

https://skillsforall.com/course/cybersecurity-essentials?userLang=en-US

[2] **Coursera**

Palo Alto Networks Cybersecurity Foundation

https://www.coursera.org/learn/palo-alto-networks-cybersecurity-foundation-a

# Laborator – Aplicatia 1

**Obiective:**

- Cunoasterea tehnicilor de analiză big data care se pot aplica pe seturile de date experimentale colectate din stațiile de tratare ape uzate;
- Însușirea termenilor specifici și înțelegerea importanței monitorizării online ca aspect important al digitalizarii stațiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În era digitală, analiza datelor a devenit o unealtă crucială pentru optimizarea proceselor în diferite domenii, inclusiv gestionarea resurselor de apă uzată. Acest referat de laborator se concentrează pe utilizarea PySpark, un framework de procesare a datelor în limbajul Python, pentru a analiza date legate de calitatea apei uzate și factorii meteorologici asociate cu aceasta. Scopul este de a dezvolta un model de regresie liniară care să poată prezice cantitatea de apă uzată în funcție de variabilele observate.

**Calitatea Apei Uzate:** Datele de calitate a apei uzate pot include parametri precum concentrația de poluanți (de exemplu, BOD, COD, azot, fosfor, metale grele), indicatori de pH, soliditate, temperatura apei și alți parametri relevanți. Aceste date pot fi colectate de la stațiile de tratare a apei uzate sau laboratoare specializate.

**Date Meteorologice:** Datele meteorologice pot include informații precum temperatura aerului, precipitațiile, umiditatea, direcția și viteza vântului, presiunea atmosferică și altele. Aceste date pot fi colectate de la stații meteorologice locale sau din surse meteorologice oficiale.

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatia1").getOrCreate()
```

**Pasul 3: Incarcați Datele**

Încărcăm datele de calitate a apei uzate dintr-un fișier CSV și datele meteorologice din alt fișier CSV. Aceste date vor fi stocate în DataFrame-uri PySpark

```
# Încărcați datele de apă uzată într-un DataFrame PySpark
data_apa_uzata_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Încărcați datele meteorologice într-un DataFrame PySpark (înlocuiți cu
fișierul dvs. de date meteorologice)
date_meteorologice_df = spark.read.csv("date_meteorologice.csv", header=True,
inferSchema=True)
```

**Pasul 4: Concatenați datele**

Unim cele două DataFrame-uri folosind o coloană comună, cum ar fi "Data". Aceasta va permite să analizăm datele din cele două surse în mod corespunzător.

```
# Uniți cele două DataFrame-uri folosind o coloană comună ("Data") folosind
metoda "join"
data_combinata_df = data_apa_uzata_df.join(date_meteorologice_df, "Data",
"inner")
```

**Pasul 5: Pregătiți Datele**

Realizăm pregătirea datelor, inclusiv vectorizarea caracteristicilor de interes. În acest caz, coloanele de interes includ "CalitateApaUzata", "Temperatura" și "Precipitații".

```
# Definim coloanele de caracteristici
coloane_caracteristici = ["CalitateApaUzata", "Temperatura", "Precipitații"]

# Creăm un asamblor pentru vectorizarea caracteristicilor
asamblor = VectorAssembler(inputCols=coloane_caracteristici,
outputCol="caracteristici")

# Aplicăm asamblorul pentru a transforma datele
date_pregatite = asamblor.transform(data_combinata_df)
```

**Pasul 6: Implementați modelul de Regresie Liniară**

Definim un model de regresie liniară care va fi antrenat pe datele pregătite. Acest model va ajuta la prezicerea cantității de apă uzată în funcție de caracteristicile observate.

```
from pyspark.ml.regression import LinearRegression

# Definim modelul de Regresie Liniară
rl = LinearRegression(featuresCol="caracteristici",
labelCol="CantitateApaUzata")
```

**Pasul 7: Divizați datele și antrenați modelul**

Divizăm datele în seturi de antrenament și testare și apoi antrenăm modelul de regresie liniară.

