

Mădălin NEAGU

Gheorghe SEBESTYEN

INFORMATICĂ INDUSTRIALĂ

Teorie și aplicații pentru sisteme industriale moderne



U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-858-0

Mădălin NEAGU

Gheorghe SEBESTYEN

INFORMATICĂ INDUSTRIALĂ

Teorie și aplicații pentru sisteme industriale moderne



U.T. PRESS
Cluj-Napoca, 2026
ISBN 978-606-737-858-0



Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Anca Hângan

Conf.dr.ing. Bogdan Iancu

Pregătire format electronic on-line: Gabriela Groza

Copyright © 2026 Editura U.T.PRESS

Reproducerea integrală sau parțială a textului sau ilustrațiilor din această carte este posibilă numai cu acordul prealabil scris al editurii U.T.PRESS.

ISBN 978-606-737-858-0

Cuprins

Partea I. Fundamentele informaticii industriale.....	1
1. Informatica industrială: evoluție, concepte și cerințe ingineresti	1
1.1 Domeniul și specificul informaticii industriale	1
1.2 De la automatizarea programabilă la industria conectată	2
1.3 De la Industria 4.0 la Industria 5.0	3
1.4 Cerințe ingineresti transversale	3
2. Automatizare industrială, sisteme cyber-fizice și execuție în timp real	5
2.1 Arhitecturi de automatizare: PLC, DCS și SCADA	5
2.2 Sistemele cyber-fizice și lanțul temporal complet.....	7
2.3 Cerințe de funcționare în timp real și planificarea execuției	8
2.4 Alegerea platformei hardware si a sistemului de operare pentru funcționare în timp real	10
2.5 Diagnosticarea controlului și repornirea secvențelor	11
Partea a II-a. Conectivitate și infrastructuri de date	13
3. Edge, IIoT și comunicații industriale moderne.....	13
3.1 Alocarea funcțiilor în continuumul dispozitiv–edge–fog–cloud	13
3.2 IIoT, OPC UA și interoperabilitatea semantică	16
3.3 Protocoale IIoT pentru telemetrie și administrarea dispozitivelor	17
3.4 Rețele senzoriale fără fir, LPWAN și 5G cu capabilități reduse.....	19
3.5 Determinismul comunicației, EtherCAT și TSN	20
3.6 Rețele private 5G și comunicații industriale mobile	22
3.7 Echipamente de interconectare, instalații existente și administrarea ciclului de viață	22
3.8 Inteligența artificială la edge: optimizare, distribuire și exploatare	23
Partea a III-a. Inteligență și control	25
4. Date industriale și procesarea semnalelor.....	25
4.1 Contextul și ciclul de viață al datelor industriale	25
4.2 Arhitecturi de procesare pe loturi, continuă și orientată pe evenimente.....	26
4.3 Calitatea, proveniența, guvernanta și evoluția schemelor	27
4.4 Achiziția și preprocesarea semnalelor industriale	28
4.5 Filtrarea și reprezentările timp–frecvență.....	29
4.6 Extragerea caracteristicilor și reprezentarea temporală a datelor	30
5. Detectarea anomaliilor și mentenanța predictivă.....	32
5.1 Normalitatea, anomaliile și strategiile de mentenanță	32
5.2 Familii de metode pentru detectarea anomaliilor	33
5.3 PHM, estimarea RUL și incertitudinea.....	34
5.4 Validarea și exploatarea de încredere a modelelor	36
5.5 Modalități de monitorizare a stării echipamentelor	37
5.6 Strategii de monitorizare și implementarea distribuită.....	38
5.7 Integrarea cu procesul de mentenanță și verificarea rezultatului.....	39
6. Viziune artificială și inspecție optică industrială	40
6.1 Arhitectura stației și formularea cerinței de inspecție.....	40

6.2 Iluminarea, optica, aparatul de achiziție și calibrarea	41
6.3 Metode clasice, învățare profundă și detectarea anomaliilor	42
6.4 Datele, validarea și evaluarea performanței	44
6.5 Robustețea stației și administrarea schimbărilor	45
6.6 Integrarea cu PLC-ul, trasabilitatea și mecanismul de respingere.....	46
7. Identificarea sistemelor, control predictiv și gemeni digitali.....	48
7.1 Scopul modelului, proiectarea experimentului și domeniul de validitate	48
7.2 Structura, estimarea și validarea modelului.....	49
7.3 Simularea și validarea graduală.....	51
7.4 Controlul predictiv bazat pe model	52
7.5 Gemeni digitali și firul digital	54
7.6 Integrarea identificării, simulării, controlului și geamănului digital	55
Partea a IV-a. Siguranță, securitate și operare centrată pe om.....	56
8. Siguranța în funcționare, securitate și operare industrială centrată pe om	56
8.1 Siguranța în funcționare și comportamentul la defect	56
8.2 Redundanță, funcționare degradată și recuperare.....	58
8.3 Securitatea cibernetică industrială și reziliența ciclului de viață.....	59
8.4 Cadrul european și efectele asupra proiectării industriale.....	61
8.5 HMI, managementul alarmelor și autoritatea umană	62
8.6 Validarea operațională integrată	64
Partea a V-a. Integrarea fabricii	66
9. Robotică industrială, roboți colaborativi și sisteme mobile autonome.....	66
9.1 Sistemul robotic și performanța aplicației	66
9.2 Cinematica, dinamica și controlul interacțiunii.....	67
9.3 Percepția, prinderea bazată pe învățare și învățarea prin întărire	69
9.4 Arhitectura distribuită, integrarea și simularea.....	70
9.5 Roboți colaborativi și colaborarea om–robot	71
9.6 AMR, managementul flotei și ciclul de viață	72
10. MES, ERP și integrarea operațională	74
10.1 Rolurile PLC–SCADA–MES–ERP și autoritatea fiecărui nivel.....	74
10.2 Execuția producției: ordine, producție în curs, rețete și resurse.....	75
10.3 Date operaționale, calitate, genealogia produsului și indicatori.....	77
10.4 Integrarea orientată pe evenimente, consistența și funcționarea fără conexiune.....	78
10.5 ERP, PLM, WMS și opțiuni de implementare	79
10.6 Sprijinul decizional și urmărirea rezultatului	80
10.7 Implementarea și schimbarea organizațională.....	81
11. Fabrica inteligentă, sustenabilitate și arhitecturi industriale emergente	83
11.1 Arhitectura integrată și perspectivele de proiectare.....	83
11.2 Asset Administration Shell, firul digital și spațiile de date industriale.....	84
11.3 Măsurarea resurselor și sustenabilitatea operațională.....	85
11.4 Automatizarea definită prin software și administrarea infrastructurii industriale edge.....	87

11.5 Proiectare și configurare tehnică asistate de inteligența artificială și modele fundamentale industriale	87
11.6 Implementarea graduală a fabricii inteligente	88
Sinteza părții teoretice. Principii transversale de proiectare.....	90
Partea a VI-a — Studii de caz și aplicații integrate	94
12. Studii de caz de informatică industrială.....	94
Studiul de caz 1. Mentenanța predictivă a unui ansamblu motor–pomă	96
Studiul de caz 2. Inspecția optică automată a componentelor fabricate	97
Studiul de caz 3. Operarea eficientă a unui atelier din punct de vedere energetic și a consumurilor materiale	98
Studiul de caz 4. Integrarea verticală a unei linii de ambalare	99
Studiul de caz 5. Modernizarea securizată a unei celule de fabricație existente	100
Studiul de caz 6. Flotă de roboți mobili autonomi (AMR) pentru logistică internă.....	101
Studiul de caz 7. Celulă colaborativă pentru asamblare mecanică.....	102
Studiul de caz 8. Monitorizarea și controlul inteligent al unei clădiri industriale (BMS).....	103
Studiul de caz 9. Mobilitate urbană cooperativă cu sisteme ADAS și V2X.....	104
Studiul de caz 10. Fermă cyber-fizică: irigație de precizie, ISOBUS și utilaj autonom	105
Lucrări de laborator	106
Lucrare de laborator 1: Dezvoltare de sisteme de tip Internet of Things (IoT) cu ESP8266	106
Lucrare de laborator 2: Rețea de Senzori Fără Fir cu Raspberry Pi 5	122
Lucrare de laborator 3: Stocarea și analiza datelor de proces în baze de date Time-Series	129
Lucrare de laborator 4: Procesarea semnalelor industriale - filtrare în domeniul timp și analiză spectrală FFT pentru detecția anomaliilor	138
Lucrare de laborator 5: Integrare industrială și comunicații seriale cu PLC FX3U și Interfață HMI ..	148
Lucrare de laborator 6: Inspecție optică industrială (AOI), control dimensional și detecție de defecte în timp real	157
Lucrare de laborator 7: Simularea și programarea unei celule robotizate industriale de manipulare ..	166
Lucrare de laborator 8: Modelarea unui Geamăn Digital (Digital Twin) și Punere în Funcțiune Virtuală Hardware-in-the-Loop (HIL).....	174
Lucrare de laborator 9: Sistem MES pentru calculul automatizat al indicatorului OEE și urmărirea producției	183
Bibliografie	196

Listă de abrevieri

AAS – Asset Administration Shell
ADAS – Advanced Driver Assistance Systems
ADC – Analog-to-Digital Converter
AGV – Automated Guided Vehicle
AI – Artificial Intelligence
AMR – Autonomous Mobile Robot
AOI – Automated Optical Inspection
API – Application Programming Interface
AR – Autoregressive
ARMAX – Autoregressive Moving Average with Exogenous Inputs
ARX – Autoregressive with Exogenous Inputs
BACnet – Building Automation and Control Network
BLE – Bluetooth Low Energy
BMS – Building Management System
BOM – Bill of Materials
C-ITS – Cooperative Intelligent Transport Systems
C-V2X – Cellular Vehicle-to-Everything
CAM – Cooperative Awareness Message
CAN – Controller Area Network
CBOR – Concise Binary Object Representation
CIP – Common Industrial Protocol
CMMS – Computerized Maintenance Management System
CNC – Computer Numerical Control
CNN – Convolutional Neural Network
CoAP – Constrained Application Protocol
CPM – Collective Perception Message
CPS – Cyber-Physical System
CPU – Central Processing Unit
CRA – Cyber Resilience Act
CRC – Cyclic Redundancy Check
CSV – Comma-Separated Values
CUSUM – Cumulative Sum
DALI – Digital Addressable Lighting Interface
DCS – Distributed Control System
DDC – Direct Digital Control
DENM – Decentralized Environmental Notification Message
DGA – Dissolved Gas Analysis
DM – Deadline Monotonic
DMR – Double Modular Redundancy
DOF – Degrees of Freedom
DSS – Decision Support System
DTLS – Datagram Transport Layer Security
EDF – Earliest Deadline First
ELT – Extract, Load, Transform
eMBB – enhanced Mobile Broadband
eRedCap – enhanced Reduced Capability
ERP – Enterprise Resource Planning
ETL – Extract, Transform, Load
EWMA – Exponentially Weighted Moving Average

FBD – Function Block Diagram
FDD – Fault Detection and Diagnosis
FFT – Fast Fourier Transform
FIFO – First In, First Out
FIR – Finite Impulse Response
FMEA – Failure Mode and Effects Analysis
FOTA – Firmware Over-the-Air
FOV – Field of View
FPGA – Field-Programmable Gate Array
FSoE – Safety over EtherCAT
GNSS – Global Navigation Satellite System
GPIO – General-Purpose Input/Output
GPU – Graphics Processing Unit
GUI – Graphical User Interface
HEPA – High-Efficiency Particulate Air
HIL – Hardware-in-the-Loop
HMI – Human-Machine Interface
HTTP – Hypertext Transfer Protocol
HVAC – Heating, Ventilation, and Air Conditioning
I/O – Input/Output
I2C – Inter-Integrated Circuit
IaaS – Infrastructure as a Service
IDE – Integrated Development Environment
IDMZ – Industrial Demilitarized Zone
IIoT – Industrial Internet of Things
IIR – Infinite Impulse Response
IIRA – Industrial Internet Reference Architecture
IMU – Inertial Measurement Unit
IoT – Internet of Things
IoU – Intersection over Union
IPC – Industrial Personal Computer
IRT – Isochronous Real-Time
IT – Information Technology
JSON – JavaScript Object Notation
k-NN – k-Nearest Neighbors
KPI – Key Performance Indicator
LCA – Life Cycle Assessment
LD – Ladder Diagram
LED – Light-Emitting Diode
LiDAR – Light Detection and Ranging
LOF – Local Outlier Factor
LoRaWAN – Long Range Wide Area Network
LPWAN – Low Power Wide Area Network
LSTM – Long Short-Term Memory
LTE-M – Long-Term Evolution for Machines
LwM2M – Lightweight Machine to Machine
MAE – Mean Absolute Error
MAPEM – MAP Extended Message
MCL – Monte Carlo Localization
MCSA – Motor Current Signature Analysis
MEC – Multi-access Edge Computing
MES – Manufacturing Execution System

MIL – Model-in-the-Loop
ML – Machine Learning
mMTC – massive Machine-Type Communication
MPC – Model Predictive Control
MQTT – Message Queuing Telemetry Transport
MTBF – Mean Time Between Failures
MTTR – Mean Time To Repair
NB-IoT – Narrowband IoT
NO – Normally Open
NPU – Neural Processing Unit
NRT – Non-Real-Time
NTP – Network Time Protocol
OBU – On-Board Unit
ODE – Open Dynamics Engine
OOE – Overall Equipment Effectiveness
OPC UA – OPC Unified Architecture
OSCORE – Object Security for Constrained RESTful Environments
OT – Operational Technology
PCB – Printed Circuit Board
PCIe – Peripheral Component Interconnect Express
PFL – Power and Force Limiting
PHM – Prognostics and Health Management
PID – Proportional-Integral-Derivative
PIL – Processor-in-the-Loop
PKI – Public Key Infrastructure
PL – Performance Level
PLC – Programmable Logic Controller
PLM – Product Lifecycle Management
PRBS – Pseudo-Random Binary Sequence
PTP – Precision Time Protocol
PWM – Pulse-Width Modulation
QoS – Quality of Service
R-CNN – Region-based Convolutional Neural Network
RAM – Random Access Memory
RAMI 4.0 – Reference Architecture Model Industrie 4.0
RBAC – Role-Based Access Control
RedCap – Reduced Capability
REST – Representational State Transfer
RGB – Red Green Blue
RISC – Reduced Instruction Set Computer
RL – Reinforcement Learning
RLS – Recursive Least Squares
RM – Rate Monotonic
RMS – Root Mean Square
RMSE – Root Mean Squared Error
ROI – Return on Investment
ROS – Robot Operating System

RSU – Roadside Unit
RTK – Real-Time Kinematic
RTU – Remote Terminal Unit
RUL – Remaining Useful Life
SCADA – Supervisory Control and Data Acquisition
SCARA – Selective Compliance Assembly Robot Arm
SDA – Software-Defined Automation
SEC – Specific Energy Consumption
SEU – Significant Energy Uses
SFC – Sequential Function Chart
SIL – Software-in-the-Loop
SLAM – Simultaneous Localization and Mapping
SPaT – Signal Phase and Timing
SPI – Serial Peripheral Interface
SQL – Structured Query Language
SRM – Signal Request Message
SSID – Service Set Identifier
SSM – Signal Status Message
ST – Structured Text
STFT – Short-Time Fourier Transform
STRIDE – Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
SVM – Support Vector Machine
TCN – Temporal Convolutional Network
TCP – Transmission Control Protocol
TLS – Transport Layer Security
TMR – Triple Modular Redundancy
TPM – Total Productive Maintenance
TSDB – Time-Series Database
TSN – Time-Sensitive Networking
UAEX – OPC UA Field eXchange
UART – Universal Asynchronous Receiver/Transmitter
UDP – User Datagram Protocol
URI – Uniform Resource Identifier
URLLC – Ultra-Reliable Low-Latency Communication
USB – Universal Serial Bus
V2X – Vehicle-to-Everything
VOC – Volatile Organic Compounds
VPN – Virtual Private Network
WCET – Worst-Case Execution Time
WIP – Work in Progress
WMS – Warehouse Management System
WSN – Wireless Sensor Network
XML – Extensible Markup Language
YOLO – You Only Look Once

Prefață

Informatica industrială îmbină concepte și metode din automatizare, tehnologia informației, comunicații, ingineria controlului, managementul datelor și exploatarea proceselor fizice. Între 2015 și 2026, domeniul a fost transformat de integrarea sistemelor cyber-fizice, a Internetului Industrial al Lucrurilor (IIoT), a platformelor edge și cloud, a rețelelor industriale moderne și a inteligenței artificiale. În aceeași perioadă, securitatea cibernetică, siguranța, disponibilitatea, interoperabilitatea și eficiența energetică au devenit cerințe fundamentale, iar trecerea spre o industrie centrată pe om, rezilientă și sustenabilă (Industry 5.0) reconfigurează relația dintre operator, mașină și sistemul decizional.

Cartea oferă o prezentare unitară a conceptelor, arhitecturilor și tehnologiilor care definesc informatica industrială contemporană. Fundamentele – sistemele PLC, DCS și SCADA, execuția în timp real, procesarea semnalelor – sunt păstrate, deoarece ele susțin arhitecturile industriale actuale. Accentul se deplasează însă către transformările ultimului deceniu: conectivitatea IIoT și distribuirea funcțiilor în continuumul dispozitiv–edge–fabrică–cloud, comunicațiile deterministe și interoperabilitatea semantică, administrarea datelor industriale, mentenanța predictivă și viziunea artificială, controlul predictiv și gemenii digitali, robotica colaborativă și integrarea sistemelor de producție cu celelalte servicii informatice ale unei companii (ex. planificare, achiziție/vanzări, marketing, evidența personalului, management, etc.).

Lucrarea de față reprezintă o completare și o actualizare a suportului de curs utilizat până acum [1] la disciplina Informatică Industrială: păstrează structura didactică validată în timp, dar revizuieste conținutul în raport cu evoluțiile tehnologice și normative ale ultimului deceniu, adaugă teme apărute între timp — procesarea la edge, mentenanța predictivă, gemenii digitali, securitatea cibernetică industrială, robotica colaborativă — și propune o parte aplicativă complet regândită.

Lucrarea nu urmărește o descriere exhaustivă a protocoalelor, algoritmilor sau produselor comerciale. Tehnologiile sunt analizate prin prisma rolului lor într-un sistem industrial complet și prin consecințele pe care le au deciziile de proiectare. Alegerea unei platforme de calcul, a unui protocol de comunicație sau a unui model de inteligență artificială este raportată la timpul de răspuns, calitatea datelor, disponibilitatea infrastructurii, consecințele unei defectări, cerințele de securitate și capacitatea organizației de a exploata și întreține soluția. Din acest motiv, cartea urmărește constant lanțul care leagă procesul fizic de funcția de control, infrastructura informațională și nivelul decizional.

Conținutul reunește trei componente complementare. Partea teoretică acoperă fundamentele sistemelor industriale, infrastructurile de comunicație și de date, metodele de analiză și control, securitatea, siguranța în funcționare, interacțiunea om–sistem și integrarea operațională. Studiile de caz ilustrează aplicarea acestor principii în contexte industriale diverse, iar lucrările de laborator propun activități practice de analiză, configurare, implementare și evaluare.

Cartea se adresează studenților, cadrelor didactice, cercetătorilor, inginerilor și practicienilor implicați în proiectarea și integrarea sistemelor informatice industriale moderne. Fundamentarea solidă, împreună cu tehnologiile dezvoltate sau consolidate între 2015 și 2026, oferă un cadru coerent pentru înțelegerea, analiza și dezvoltarea aplicațiilor actuale de informatică industrială.

Partea I. Fundamentele informaticii industriale

1. Informatica industrială: evoluție, concepte și cerințe ingineresti

Sistemele informatice industriale nu sunt simple extensii ale aplicațiilor IT. Ele controlează procese fizice reale – linii de producție, rețele electrice, instalații chimice – unde o eroare de calcul sau o întârziere de o fracțiune de secundă poate avea consecințe grave. Conectarea echipamentelor și folosirea unor platforme de calcul performante nu aduc, prin ele însele, valoare. Ceea ce contează este ca soluția să respecte, simultan, cerințe stricte de timp de răspuns, siguranță, disponibilitate, securitate, interoperabilitate și mentenabilitate, pe o durată de exploatare care poate atinge decenii. O funcție software poate fi logic impecabilă și totuși inutilă, dacă rezultatul sosește după momentul în care procesul avea nevoie de el. La fel, transferul de date către o platformă externă poate îmbunătăți vizibilitatea asupra producției, dar poate crea noi dependențe și riscuri, dacă interfețele, responsabilitățile și limitele de securitate nu sunt definite explicit [2], [3], [4].

Acest capitol fixează cadrul conceptual al cărții: definește domeniul informaticii industriale, prezintă evoluția de la automatizarea clasică la industria conectată și la paradigma Industry 5.0, iar apoi identifică cerințele transversale care străbat întreaga lucrare.

1.1 Domeniul și specificul informaticii industriale

Informatica industrială este disciplina inginerescă ce se ocupă de achiziția, reprezentarea, comunicarea, procesarea, protejarea și utilizarea operațională a informației asociate proceselor industriale. Ea împrumută concepte și metode din automatizări, sisteme încorporate, comunicații, inginerie software, teoria controlului, managementul datelor, inteligență artificială, securitate cibernetică și sistemele informaționale de întreprindere. Caracterul său interdisciplinar derivă din nevoia de a corela comportamentul aplicațiilor software cu dinamica echipamentelor, materialelor și proceselor asupra cărora acestea acționează.

Ceea ce diferențiază un sistem informatic industrial de unul informațional convențional este legătura directă cu procesul fizic. Sensorii măsoară starea echipamentelor, a produselor și a mediului. Software-ul preia aceste măsurări și generează comenzi, alarme, recomandări, decizii de producție sau acțiuni de mentenanță. Prin intermediul actuatorilor și al sistemelor de control, aceste rezultate modifică, la rândul lor, procesul. Se închide astfel o buclă continuă între lumea fizică și reprezentarea ei digitală: starea procesului dictează decizia software, iar decizia software modelează starea viitoare a procesului. Această interacțiune bidirecțională este o caracteristică definitorie a sistemelor cyber-fizice, a căror funcționare este detaliată în capitolul 2 [5], [6].

Informația industrială este distribuită pe niveluri tehnice și organizaționale, fiecare cu responsabilități, autorități și ritmuri de operare distincte: de la interacțiunea directă cu procesul și controlul local, până la supravegherea instalației, coordonarea producției și administrarea întreprinderii [7]. Rolurile sistemelor PLC (*Programmable Logic Controller*), DCS (*Distributed Control System*) și SCADA (*Supervisory Control and Data Acquisition*) sunt detaliate în capitolul 2, iar integrarea dintre MES (*Manufacturing Execution System*) și ERP (*Enterprise Resource Planning*) este tratată în capitolul 10.