```
# Divizăm datele în seturi de antrenament și testare
date_antrenament, date_testare = date_pregatite.randomSplit([0.7, 0.3])

# Antrenăm modelul de Regresie Liniară
model_rl = rl.fit(date_antrenament)
```

**Pasul 8: Realizați predicții și evaluați performanța**

Folosim modelul antrenat pentru a realiza predicții pe datele de testare și apoi evaluăm performanța modelului folosind metrica RMSE (Rădăcina Eroare Medie Patratică).

```
# Evaluați performanța modelului
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator(labelCol="CantitateApaUzata",
predictionCol="previziune", metricName="rmse")
rmse = evaluator.evaluate(previziuni)
print(f"Rădăcina Eroare Medie Patratică (RMSE): {rmse}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 9: Interpretați rezultatele și explicați operațiile realizate.**

# Laborator – Aplicatia 2

**Obiective:**

- Cunoasterea tehnicilor de analiză big data care se pot aplica pe seturile de date experimentale colectate din stațiile de tratare ape uzate;
- Însușirea termenilor specifici și înțelegerea importanței monitorizării online ca aspect important al digitalizarii stațiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În acest laborator, vom utiliza PySpark pentru a analiza date referitoare la calitatea apei uzate. Scopul este de a dezvolta un model de clasificare pentru a determina dacă apa uzată este de "Calitate Bună" sau "Calitate Scăzută" pe baza caracteristicilor observate.

Un exemplu de sursă pentru un set de date legat de calitatea apei uzate și caracteristicile asociate:

**UCI Machine Learning Repository - Water Quality Data Set:**

Acest set de date conține informații despre calitatea apei uzate și caracteristicile apei, cum ar fi temperatură, pH, soliditate și altele. Puteți descărca setul de date de la UCI Machine Learning Repository folosind următorul link: https://www.kaggle.com/datasets/adityakadiwal/water-potability

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatie2").getOrCreate()
```

**Pasul 3: Incărcați Datele**

Încărcați datele de calitate a apei uzate și caracteristicile asociate în DataFrame-uri PySpark.

```
# Încărcați datele de calitate a apei uzate într-un DataFrame PySpark
wastewater_df = spark.read.csv("wastewater_data.csv", header=True,
inferSchema=True)

# Încărcați caracteristicile asociate cu apa uzată (exemplu: temperatură, pH,
conductivitate)
features_df = spark.read.csv("water_features.csv", header=True,
inferSchema=True)
```

**Pasul 4: Pregătiți datele**

Realizați pregătirea datelor, inclusiv prelucrarea caracteristicilor și etichetarea datelor pentru clasificare.

```
from pyspark.ml.feature import VectorAssembler

# Definiți coloanele de caracteristici pentru vectorizare
feature_columns = ["Temperature", "pH", "Conductivity"]

# Creează un asamblor pentru a combina caracteristicile într-o singură
coloană
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")

# Aplică asamblorul pentru caracteristici
data = assembler.transform(features_df)

# Etichetați datele: "Good" sau "Low" calitate a apei uzate
from pyspark.sql.functions import when
wastewater_df = wastewater_df.withColumn("QualityCategory",
    when(wastewater_df["WastewaterQuality"] >= 80, "Good").otherwise("Low")
)
```

**Pasul 5: Implementați modelul de clasificare**

Definiți un model de clasificare, cum ar fi Random Forest, și antrenați-l pentru a clasifica apa uzată ca "Calitate Bună" sau "Calitate Scăzută".

```
from pyspark.ml.classification import RandomForestClassifier
```

```
# Definiți modelul de clasificare
rf = RandomForestClassifier(featuresCol="features",
labelCol="QualityCategory", numTrees=10)