Calitatea informației are consecințe directe asupra funcționării fizice și economice. Un marcaj temporal (*timestamp*) incorect poate bloca reconstituirea succesiunii de evenimente; o măsurare întârziată poate duce la o decizie de control nepotrivită; absența unui identificator de produs poate rupe lanțul de trasabilitate. În plus, o comandă emisă de un utilizator neautorizat, sau calculată pe baza unor date compromise, poate pune în pericol personalul, calitatea produsului, mediul sau disponibilitatea

instalației. De aceea, calitatea datelor, sincronizarea temporală, proveniența informației, controlul accesului, diagnosticarea și administrarea ciclului de viață nu sunt funcții adăugate ulterior implementării, ci proprietăți care se proiectează, se verifică și se mențin pe întreaga durată de exploatare [2], [3].

1.2 De la automatizarea programabilă la industria conectată

Evoluția către industria conectată devine mai clară dacă pornim de la fundamentele etapei anterioare. Etapa consacrată drept Industria 3.0 a marcat introducerea la scară largă a electronicii, a calculatoarelor de proces și a controlerelor logice programabile (PLC). Aceste tehnologii, alături de sistemele de control distribuit (DCS) și roboții industriali, au asigurat salturi majore în precizie și productivitate, însă au lăsat în urmă numeroase „insule de automatizare” izolate, dependente de protocoale proprietare.

Pentru a depăși această fragmentare și pentru o organizare mai bună a fluxurilor de informații, industria a adoptat ierarhia standardizată definită prin modelul Purdue și standardul ANSI/ISA-95. Această arhitectură a structurat fabrica pe niveluri funcționale distincte: de la procesul fizic și senzori (Nivelul 0), controlul în timp real (Nivelul 1) și supravegherea SCADA (Nivelul 2), până la execuția producției în MES (Nivelul 3) și planificarea resurselor în ERP (Nivelul 4). Pe această fundație ierarhică s-a grefat ulterior saltul către Industria 4.0, care a transformat fluxurile piramidale rigide într-o rețea flexibilă de sisteme cyber-fizice, edge computing și comunicații industriale interoperabile.

Industria 4.0 a extins această bază prin comunicații industriale generalizate, sisteme cyber-fizice, dispozitive IIoT, platforme edge și cloud și forme distribuite de inteligență. Transformarea nu înseamnă doar conectarea mașinilor la o rețea, ci capacitatea echipamentelor, a aplicațiilor de producție, a sistemelor de mentenanță și a aplicațiilor întreprinderii de a schimba informații contextualizate pe tot parcursul ciclului de viață al activelor și al produselor. Datele despre configurație, stare, producție, calitate și mentenanță pot fi corelate, susținând decizii care depășesc limitele unei singure mașini sau ale unui singur departament [5], [6], [8].

O instalație IIoT trebuie privită ca un ansamblu de sisteme interconectate (*system of systems*), nu ca o simplă colecție de senzori conectați la internet [8]. Dispozitivele de câmp („*field devices*”), controlerele, echipamentele de interconectare, brokerii de mesaje, bazele de date, serviciile analitice și sistemele informatice ale întreprinderii pot proveni de la furnizori diferiți și pot evolua în ritmuri diferite. Pentru a coopera, ele au nevoie de identificarea unică a echipamentelor, de modele informaționale coerente, de sincronizare temporală, de comunicații securizate și de o administrare riguroasă a versiunilor. Totodată, trebuie să se știe precis cine are dreptul să emită o comandă și cum se validează acea decizie înainte de a produce un efect fizic [8], [9].

Cerințele operaționale și durata lungă de viață diferențiază IIoT-ul de majoritatea aplicațiilor IoT destinate consumatorilor. O instalație industrială poate funcționa zeci de ani; bibliotecile software, sistemele de operare, protocoalele și furnizorii de platforme digitale se schimbă însă mult mai repede. Arhitectura trebuie, așadar, să permită actualizări validate, modificări controlate ale configurației și integrarea echipamentelor mai vechi, fără ca fiecare schimbare tehnologică să provoace o întrerupere majoră a producției [10]. Conform regulii autonomiei locale (secțiunea 2.1), pierderea conexiunii cu serviciile centrale, cu platformele cloud sau cu rețeaua externă nu trebuie să împiedice funcțiile locale esențiale să mențină procesul într-o stare stabilă sau sigură [3], [8], [11]. Criteriile de repartizare a funcțiilor de-a lungul continuumului dispozitiv–edge–fabrică–cloud, împreună cu rolurile specifice fiecărui nivel, sunt dezvoltate în detaliu în capitolul 3 [11], [12], [13].

Conectivitatea industrială își dovedește valoarea doar atunci când se reflectă în rezultate operaționale măsurabile. Succesul unei inițiative nu se măsoară prin numărul de dispozitive conectate sau prin volumul brut de informație colectată, ci prin efectele concrete asupra procesului. O arhitectură mai complexă se justifică numai dacă beneficiile ei pot fi demonstrate și menținute pe întreaga durată de exploatare [14], [15], [16].

1.3 De la Industria 4.0 la Industria 5.0

Industria 5.0 nu înlocuiește tehnologiile dezvoltate în contextul Industriei 4.0 și nu presupune o revenire la procese mai puțin automatizate [17]. Conceptul extinde însă criteriile după care proiectăm și evaluăm sistemele industriale. Productivitatea, flexibilitatea și conectivitatea rămân repere importante, dar ele sunt analizate acum alături de trei dimensiuni complementare: orientarea către om, reziliența și sustenabilitatea [18]. Scopul nu mai este automatizarea cu orice preț, ci utilizarea tehnologiei pentru a construi sisteme care sprijină activitatea umană, se adaptează perturbărilor și funcționează în limite economice, sociale și de mediu acceptabile [18], [19].

Orientarea către om presupune o distribuție conștientă a funcțiilor și a responsabilităților între operator și sistemul automat. Tehnologia poate prelua operațiile repetitive, periculoase sau care cer viteze de reacție imposibile pentru om. Oamenii contribuie prin interpretarea contextului, prin gestionarea situațiilor neprevăzute, prin formularea obiectivelor și prin asumarea răspunderii pentru deciziile care nu pot fi delegate în siguranță. În anumite aplicații, o soluție de sprijin decizional, în care operatorul păstrează autoritatea finală, se dovedește mai robustă decât delegarea completă către un algoritm [19], [21].

Consecințele concrete ale acestei orientări diferă de la un domeniu la altul [20]. În sistemele HMI (*Human-Machine Interface*) și SCADA, ea înseamnă o prezentare clară a stării procesului, a calității datelor și a acțiunilor disponibile. În robotica colaborativă, implică proiectarea și validarea întregii interacțiuni dintre om, robot, unealtă, piesă și spațiul de lucru. Iar în aplicațiile bazate pe inteligență artificială, operatorul trebuie să poată înțelege semnificația rezultatului, nivelul de încredere, limitele modelului și momentul în care intervenția sa devine necesară. Aceste cerințe sunt dezvoltate în capitolele dedicate operării centrate pe om, inteligenței artificiale industriale și roboticii colaborative [21], [22], [23], [24], [25].

Reziliența descrie capacitatea unui sistem de a-și păstra funcțiile esențiale și de a reveni controlat după o perturbare. Sustenabilitatea lărgeste evaluarea, incluzând utilizarea resurselor, impactul asupra mediului și ciclul de viață al echipamentelor [26]. Împreună cu orientarea către om, aceste dimensiuni completează criteriile tradiționale de productivitate, flexibilitate și conectivitate [27]. Principiile tehnice ale rezilienței sunt dezvoltate în capitolul 8, iar măsurarea resurselor și sustenabilitatea operațională sunt tratate în secțiunea 11.3 [16], [20].

1.4 Cerințe ingineresti transversale

Proiectarea unui sistem industrial impune satisfacerea simultană a mai multor categorii de cerințe, cu riscul ca îmbunătățirea uneia să o degradeze pe alta [28]. De exemplu, creșterea conectivității sporește vizibilitatea asupra procesului, dar lărgeste și suprafața potențială de atac cibernetic. Procesarea în cloud reduce necesarul local de calcul, însă introduce latență și dependență de infrastructura externă. Redundanța crește disponibilitatea, dar sporește complexitatea, efortul de testare și riscul stărilor inconsistente. Cerințele nu trebuie, așadar, maximizate izolat, ci formulate în raport cu funcția sistemului, cu gravitatea unui eventual defect și cu condițiile reale de exploatare. Tabelul 1.1 sintetizează cele șase proprietăți transversale care ghidează întreaga lucrare.

Proprietățile discutate sunt interdependente. Locul de execuție al unei funcții, protocolul ales sau utilizarea unui model de inteligență artificială influențează simultan timpul de răspuns, disponibilitatea, interoperabilitatea, securitatea, mentenabilitatea și responsabilitatea operatorului. Din acest motiv, un sistem industrial nu se evaluează pe componente izolate, ci pe întregul lanț care leagă măsurarea, prelucrarea, decizia și acțiunea fizică. Repartizarea funcțiilor trebuie să se bazeze pe cerințele procesului, pe autoritatea fiecărui nivel și pe consecințele unei funcționări incorecte – nu doar pe disponibilitatea unei anumite tehnologii.

Tabelul 1.1. Cerințe transversale pentru sistemele informatice industriale

Proprietate	Întrebarea principală de proiectare	Elemente care trebuie definite
Timp și determinism	Rezultatul este disponibil la momentul cerut de proces?	Termen-limită (<i>deadline</i>), timp maxim de calcul (WCET - <i>Worst-Case Execution Time</i>), perioadă, variație temporală (<i>jitter</i>) și sincronizare de precizie [2].
Siguranța în funcționare și recuperare	Cum reacționează sistemul la apariția defectelor?	Detecție și izolare a defectelor, stare sigură (<i>fail-safe</i>), mod degradat de lucru, redundanță și recuperare coerentă [29].
Interoperabilitate semantică	Pot nodurile eterogene să schimbe informații în mod consecvent?	Modele informaționale standardizate (OPC UA - <i>OPC Unified Architecture</i>), unități de măsură, marcaje temporale unificate și drepturi de comandă [7], [30].
Securitate cibernetică	Este protejat procesul fizic fără a bloca disponibilitatea?	Segmentare pe zone și conducte (IEC 62443), autentificare, inventar de active OT (<i>Operational Technology</i>), jurnalizare și proceduri de backup [3], [4], [31].
Mentenabilitate și ciclu de viață	Poate fi sistemul diagnosticat, extins și restaurat facil?	Arhitectură modulară, interfețe documentate, managementul versiunilor, proceduri de calibrare și retragere securizată [3], [8], [9].
Inteligență artificială de încredere	Este modelul robust și verificabil în regim operațional?	Date reprezentative, timp garantat de inferență, filtrarea alarmelor false, monitorizarea derivei (<i>drift</i>) și autoritate umană [24], [25], [32].

2. Automatizare industrială, sisteme cyber-fizice și execuție în timp real

Automatizarea industrială modernă se bazează pe cooperarea unor sisteme care lucrează la viteze diferite și cu responsabilități distincte. Controlerele logice programabile (PLC) execută controlul mașinilor, al secvențelor și al celulelor de producție. Sistemele de control distribuit (DCS) coordonează procese continue sau pe loturi (*batch processing*). Platformele SCADA asigură supravegherea, gestiunea alarmelor și păstrarea istoricului operațional. La nivelurile superioare, sistemele MES și aplicațiile informatice ale întreprinderii folosesc datele de producție pentru trasabilitate, planificare, managementul calității și administrarea resurselor. Aceste patru categorii nu sunt variante concurente; într-o instalație complexă, ele formează nivelurile complementare ale aceleiași arhitecturi [3], [7], [33].

Capitolul analizează modul în care funcțiile de control, supraveghere și diagnosticare se distribuie între PLC, DCS și SCADA, apoi examinează cerințele de timp, criteriile de alegere a platformei de execuție și reluarea controlată a secvențelor după o oprire sau defecțiune.

2.1 Arhitecturi de automatizare: PLC, DCS și SCADA

Controlerul logic programabil (PLC) este un calculator industrial conceput să execute funcții de control în medii severe, caracterizate prin temperaturi extreme, vibrații, perturbații electromagnetice și cerințe ridicate de disponibilitate. Ciclul său clasic de funcționare citește intrările fizice și datele primite prin comunicație, execută programul, actualizează ieșirile și rulează serviciile interne de diagnostic. Acest ciclu trebuie corelat cu dinamica procesului: secvențele obișnuite tolerează variații de milisecunde, dar controlul mișcării, sincronizarea axelor sau protecțiile rapide impun limite mult mai stricte. Pentru aceste funcții, timpul maxim de scanare în cele mai solicitante condiții este mult mai relevant decât media [2], [33].

Programele PLC sunt scrise, de regulă, în unul din cele 5 limbaje standardizate prin IEC 61131-3 [34]: *Structured Text* (ST), limbaj textual de nivel înalt, se potrivește algoritmilor și manipulării structurilor de date complexe; *Ladder Diagram* (LD), un limbaj grafic destinat electricienilor, care redă interblocările într-o formă apropiată de schemele electrice cu contacte; *Function Block Diagram* (FBD), un limbaj grafic, care compune funcțiile aplicației prin blocuri conectate grafic; *Sequential Function Chart* (SFC) descrie etapele și tranzițiile unui proces secvențial; iar *Instruction List* (IL) este un limbaj textual asemănător limbajului de asamblare. Alegerea limbajului ține de natura funcției, de complexitatea algoritmului, de experiența echipei de mentenanță și de ușurința testării și diagnosticării. O implementare compactă poate deveni greu de urmărit în timpul unei opriri; o structură modulară, cu stări și interfețe explicite, reduce semnificativ timpul de diagnosticare [34].

PLC-urile actuale integrează servicii OPC UA, MQTT (*Message Queuing Telemetry Transport*), funcții de monitorizare a stării echipamentelor, conectivitate edge și, în unele cazuri, execuție virtualizată. Aceste extensii îmbogățesc schimbul de informații, însă nu modifică un principiu fundamental: controlul imediat, interblocările, protecțiile și funcțiile de aducere în stare sigură trebuie să rămână disponibile fără a depinde de SCADA, de brokerii de mesaje, de platformele edge sau de serviciile cloud. Această regulă a autonomiei locale – valabilă și pentru buclele DCS – limitează autoritatea comenzilor de supraveghere. Pentru controlul virtualizat, trebuie verificate suplimentar comportamentul temporal, izolarea resurselor, accesul la I/O și reacția la pierderea infrastructurii de calcul [33], [35].

PLC-urile pot fi compacte sau modulare și pot include funcții de siguranță, control al mișcării și mecanisme de redundanță. Într-o configurație redundantă, trebuie definite detectarea defectului, sincronizarea stării între canale, condițiile de comutare și starea programului după comutare. Independența canalelor și riscul defectelor de cauză comună sunt tratate în secțiunea 8.2 [36].

În timp ce PLC-ul predomină în automatizarea mașinilor, sistemele de control distribuit (DCS) reprezintă soluția de referință pentru procesele continue și pe loturi, acolo unde numeroase bucle de reglare interacționează în industrii precum cea chimică, energetică, farmaceutică sau în tratarea apei [37]. Caracterul „distribuit” se referă în primul rând la partajarea responsabilităților: controlerele locale execută reglarea, secvențele și interblocările; stațiile de operare afișează starea și alarmele; stațiile de configurare administrează parametrii; sistemul de arhivare păstrează datele pentru analiză [38].

Pentru un DCS, trebuie definite din proiectare efectele indisponibilității fiecărui serviciu superior: ce informații dispar, ce valori sunt păstrate sau înlocuite, cât timp poate funcționa sistemul în mod degradat și ce operații se suspendă. Integrarea oferită de un DCS poate reduce efortul de proiectare și asigura coerența între control, alarmare și arhivare, însă creează frecvent o dependență semnificativă de ecosistemul furnizorului. Compatibilitatea pe termen lung, posibilitățile de migrare, costul licențelor și actualizările de securitate trebuie evaluate împreună cu performanța nominală [36], [37], [38]. În același timp, baza comună de date de configurare, serviciile de identitate și de timp și infrastructura de rețea constituie dependențe critice pentru funcțiile de coordonare și supraveghere, iar analiza punctelor unice de defectare și a cauzelor comune este tratată în secțiunea 8.2 [3], [36].

Sistemele SCADA operează la nivel de supraveghere. Ele colectează stări și măsurări, gestionează alarmele, păstrează istoricul operațional și transmit comenzi de supraveghere către nivelurile inferioare. O arhitectură SCADA poate cuprinde instrumente de câmp, PLC-uri, RTU-uri (*Remote Terminal Unit*), servere de aplicație și interfețe către MES sau sistemele informatice ale întreprinderii [39]. Domeniile de utilizare sunt vaste: fabrici, dar și infrastructuri distribuite – rețele electrice, conducte, sisteme de apă și transport [3], [40], [41], [42]. Comenzile transmise trebuie însă limitate la nivelul de supraveghere: modificarea unei referințe, selectarea unui mod, pornirea unei secvențe, iar acceptarea și executarea rămân în responsabilitatea controlerului local. Interfața SCADA trebuie, de asemenea, să semnalizeze explicit când comunicația este pierdută și cât de actuale sunt datele; consecințele afișării datelor învechite sunt discutate în secțiunea 8.5 [2], [3].

Modelele clasice Purdue și IEC 62264 delimitează procesul fizic, controlul, supravegherea, managementul operațiunilor și funcțiile informatice ale întreprinderii. Această delimitare își păstrează valoarea pentru că fixează responsabilitățile și proprietatea informației. Arhitecturile actuale nu mai sunt strict ierarhice – OPC UA, MQTT și platformele edge permit fluxuri directe între niveluri – însă aceste fluxuri trebuie să traverseze interfețe definite, zone de securitate și echipamente de interconectare administrate. Accesul direct și necontrolat la variabilele unui PLC poate eluda atât responsabilitățile arhitecturale, cât și mecanismele de securitate [3], [4], [7], [35], [43]. Tabelul 2.1 sintetizează poziționarea comparativă a acestor sisteme.

Delimitările din tabel sunt orientative. PLC-urile moderne pot executa și control de proces și servicii edge; un DCS poate utiliza controlere similare PLC-urilor; un SCADA poate integra ambele categorii. Selecția pornește de la proces, de la timpul de răspuns, de la distribuția activelor și de la competența organizației. Costul total include nu doar componentele fizice și licențele, ci și instrumentele de proiectare, formarea personalului, actualizările și opririle necesare intervențiilor [33], [37], [38], [39].

Tabelul 2.1. Poziționarea comparativă a sistemelor PLC, DCS și SCADA

Caracteristică	PLC	DCS	SCADA
Funcție principală	Controlul mașinilor, secvențelor logice și mișcării	Control integrat al proceselor continue și pe loturi	Supraveghere, alarmare și dispecerizare
Domeniu tipic	Mașină individuală sau celulă	Instalație completă sau unitate de proces	Fabrică, rețele geografice și active distribuite
Relația cu procesul	Directă (I/O determinist rapid)	Directă (prin controlere distribuite)	Supraveghere (prin PLC/RTU)
Scară temporală dominantă	Milisecunde–secunde	Zeci de milisecunde–secunde	Secunde–minute
Funcționarea fără nivelul superior	Obligatorie (autonomie totală locală)	Obligatorie pentru buclele locale	Păstrează autonomia controlului local
Servicii caracteristice	I/O, interblocări, secvențe, motion control	Reglare buclă închisă, rețete batch, arhivă integrată	HMI, alarme, istoric, control de la distanță

2.2 Sistemele cyber-fizice și lanțul temporal complet

În timp ce arhitecturile tradiționale PLC, DCS și SCADA au fost proiectate inițial ca structuri ierarhice și relativ izolate, convergența IT/OT le transformă astăzi în veritabile Sisteme Cyber-Fizice (CPS). Un CPS integrează măsurarea, comunicația deterministă, calculul avansat și acționarea mecatronică într-o buclă strânsă de reacție care modifică direct dinamica procesului fizic. Relația este bidirecțională: starea fizică a procesului dictează decizia software în timp real, iar comenzile software generează starea viitoare a instalației. În acest context, o eroare aparent minoră de ordin informațional – precum un marcaj temporal decalat sau un mesaj duplicat pe rețea – nu produce doar o excepție software, ci poate genera consecințe fizice reale asupra utilajelor și siguranței [5], [6]. Figura 2.1 ilustrează acest lanț funcțional [44].

Fiecare etapă a lanțului introduce întârzieri și incertitudini [44]. Senzorii prezintă zgomot, precizie finită și erori de calibrare. Rețelele introduc latență, jitter și pierderi de pachete. Execuția software este influențată de întreruperi, planificarea sarcinilor și concurența pentru resurse. Actuatorele au propria dinamică. Corectitudinea sistemului se evaluează pe întregul lanț: o comandă perfect calculată poate deveni inadecvată dacă folosește o măsurare învechită; un actuator rapid nu compensează o decizie sosită prea târziu. În rețelele de senzori fără fir, tehnici precum difuzia dirijată (*directed diffusion*) au demonstrat cum rutarea și agregarea datelor se pot adapta dinamic la topologia rețelei [45].

Un aspect adesea neglijat este diferența dintre vechimea informației și latența comunicației. Un mesaj poate traversa rapid rețeaua și, totuși, să transporte o valoare staționată îndelung într-o memorie tampon. Marcajul temporal trebuie să reflecte momentul observației fizice, nu momentul transmiterii. În sistemele distribuite, deriva ceasurilor, asimetria căilor de comunicație și perioadele de eșantionare diferite pot face ca două valori aparent simultane să descrie momente diferite ale procesului.