# Divizați datele în seturi de antrenament și testare
train_data, test_data = data.randomSplit([0.7, 0.3])

# Antrenați modelul pe datele de antrenament
model = rf.fit(train_data)
```

**Pasul 6: Evaluați performanța modelului**

Evaluează performanța modelului de clasificare folosind metrici precum precizia, recuperarea și F1-score.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Realizați predicții pe datele de testare
predictions = model.transform(test_data)

# Evaluator pentru metrici de clasificare
evaluator = MulticlassClassificationEvaluator(labelCol="QualityCategory",
metricName="accuracy")

# Calculați precizia modelului
accuracy = evaluator.evaluate(predictions)
print(f"Precizia modelului: {accuracy}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 7: Interpretați rezultatele și explicați operațiile realizate.**

# Laborator – Aplicatia 3

**Obiective:**

- Cunoasterea tehnicilor de analiza big data care se pot aplica pe seturile de date experiementale colectate din statiile de tratare ape uzate;
- Însușirea termenilor specifici si intelegerea importantei monitorizarii online ca aspect important al digitalizarii statiilor de epurare ape uzate.
- Integrarea datelor, învățarea automată și modelarea predictivă

**Introducere**

În acest laborator, ne vom concentra pe utilizarea PySpark pentru analiza datelor de calitate a apei uzate și pentru dezvoltarea unui model de regresie liniară care să poată prezice consumul de apă uzată pe baza caracteristicilor observate.

**Pasul 1: Deschideti Google Colab** (https://colab.research.google.com/) si **parcurgeti pasii de instalare**:

```
# Instalați Java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# Instalați Spark (modificați numărul versiunii, dacă este necesar)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# Dezarhivați fișierul Spark în directorul curent
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# Setați calea directorului Spark la variabilele de mediu ale sistemului dumneavoastră.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# Instalați findspark folosind pip
!pip install -q findspark
```

**Pasul 2: Configurati sesiunea Spark**

Începem prin configurarea unei sesiuni Spark folosind PySpark. Această sesiune va permite să lucrăm cu datele mari și să folosim funcționalitățile puternice de analiză.

```
from pyspark.sql import SparkSession

# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluAplicatie3").getOrCreate()
```

**Pasul 3: Încărcați Datele**

Încărcați datele de calitate a apei uzate și caracteristicile asociate în DataFrame-uri PySpark.

```
from pyspark.sql import SparkSession
# Creați o sesiune Spark
spark = SparkSession.builder.appName("ExempluApaUzata2").getOrCreate()

# Încărcați datele de calitate a apei uzate într-un DataFrame PySpark
wastewater_df = spark.read.csv("date_apa_uzata.csv", header=True,
inferSchema=True)

# Încărcați caracteristicile asociate cu apa uzată (exemplu: temperatură, pH,
conductivitate)
features_df = spark.read.csv("date_caracteristici_apa.csv", header=True,
inferSchema=True)
```

**Pasul 4: Pregătiți datele**

Realizați pregătirea datelor, inclusiv prelucrarea caracteristicilor și etichetarea datelor pentru regresie.

```
from pyspark.ml.feature import VectorAssembler

# Definiți coloanele de caracteristici pentru vectorizare
feature_columns = ["Temperature", "pH", "Conductivity"]

# Creează un asamblor pentru a combina caracteristicile într-o singură
coloană
assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")

# Aplică asamblorul pentru caracteristici
data = assembler.transform(features_df)

# Definiți coloana de etichetă (cantitatea de apă uzată)
label_column = "WastewaterFlow"
```

**Pasul 5: Implementați modelul de regresie Liniară**

Definiți un model de regresie liniară și antrenați-l pentru a prezice cantitatea de apă uzată.

```
from pyspark.ml.regression import LinearRegression

# Definiți modelul de regresie liniară
lr = LinearRegression(featuresCol="features", labelCol=label_column)

# Divizați datele în seturi de antrenament și testare
train_data, test_data = data.randomSplit([0.7, 0.3])
```

```
# Antrenați modelul de regresie liniară
model = lr.fit(train_data)
```

**Pasul 6: Realizați predicții și evaluați performanța**

Folosiți modelul antrenat pentru a face predicții pe datele de testare și evaluați performanța modelului.

```
# Realizați predicții pe datele de testare
predictions = model.transform(test_data)

# Evaluator pentru metricile de regresie
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(labelCol=label_column,
predictionCol="prediction", metricName="rmse")

# Calculați eroarea medie pătratică
rmse = evaluator.evaluate(predictions)

print(f"Rădăcina Eroare Medie Pătratică (RMSE): {rmse}")
```

```
# Oprește sesiunea Spark
spark.stop()
```

**Pasul 7: Interpretați rezultatele și explicați operațiile realizate.**