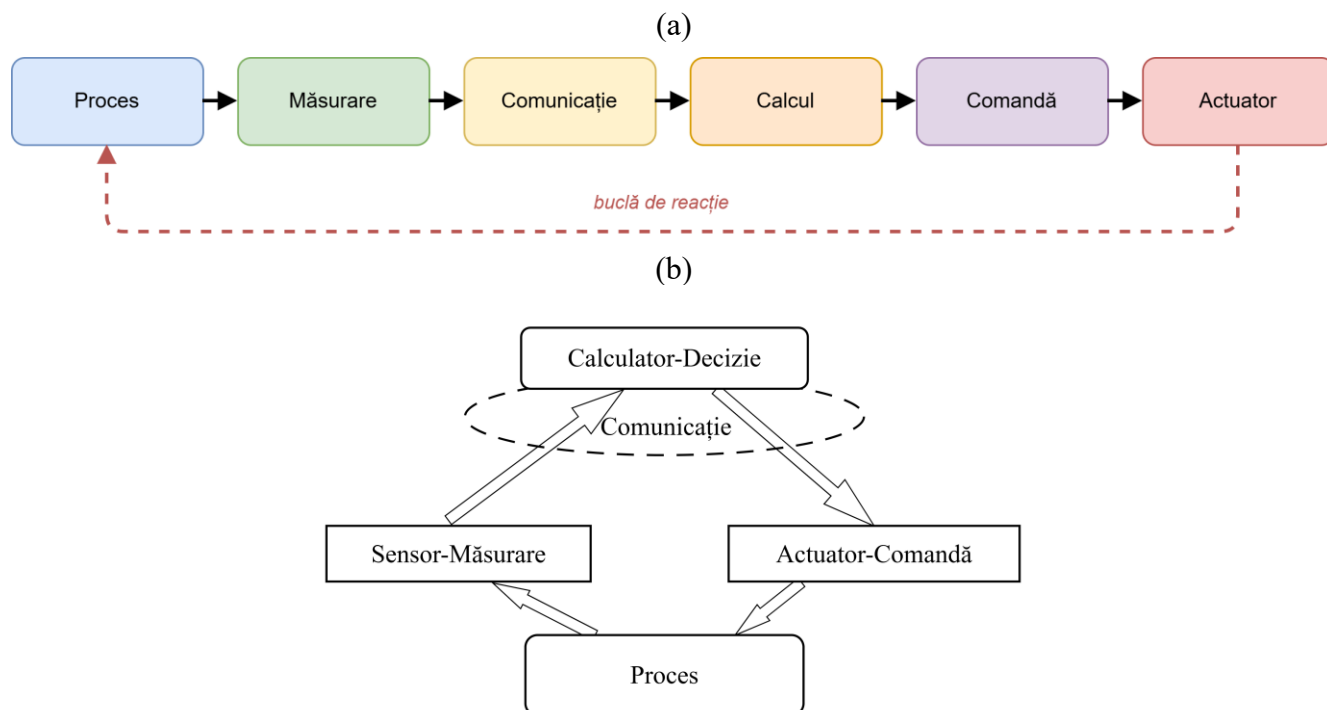


Figura 2.1. Structura funcțională a unui sistem cyber-fizic: (a) lanțul de procesare și (b) bucla de reacție

Concluzia practică este că o funcție care influențează direct procesul trebuie executată acolo unde respectarea termenului-limită poate fi garantată și unde întreruperea comunicației nu invalidează acțiunea – o consecință directă a regulii autonomiei locale. Criteriile complete de repartizare între niveluri sunt dezvoltate în capitolul 3 [11], [12].

2.3 Cerințe de funcționare în timp real și planificarea execuției

Pentru multe funcții industriale, întârzierile nu reprezintă doar o problemă de performanță, ci o componentă a corectitudinii. Un sistem în timp real este corect numai dacă furnizează rezultatul logic în intervalul specificat. Termenul-limită nu este o țintă opțională de optimizare, ci o parte integrantă a cerinței funcționale. O sarcină executată la sute de milisecunde poate fi de timp real strict dacă depășirea este inacceptabilă; o aplicație care răspunde în microsecunde dar produce ocazional întârzieri mari nu poate oferi garanții de timp real [2]. Tabelul 2.2 sintetizează principalele clase de cerințe temporale, cu exemple concrete.

Tabelul 2.2. Categoriile de cerințe de funcționare în timp real

Clasă	Consecința depășirii termenului-limită de execuție	Exemple
Timp real strict	Defectare critică sau avarie a funcției	Protecții rapide, controlul mișcării, sincronizare axe
Timp real ferm	Rezultatul întârziat este complet inutil (fără daune)	Comandă de respingere piesă pe conveyer de sortare
Timp real flexibil	Serviciul este degradat, dar rămâne util	Actualizare grafice HMI, telemetrie, rapoarte de schimb

O platformă industrială rulează un amestec eterogen de sarcini – control, comunicație, vizualizare, jurnalizare, diagnostic – care concurează pentru procesor, memorie și I/O. Arhitectura trebuie să împiedice funcțiile necritice să perturbe funcțiile cu termene-limită stricte, prin stabilirea de priorități, separarea resurselor sau executarea pe componente hardware distincte.

În ingineria sistemelor de timp real, evaluarea determinismului și a gradului de încărcare a procesorului se fundamentează pe modelul clasic de sarcină (task model), în care fiecare activitate periodică sau sporadică este definită prin setul de parametri:

$$\tau_i = (T_i, C_i, D_i, J_i, P_i)$$

Semnificația acestor parametri determină direct comportamentul la execuție:

- T_i (Perioada): intervalul nominal sau durata minimă dintre două activări consecutive ale sarcinii;
- C_i (WCET – *Worst-Case Execution Time*): timpul maxim de calcul pe procesor, determinat pe calea critică fără întreruperi;
- D_i (Termenul-limită / *Deadline*): durata maximă admisibilă de la declanșare până la furnizarea rezultatului valid ($D_i \leq T_i$);
- J_i (Variația temporală / *Jitter*): abaterea maximă a momentului efectiv de lansare a sarcinii față de ceasul de referință;
- P_i (Prioritatea): rangul alocat sarcinii în schemele de planificare preemptivă (statică sau dinamică).

Dimensionarea corectă a acestui model este esențială: o subestimare a parametrului C_i sau ignorarea fenomenului de inversare de prioritate la accesul pe resurse partajate compromise garanțiile de timp real, conducând la ratarea termenelor-limită în producție [2].

Dintre politicile de planificare, cele cu priorități fixe – *Rate Monotonic* (RM) și *Deadline Monotonic* (DM) – sunt populare în sistemele încorporate tocmai pentru că se pretează la o analiză relativ simplă. RM acordă prioritate sarcinilor cu perioada mai scurtă; DM, celor cu deadline-ul relativ mai mic. *Earliest Deadline First* (EDF) folosește priorități dinamice și poate utiliza procesorul mai eficient, dar comportamentul la suprasarcină trebuie atent evaluat. În practica industrială sunt frecvente arhitecturile mixte: controlul și comunicația ciclică urmează o planificare stabilă, iar funcțiile de diagnostic și alarmare sunt activate dinamic [2].

Timpul total de răspuns al unei sarcini include, pe lângă execuția proprie, interferența sarcinilor mai prioritare, blocajele la resursele partajate și întârzierile cauzate de întreruperi. Inversarea priorităților – când o sarcină cu prioritate ridicată așteaptă o resursă deținută de una cu prioritate inferioară – este limitată prin mecanisme precum moștenirea priorității (*priority inheritance*) și protocoalele de plafonare (*priority ceiling*). În sistemele multi-core, se adaugă interferența prin memoria cache partajată și magistrala de memorie [2].

Predictibilitatea memoriei este la fel de importantă ca planificarea procesorului. Codul cu constrângeri de timp real trebuie să evite paginarea la cerere, alocările dinamice fără limite și mecanismele automate de eliberare a memoriei („*garbage collection*”) cu durată impredictibilă. Soluțiile tipice includ alocarea statică, zonele de memorie prealocate și blocarea paginilor în memoria fizică. Alocarea dinamică nu este exclusă, dar timpul ei maxim și comportamentul la epuizarea memoriei trebuie cunoscute din proiectare.

2.4 Alegerea platformei hardware și a sistemului de operare pentru funcționare în timp real

Constrângerile de timp real, izolare și predictibilitate ridică problema alegerii platformei hardware și a sistemului de operare. Răspunsul depinde de specificul aplicației. Un microcontroler care citește un senzor și comandă o supapă are cerințe complet diferite față de un calculator industrial care combină controlul cu comunicații complexe, baze de date și inferență AI. O putere brută de calcul mai mare nu garantează un comportament temporal predictibil; o platformă ultra-deterministă poate fi nepotrivită dacă nu oferă stiva de comunicații și instrumentele software necesare. În rețelele de senzori fără fir, TinyOS a demonstrat fezabilitatea unei arhitecturi minimaliste, bazate pe componente, pentru resurse extrem de limitate [46]. Tabelul 2.3 oferă o imagine de ansamblu asupra principalelor platforme de execuție cu relevanță industrială.

Tabelul 2.3. Exemple orientative de platforme de execuție

Platformă	Domeniu caracteristic	Considerații principale
FreeRTOS sau Zephyr [47], [48]	Microcontrolere, senzori inteligenți, noduri TinyML	Consum ultra-redus, amprentă minimă de memorie, determinism
QNX sau VxWorks [49]	Sisteme critice ce impun izolare și certificare	Microkernel certificat, predictibilitate, suport comercial
Linux cu PREEMPT_RT	Calculatoare industriale (IPC), edge, viziune, AI	Ecosistem software bogat; impune configurare și validare WCET
FPGA, unitate de acționare sau controler dedicat	Controlul mișcării, sincronizare, bucle ultra-rapide	Latență minimă (sub μs) și jitter nul; flexibilitate software redusă [50]

O opțiune aparte, care merită o discuție mai amplă, este Linux cu patch-ul PREEMPT_RT. Acesta transformă nucleul într-un mediu mult mai predictibil, aducând sub controlul planificatorului cât mai multe activități interne: secțiunile de nucleu neîntreruptibile sunt reduse, întreruperile sunt convertite în fire de execuție planificabile și se folosesc mecanisme de blocare compatibile cu moștenirea priorității. PREEMPT_RT devine atractiv când funcțiile sensibile la timp coexistă cu sisteme de fișiere, containere, comunicații complexe sau aplicații de analiză [51], [52].

Totuși, validarea temporală a unui sistem Linux cu PREEMPT_RT nu este însă o formalitate. Ea trebuie să includă afinitatea firelor de execuție și a întreruperilor, izolarea nucleelor de procesor, blocarea paginilor în memorie, dezactivarea stărilor de economisire a energiei și analiza activității driverelor. Configurația testată trebuie să fie identică cu cea din exploatare. Aceste măsuri reduc interferențele, dar nu fac din Linux o soluție universală. Pentru controlul mișcării de înaltă precizie sau pentru funcțiile care necesită certificare, poate fi nevoie de o platformă de timp real separată. Pentru aplicațiile care trebuie să respecte cerințe de siguranță funcțională (de exemplu, conform IEC 61508 [53]), platforma trebuie evaluată nu doar din perspectiva predictibilității temporale, ci și a capabilităților de diagnosticare, izolare și tolerare a defectelor oferite de sistemul de operare și de mediul de execuție.

Nu în ultimul rând, arhitectura nucleului are un impact semnificativ asupra comportamentului sistemului. Un micronucleu, cum este cel al QNX [54], păstrează în nucleu doar funcțiile fundamentale și execută driverele și serviciile în procese izolate, limitând propagarea defectelor. Un nucleu monolitic poate avea o comunicare internă mai rapidă, dar un defect într-un driver privilegiat poate afecta o parte extinsă a sistemului [52].

Selecția finală ia în calcul WCET-ul, calitatea driverelor, compatibilitatea cu procesorul, capacitățile de comunicație, cerințele de siguranță și durata asistenței oferite de furnizor sau de comunitate. O platformă performantă în benchmark-uri poate fi nepotrivită dacă driverele introduc latențe necontrolate sau dacă sistemul nu poate fi actualizat și restaurat în siguranță [55].

2.5 Diagnosticarea controlului și repornirea secvențelor

Oricât de riguroasă ar fi proiectarea, defectele sunt inevitabile. Un PLC poate întrerupe o secvență din multiple cauze: senzor defect, cablu întrerupt, alimentare instabilă, modul I/O blocat, eroare de comunicație, componentă uzată, eroare software sau acțiune umană [56]. În ceea ce privește amenințările intenționate (atacuri cibernetice), metodologiile de modelare precum STRIDE, aplicate sistemelor cyber-fizice, facilitează identificarea sistematică a vectorilor de atac [57].

Simptomul vizibil nu indică întotdeauna cauza primară. Un senzor defect poate bloca o condiție permisivă, care blochează o secvență, ceea ce duce la o alarmă de „timp depășit” într-un alt modul. Operatorul vede alarma finală, dar evenimentul declanșator se află la mai multe niveluri de dependență [56].

Urmărirea acestei propagări este și mai dificilă când programul folosește condiții comune, logică duplicată, adresare indirectă sau interacțiuni între mai multe controlere. Prima alarmă înregistrată în jurnal nu este întotdeauna cauza originară, mai ales dacă marcasele temporale, ciclurile de scanare și întârzierile de comunicație nu sunt corelate. Un sistem cu adevărat diagnosticabil trebuie să ofere mult mai mult decât o listă de alarme. El trebuie să expună stările intrărilor brute și validate, cauzele interblocărilor, pasul curent al secvenței, valorile temporizatoarelor, calitatea legăturii de comunicație, codurile de diagnostic ale dispozitivelor și versiunea configurației. Să ne arate de ce o funcție nu poate porni, nu doar că timpul alocat a fost depășit [56], [58].

Diagnosticarea începe întotdeauna cu confirmarea stării fizice [56]. O valoare corectă pe ecran poate fi falsă dacă senzorul este defect sau o condiție intermediară maschează semnalul. Inspecția vizuală, măsurările electrice, istoricul alarmelor și graficele de evoluție oferă perspective complementare. Funcțiile de referință încrucișată și urmele de execuție reduc spațiul de căutare. Forțarea manuală a valorilor și simularea semnalelor pot, de asemenea, izola o componentă defectă, dar trebuie controlate prin proceduri stricte, deoarece pot ocoli protecțiile și genera acțiuni fizice nedorite. Metodele clasice funcționează bine când programul este modular și documentația corespunde realității; devin costisitoare când logica este duplicată sau mai multe controlere partajează aceeași memorie [56].

Analiza statică și dinamică a codului completează verificarea fizică. Secționarea programului (*slicing*) identifică instrucțiunile care influențează o variabilă; grafurile de dependență arată relațiile între condiții și ieșiri; urmele de execuție (*traces*) păstrează traseul către o anumită stare. Metodele bazate pe date pot prioritiza cauzele probabile, dar rezultatul trebuie interpretat prin structura programului și cunoașterea instalației [58]. Corelația nu implică cauzalitate, iar datele istorice pot omite tocmai stările rare care produc cele mai grave defecte. Tratarea sistematică a detectării anomaliilor face obiectul capitolului dedicat mentenanței predictive [58], [59]. Abordări complementare, precum diagnosticarea bazată pe modele (*model-based diagnosis*), utilizează modele formale ale comportamentului așteptat pentru a localiza defectele și pentru a reduce spațiul de căutare în sisteme complexe.

Un ultim aspect, adesea subestimat în practică, este faptul că repornirea aplicației software nu readuce instalația la starea normală. După reinițializare, memoria PLC-ului poate fi proaspătă, dar lumea fizică nu s-a resetat. Înainte de reluare, trebuie verificate intrările și ieșirile, poziția echipamentelor, comunicațiile și produsele aflate între operații. În funcție de situație, poate fi necesară oprirea controlată, referențierea axelor sau revenirea la un pas de secvență definit. Logica de repornire trebuie

să știe ce variabile se păstrează, ce ieșiri sunt forțate la valori prestabilite și când este obligatorie confirmarea operatorului. Principiile generale privind funcționarea degradată și recuperarea sunt dezvoltate în secțiunile 8.1–8.2 [36].

O structură PLC modulară, cu interfețe explicite, facilitează diagnosticarea. Fiecare modul ar trebui să expună starea curentă, condițiile de pornire, comanda activă, confirmarea finalizării și cauza blocării. Mesajul către operator trebuie să identifice echipamentul și condiția care împiedică execuția și să indice intervenția permisă. Proiectarea alarmelor și prioritizarea lor sunt tratate în secțiunea 8.5, iar securitatea configurației în 8.3.

Concluzia acestui capitol poate fi formulată astfel: proiectarea pentru diagnosticare și repornire se face înainte de punerea în funcțiune, nu după. Adăugarea ulterioară de alarme nu compensează o structură software opacă. Într-un sistem industrial, capacitatea de a explica de ce o funcție nu poate fi executată și de a restabili o stare coerentă este aproape la fel de importantă ca execuția corectă în regim nominal.

Partea a II-a. Conectivitate și infrastructuri de date

3. Edge, IIoT și comunicații industriale moderne

Capitolul de față tratează două probleme fundamentale ale oricărei arhitecturi industriale moderne: unde plasăm funcțiile de calcul și cum asigurăm schimbul de date între ele. În prima parte, sunt prezentate criteriile de repartizare a funcțiilor de-a lungul continuumului dispozitiv–edge–fog–cloud. În a doua parte, sunt analizate tehnologiile care fac posibil acest schimb: interoperabilitatea semantică, protocoalele IIoT, comunicațiile fără fir, rețelele deterministe și integrarea instalațiilor existente. Capitolul se încheie cu o discuție despre executarea aplicațiilor de inteligență artificială la edge. Alegerea arhitecturii și a tehnologiei de comunicație trebuie întemeiată pe cerințele procesului, pe proprietățile datelor și pe consecințele indisponibilității, nu doar pe capacitatea nominală a unei platforme [8], [11], [12].

Tehnologia de comunicație nu este universală. O rețea potrivită pentru telemetrie nu este automat potrivită pentru controlul mișcării, iar un protocol care garantează confirmarea livrării nu garantează și respectarea unui termen-limită. Din acest motiv, arhitecturile industriale combină protocoale deterministe pentru control, modele informaționale pentru interoperabilitate, mecanisme de publicare–abonare (*publish–subscribe*) pentru telemetrie și evenimente, precum și tehnologii fără fir adaptate mobilității ori dispersiei geografice. În toate cazurile, integrarea trebuie să păstreze autonomia funcțiilor locale și să limiteze explicit autoritatea nivelurilor superioare asupra procesului.

3.1 Alocarea funcțiilor în continuumul dispozitiv–edge–fog–cloud

Figura 3.1 sintetizează nivelurile arhitecturale și rolurile lor principale, împreună cu valori orientative pentru latență și volum de date.

La nivelul de câmp se găsesc senzorii, actuatorii, microcontrolerele, variatoarele de viteză, controlerele de mișcare și PLC-urile. Aceste componente interacționează direct cu procesul și execută funcțiile cu cele mai stricte cerințe temporale. Resursele lor de calcul pot fi limitate, însă apropierea de proces oferă trei avantaje decisive: acces direct la I/O, latență redusă și independență față de serviciile externe. Eșantionarea, condiționarea semnalului, reglarea, interblocările și protecțiile trebuie să rămână pe platforme proiectate și validate pentru aceste responsabilități. Funcțiile analitice adăugate în dispozitiv nu trebuie să afecteze predictibilitatea controlului de bază, iar consumul lor de resurse trebuie limitat sau izolat [2], [11].

Procesarea la edge plasează prelucrarea pe mașină, lângă senzor sau într-un echipament industrial de interconectare [12]. Nodul edge poate realiza conversia de protocoale, validarea măsurărilor, filtrarea, agregarea, contextualizarea, compresia, stocarea temporară și inferența locală. Reducerea distanței dintre măsurare și decizie permite reacția fără a aștepta un serviciu central și menține funcția pe perioada întreruperii conexiunii externe. Procesarea locală reduce și traficul: o cameră poate transmite rezultatul inspecției în locul fluxului video complet; un sistem de monitorizare a vibrațiilor poate transmite indicatori calculați local și poate păstra temporar formele de undă brute asociate evenimentelor [11], [12].

Agregarea nu trebuie să elimine informația necesară investigației, auditului sau reantrenării. Pentru semnalele de mare volum se poate folosi o memorie tampon circulară care păstrează datele brute, iar apariția unei condiții importante declanșează salvarea intervalului anterior și ulterior evenimentului. Trebuie găsit un compromis între traficul transmis, capacitatea locală de stocare, durata păstrării și

informația necesară reconstruirii unui incident. O arhitectură care păstrează doar scorul final al unui model poate deveni imposibil de verificat atunci când procesul produce un rezultat neașteptat.

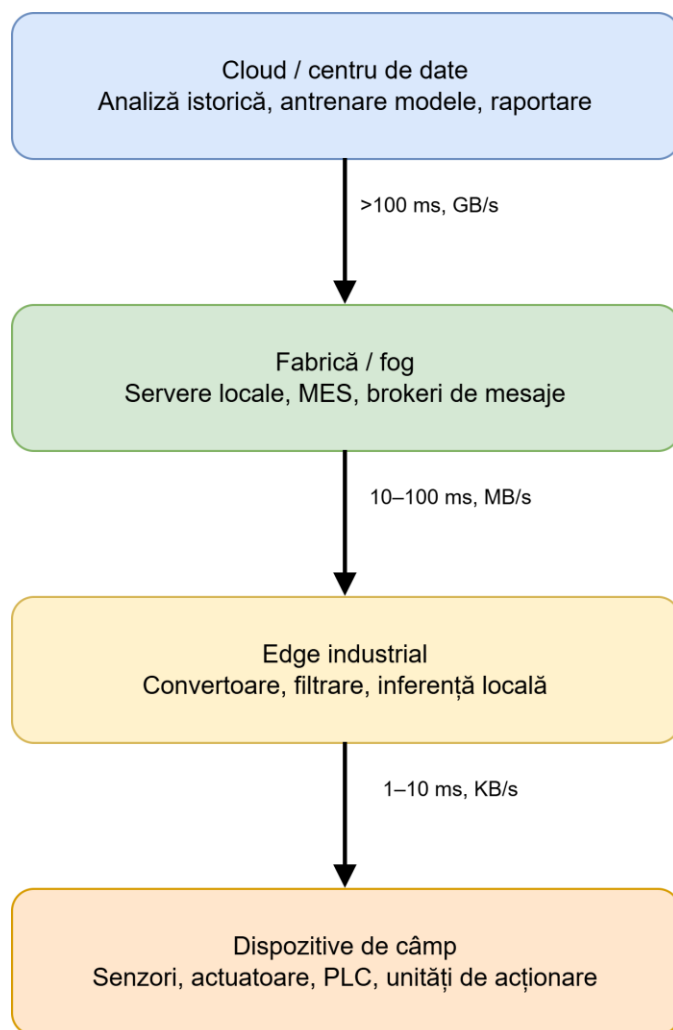


Figura 3.1. Arhitectura în straturi a unei platforme IIoT industriale

Nivelul fabricii, numit uneori nivel intermediar sau fog, coordonează mai multe mașini, linii sau noduri edge [60]. El poate include servere industriale, grupuri locale de calculatoare, intermediari de mesaje (*brokers*), sisteme de arhivare a datelor de proces, baze de date operaționale și servicii MES. Distincția dintre edge și nivelul intermediar nu ține doar de forma fizică: un echipament de interconectare care deservește o singură mașină are un rol edge; un server care distribuie aplicații și coordonează resurse la nivelul fabricii are un rol intermediar. Acest nivel oferă o infrastructură locală controlată și poate menține serviciile operaționale atunci când platforma cloud devine indisponibilă, fără a prelua controlul critic din PLC-uri [13], [61].

Cloud-ul și centrele de date furnizează resurse elastice pentru stocare, simulare, analiza flotelor de active, colaborare între locații și antrenarea modelelor pe volume mari de date [13]. Migrarea componentelor SCADA în infrastructuri cloud de tip IaaS (*Infrastructure as a Service*) a fost demonstrată practic, însă astfel de arhitecturi ridică provocări de securitate specifice [62], [63]. Utilizarea lor se decide în funcție de rezidența datelor, disponibilitatea conexiunii, cost, dependența de

furnizor și posibilitatea de migrare. Conform regulii autonomiei locale (secțiunea 2.1), pierderea serviciului central nu trebuie să ducă la pierderea controlului local. Conectivitatea cloud se justifică atunci când îmbunătățește coordonarea și exploatarea, nu când devine o condiție ascunsă pentru funcționarea sigură.

Tabelul 3.1. Amplasarea orientativă a funcțiilor industriale

Funcție	Amplasare obișnuită	Criteriu dominant
Interblocări și protecții	Controler local sau sistem de siguranță	Termen-limită strict (< 5 ms) și autonomie locală
Controlul mișcării	Unitate de acționare sau controler dedicat	Sincronizare sub-microsecundă și latență predictibilă
Validarea și filtrarea măsurărilor	Dispozitiv inteligent sau nod edge	Reducerea întârzierilor și a traficului de rețea
Inferență pentru viziune și monitorizare	Calculator industrial edge	Volum mare de date și termen de reacție scurt
Coordonarea mai multor celule	Server local al fabricii	Coordonare operațională și optimizare la nivel de secție
Arhivarea datelor de proces și servicii operaționale	Server al fabricii sau centru de date local	Disponibilitate ridicată și persistența datelor de proces
Antrenarea modelelor și analiza între locații	Centru de date sau cloud	Capacitate masivă de calcul și agregarea flotelor
Planificare și raportare la nivelul întreprinderii	Sisteme centrale (ERP/Cloud)	Scară temporală mare (zile/luni) și integrare economică

Amplasarea unei funcții trebuie să ia în calcul, simultan, scara temporală, volumul datelor, aria de coordonare, confidențialitatea, resursele platformei și comportamentul la defect [11]. Tabelul 3.1 prezintă o poziționare orientativă a principalelor funcții industriale, în funcție de criteriul dominant.

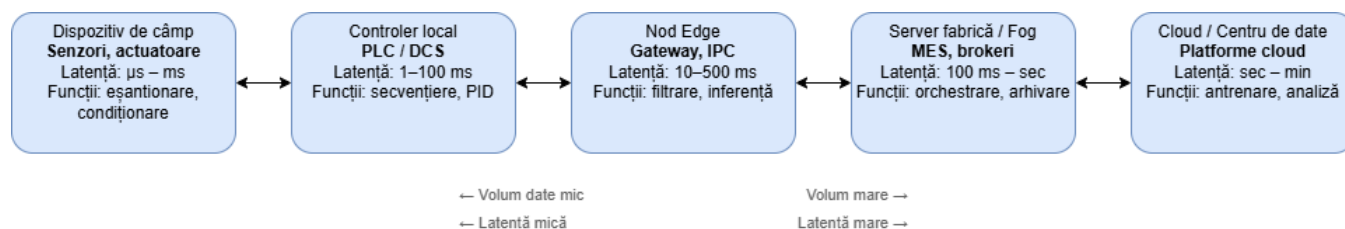


Figura 3.2 Amplasarea funcțiilor în continuul industrial: latență, volum de date și exemple

Decizia de amplasare a unei funcții software în continuul industrial este guvernată de un compromis fizic direct între latența de răspuns și volumul de date vehiculat (Figura 3.2). Cele două mărimi evoluează în sensuri divergente:

- La nivelul dispozitivului și al serverului edge local: se concentrează cerințele de timp real strict (latențe sub 1–10 ms), dar volumul de date stocat local este redus la contextul imediat al procesului;
- La nivelul platformei de fabrică și al infrastructurii cloud: latențele devin relaxate (sute de milisecunde până la minute), însă capacitatea de calcul masivă permite procesarea și corelarea unor volume mari de date istorice (*Big Data*).

O funcție critică de control în buclă închisă sau de oprire de urgență nu poate fi delegată către cloud doar pentru a profita de resurse de calcul mai ieftine, întrucât imprevizibilitatea rețelei exterioare anulează determinismul. În schimb, modelele de optimizare globală și analitica de flotă își găsesc firesc locul la nivelurile superioare.

Comportamentul în absența conexiunii trebuie stabilit înainte de implementare: care funcții continuă local, care decizii sunt suspendate, care este capacitatea memoriei tampon și cum se face reconcilierea după restabilirea comunicației. Mecanismul de stocare locală cu retransmitere ulterioară trebuie să trateze ordinea, duplicatele, expirarea și prioritățile, astfel încât alarmele importante să nu fie blocate de volumul de telemetrie obișnuită după reconectare.

Orchestrarea distribuie aplicații către noduri eterogene și stabilește resursele, numărul de instanțe și comportamentul la indisponibilitate. Optimizarea poate urmări latența, consumul energetic, traficul, costul și disponibilitatea, însă aceste obiective pot intra în conflict. Migrarea unui serviciu poate reduce timpul de calcul și, simultan, poate mări traseul de rețea; consolidarea aplicațiilor economisește energie, dar poate crea o cauză comună de indisponibilitate. În industrie, orchestrarea trebuie să țină seama de starea procesului: o relocare permisă în timpul unei opriri planificate poate fi interzisă în timpul unei operații critice. Funcțiile a căror validare depinde de un anumit procesor, accelerator, driver sau domeniu de siguranță trebuie menținute în mediul calificat [60], [61].

Pentru administrarea interoperabilă a aplicațiilor edge se pot folosi inițiative precum Margo, care urmărește reducerea dependenței dintre aplicațiile industriale și platformele proprietare de distribuire. Mecanismele de descriere și administrare a ciclului de viață sunt prezentate în secțiunea 11.4 [64].

3.2 IIoT, OPC UA și interoperabilitatea semantică

După clarificarea distribuției funcțiilor pe niveluri, aspectul esențial care se impune este comunicarea între aceste niveluri. Problema depășește simpla selecție a unui protocol. Într-un mediu IIoT, subsistemele provin adesea de la furnizori diferiți și implică protocoale, mecanisme de securitate și cicluri de viață diferite. Un controler poate rămâne în exploatare două decenii, în timp ce bibliotecile criptografice și serviciile cloud se actualizează mult mai frecvent. Arhitectura trebuie să permită modernizarea treptată, fără a impune înlocuirea simultană a întregii instalații [8].

Interoperabilitatea se realizează pe mai multe niveluri. La nivel sintactic, sistemele trebuie să interpreteze aceeași structură a mesajului. La nivel semantic, ele trebuie să atribuie aceeași semnificație valorilor, unităților, stărilor și relațiilor. La nivel operațional, informația trebuie legată de responsabilitățile și procesele în care este folosită. Un echipament de interconectare care transformă un registru Modbus într-un mesaj MQTT rezolvă doar conectivitatea și o parte a sintaxei. Pentru o integrare completă, trebuie păstrate explicit factorul de scalare, unitatea de măsură, marcajul temporal, indicatorul de calitate, identitatea activului și drepturile de scriere.

OPC Unified Architecture (OPC UA) este un cadru independent de platformă care reunește servicii de comunicație, modelarea informației, descoperirea serviciilor și mecanisme de securitate [30]. Spațiul de adrese reprezintă informația ca un graf de noduri tipizate, în care activele, variabilele, metodele,

tipurile de date și relațiile pot fi descoperite de aplicații. O măsurare poate fi însoțită de unitatea de măsură, domeniul de valori, calitatea, istoricul și echipamentul căruia îi aparține. Specificațiile însoțitoare (*Companion Specifications*) definesc structuri comune pentru mașini, robotică, procese, viziune artificială și alte domenii, reducând variațiile semantice dintre furnizori [30], [65].

Modelul Client/Server permite explorarea, citirea, scrierea, apelarea metodelor și accesul la alarme sau la istoric. OPC UA PubSub (*Publish–Subscribe*) permite distribuirea decuplată către mai mulți consumatori. OPC UA include autentificare, semnare, criptare, autorizare și audit, însă existența acestor funcții în standard nu garantează configurarea lor corectă. Certificatele, listele de încredere, rolurile și actualizările trebuie administrate pe întreaga durată de viață. OPC UA și MQTT nu sunt neapărat tehnologii concurente: OPC UA poate defini modelul informațional și serviciile industriale, în timp ce MQTT poate transporta telemetria și evenimentele printr-un intermediar de mesaje [66], [67]. Asocierea dintre ele trebuie să evite pierderea semanticii [62], [64]. Într-un domeniu conex, BACnet (*Building Automation and Control Network*) rămâne un protocol de referință pentru gestiunea sistemelor HVAC (*Heating, Ventilation and Air Conditioning*) și a iluminatului [68].

OPC UA Field eXchange (UAFX) extinde OPC UA către comunicațiile de nivel de câmp, oferind modele standardizate pentru componente de automatizare, entități funcționale, conexiuni și verificarea compatibilității. Specificațiile UAFX 1.00.03 (2025) definesc cadrul general și modelul informațional pentru *AutomationComponent* și *ConnectionManager*. Versiunea curentă acoperă, în primul rând, comunicațiile între controlere, celelalte tipuri de interacțiuni fiind prevăzute pentru dezvoltări ulterioare. UAFX urmărește interoperabilitatea independentă de furnizor pentru schimbul oportun de date, configurarea conexiunilor, diagnostic și integrarea cerințelor de securitate funcțională [35].

UAFX trebuie analizat împreună cu OPC UA PubSub și cu infrastructura de rețea. Modelul informațional definește componentele și conexiunile, PubSub susține schimbul de date, iar mecanisme precum TSN (*Time-Sensitive Networking*) pot asigura proprietăți temporale controlate. Utilizarea unei tehnologii comune nu elimină însă necesitatea configurării explicite: punctele terminale, fluxurile, perioadele, termenele-limită și politicile de acces trebuie validate pentru fiecare aplicație în parte.

3.3 Protocoale IIoT pentru telemetrie și administrarea dispozitivelor

Pe lângă OPC UA, care acoperă integrarea semantică, există o categorie de protocoale mai simple, destinate telemetriei, evenimentelor și administrării dispozitivelor cu resurse limitate. MQTT, CoAP și LwM2M răspund unor forme diferite de interacțiune și nu sunt alternative universale pentru orice funcție industrială.

MQTT utilizează modelul publicare–abonare (*publish–subscribe*). Clienții publică mesaje pe subiecte (*topics*) administrate de un intermediar de mesaje (*broker*), iar abonații primesc datele corespunzătoare filtrelor selectate. Subiectele trebuie proiectate ca interfețe stabile, iar conținutul util al mesajului trebuie să păstreze informațiile necesare interpretării și validării. MQTT definește trei niveluri de calitate a serviciului (*Quality of Service*, QoS): QoS 0 (fără confirmare), QoS 1 (cu confirmare, dar posibile duplicate) și QoS 2 (exact o singură livrare). Aceste niveluri nu garantează determinismul de timp real. Mesajele reținute (*retained messages*) și sesiunile persistente pot ajuta la reconectare, însă ultima valoare trebuie însoțită de marcaj temporal și de o perioadă de valabilitate. MQTT este potrivit pentru telemetrie și evenimente, dar nu oferă, prin el însuși, un model semantic sau control strict în timp real. Intermediarul de mesaje trebuie proiectat pentru disponibilitate, autentificare, autorizare și comportament controlat la suprasarcină [66], [69].

CoAP (*Constrained Application Protocol*) este un protocol de tip REST, destinat dispozitivelor cu resurse limitate și rețelelor cu pierderi, utilizat de regulă peste UDP. Dispozitivele expun resurse

identificate prin URI și folosesc operații similare conceptual cu GET, POST, PUT și DELETE. Mecanismul *Observe* permite urmărirea modificării unei resurse fără a repeta interogarea. OSCORE (*Object Security for Constrained RESTful Environments*) protejează mesajul la nivel de obiect și poate păstra protecția pe întregul traseu [70], [71]. Utilizarea CoAP în aplicații IoT industriale este detaliată în specificațiile de bază [70], [72] și în recomandările de securitate OSCORE [71].

LwM2M (*Lightweight Machine to Machine*) utilizează CoAP pentru a oferi un cadru complet de înregistrare, telemetrie, diagnostic, actualizare de firmware și administrare a ciclului de viață. Este relevant atunci când platforma trebuie să cunoască nu doar valorile transmise, ci și versiunea de firmware, starea operațională și credențialele dispozitivului [73].

Tabelul 3.2 compară caracteristicile esențiale ale acestor protocoale, evidențiind modelul de interacțiune și limitările fiecăruia.

Tabelul 3.2. Compararea protocoalelor IIoT

Caracteristică	MQTT	CoAP	LwM2M
Model principal	Publicare–abonare prin broker central	Cerere–răspuns și observare asincronă	Client–server structurat pe obiecte
Utilizare tipică	Telemetrie, mesagerie la eveniment, MES	Noduri constrânse, senzori IoT pe baterie	Înrolare automată, configurare, FOTA, diagnostic
Transport uzual	TCP, securizat prin TLS	UDP, securizat DTLS/OSCORE	UDP/TCP peste stiva CoAP
Intermediar	Broker de mesaje (punct unic de scalare)	Opțional (proxy / gateway intermediar)	Server LwM2M de administrare
Limitare importantă	QoS nu garantează termenele de timp real	Rețeaua UDP impune gestiunea pierderilor	Necesită modele de date standardizate (IPSO)
Potrivit pentru control determinist	Nu (lipsă determinism temporal)	Nu în bucle rapide	Nu (dedicat administrării dispozitivelor)

OPC UA	MQTT	CoAP
Model: Client/Server + PubSub	Model: Publicare/Abonare	Model: Cerere/Răspuns REST + Observe
Securitate: X.509, criptare, audit	Securitate: TLS, user/pass, ACL	Securitate: DTLS, OSCORE
QoS: Subscripții, Istoric, Alarme	QoS: 0 (max. o dată), 1, 2	QoS: Mesaje confirmabile/neconfirmabile
Transport: TCP, HTTPS, MQTT	Transport: TCP (WebSocket)	Transport: UDP

Figura 3.3 Comparăție orientativă a stivelor de comunicație: OPC UA, MQTT, CoAP

Fiecare protocol organizează diferit modelul de interacțiune, securitatea, calitatea serviciului și transportul (Figura 3.3). Aceste diferențe nu fac un protocol universal superior, ci impun o selecție ghidată de constrângerile aplicației și de integrarea cu sistemele existente. Selectarea ia în calcul dimensiunea și frecvența mesajelor, ordonarea, duplicatele, funcționarea fără conexiune și resursele dispozitivului. Pentru MQTT, drepturile de publicare și de abonare trebuie separate; pentru CoAP și LwM2M, operațiile permise asupra resurselor trebuie controlate. Indiferent de protocol, comenzile se validează local, în funcție de identitatea sursei, de marcajul temporal, de domeniul valorii și de starea procesului. Cerințele generale privind identitățile, cheile criptografice, actualizarea securizată și jurnalizarea sunt prezentate în secțiunea 8.3 [3], [4].

3.4 Rețele senzoriale fără fir, LPWAN și 5G cu capabilități reduse

Mobilitatea și dispersia geografică a unor active industriale impun utilizarea comunicațiilor fără fir. Proiectarea unui senzor fără fir trebuie să includă întregul buget energetic: măsurarea, condiționarea semnalului, procesarea, transmisia, recepția și autotestarea [74]. Durata de viață a bateriei se estimează pe baza ciclului real de funcționare, a retransmisiilor, a temperaturii și a îmbătrânirii. Reducerea frecvenței mesajelor prelungește autonomia, dar mărește vechimea informației și întârzie detectarea evenimentelor.

Pentru nodurile radio dispersate, ciclul de viață trebuie adaptat conectivității intermitente și accesului fizic dificil. Înregistrarea inițială asociază dispozitivul cu activul și cu amplasarea sa; actualizările de firmware trebuie să poată fi reluate după o întrerupere a comunicației și aplicate atomic. Monitorizarea trebuie să includă starea bateriei, calitatea legăturii radio și vechimea ultimei comunicații. Principiile generale ale ciclului de viață sunt prezentate în secțiunea 3.7, iar securitatea cibernetică în secțiunea 8.3 [3], [73].

LoRaWAN (*Long Range Wide Area Network*) utilizează o topologie de tip stea extinsă: dispozitivele transmit către echipamente de interconectare (*gateways*), iar serverul de rețea se ocupă de eliminarea duplicatelor și de coordonarea legăturii descendente. Clasa A minimizează consumul, clasa B adaugă ferestre programate, iar clasa C menține receptorul aproape continuu activ. LoRaWAN este potrivit pentru contorizare, monitorizarea mediului și senzori dispersați cu rată mică de transmisie, dar nu pentru controlul mișcării sau transmisii video [75]. Dimensionarea rețelei trebuie să includă bugetul radio, frecvența mesajelor, densitatea gateway-urilor și protecția cheilor criptografice [75], [76], [77].

NB-IoT (*Narrowband IoT*) este orientat către noduri cu trafic redus și acoperire extinsă, iar LTE-M (*LTE for Machines*) oferă o combinație mai bună între mobilitate, latență și volumul de date. Zigbee și Bluetooth Low Energy (BLE) se potrivesc pentru senzori locali și diagnostic mobil. Zigbee folosește o rețea de tip plasă (*mesh*) peste IEEE 802.15.4; BLE beneficiază de integrarea cu dispozitivele portabile. Niciuna dintre aceste tehnologii nu este un înlocuitor universal pentru rețelele industriale deterministe [78], [79], [80].

RedCap (*Reduced Capability*), introdus prin 3GPP Release 17, reduce complexitatea dispozitivelor 5G limitând lățimea de bandă și simplificând capabilitățile radio. Tehnologia țintește aplicațiile cu cerințe intermediare între senzorii minimali și terminalele 5G complete. eRedCap (*enhanced Reduced Capability*), introdus prin Release 18, continuă reducerea capabilităților pentru o categorie și mai simplă de dispozitive. RedCap și eRedCap nu înlocuiesc NB-IoT sau LTE-M, ci extind paleta de compromisuri între debit, latență, autonomie, cost și mobilitate [81].

Tabelul 3.3 oferă o poziționare orientativă a principalelor tehnologii de comunicație fără fir.

Tabelul 3.3. Poziționarea orientativă a tehnologiilor de comunicație fără fir

Tehnologie	Rază și infrastructură	Debit orientativ	Consum	Utilizare industrială caracteristică
BLE	Scurtă (<30 m), rețea locală	Redus–mediu (1–2 Mbps)	Foarte redus	Configurare inițială, senzori portabili, identificare
Zigbee	Scurtă–medie (<100 m), rețea tip plasă	Redus (250 kbps)	Redus	Rețele locale de senzori, monitorizare clădiri
LoRaWAN	Mare (1–10 km), rețea privată/publică	Foarte redus (<50 kbps)	Foarte redus	Utilități dispersate, contorizare, rezervoare izolate
NB-IoT	Rețea de operator, acoperire extinsă	Redus (<100 kbps)	Redus	Telemetrie periodică rară, active statice
LTE-M	Mare (rețea celulară telecom)	Mediu (1 Mbps)	Mediu–reduc	Urmărire active mobile, terminale logistice
5G RedCap / eRedCap	Mare (rețea celulară telecom)	Mediu (10–100 Mbps)	Optimizat	Dispozitive industriale 5G de complexitate medie
5G complet	Medie–mare (5G privat/public)	Mare (>1 Gbps)	Ridicat	Video 4K inspecție, flote AMR (<i>Autonomous Mobile Robot</i>), control distribuit

Valorile exacte depind de configurația radio, de spectru, de acoperire și de ciclul de activitate. Alegerea trebuie făcută pe baza unor măsurători în mediul real, nu doar a ratelor maxime declarate.

Miniaturizarea permite integrarea traductorului, a condiționării, a conversiei, a procesării și a comunicației într-un singur dispozitiv, fără a garanta însă calitatea măsurării [82], [83], [84]. Deriva termică, montajul, sensibilitățile încrucișate și protecția față de mediu trebuie evaluate cu aceeași rigoare ca pentru orice instrument de măsură industrial. Recuperarea energiei din vibrații, lumină sau diferențe de temperatură (*energy harvesting*) poate completa bateria, dar energia disponibilă este variabilă, iar dispozitivul trebuie să își adapteze eșantionarea fără a pierde evenimentele importante [84], [85], [86].

3.5 Determinismul comunicației, EtherCAT și TSN

Rata nominală de transfer nu descrie suficient comportamentul unei rețele industriale. Controlul depinde de latența întregului lanț, de variația întârzierii (*jitter*), de sincronizare, de pierderea mesajelor, de comportamentul la suprasarcină și de timpul de recuperare. Lanțul temporal include eșantionarea, procesarea în dispozitiv, accesul la mediu, traversarea rețelei, recepția, planificarea în aplicație și actualizarea actuatorului [2], [87].

Sistemele distribuite au nevoie de o bază temporală comună ori de câte ori măsurările trebuie corelate sau evenimentele ordonate. Sincronizarea se evaluează prin eroarea maximă și prin comportamentul la pierderea sursei de timp. Retransmiterea nu este întotdeauna soluția corectă: într-un flux periodic, un mesaj întârziat poate valora mai puțin decât următoarea actualizare. Aplicația trebuie să decidă dacă mesajul se retransmite, se abandonează sau se înlocuiește cu o valoare mai recentă.

EtherCAT este un exemplu matur de comunicație deterministă din categoria rețelelor de teren. Utilizează nivelul fizic Ethernet, însă dispozitivele procesează datele în timp ce cadrul traversează

rețeaua: fiecare nod subordonat citește informația care îi este adresată și introduce datele proprii, fără a stoca și retransmite cadrul integral. Determinismul provine din controlul centralizat al ciclului, din procesarea realizată de componente hardware specializate (*slave controllers*), din topologia cunoscută și din sincronizarea prin *Distributed Clocks* [88].

Working Counter este actualizat de fiecare nod și permite controlerului principal să detecteze diferența dintre configurația așteptată și transferul realizat. *Distributed Clocks* sincronizează ceasurile componentelor și compensează întârzierile de propagare. Safety over EtherCAT (FSoE) aplică principiul „canalului negru” (*black channel*): protocolul de siguranță detectează coruperea, repetarea, pierderea și erorile de secvență fără a presupune că rețeaua standard este sigură [88], [89].

Time-Sensitive Networking (TSN) reprezintă o familie de mecanisme IEEE 802.1 pentru sincronizare, planificarea transmisiei, modelarea traficului, limitarea latenței și reziliență în rețele Ethernet comutate [87], [90]. TSN nu este un protocol unic și nu creează automat determinism. IEEE 802.1AS furnizează baza temporală comună [91], iar *Time-Aware Shaper* (IEEE 802.1Qbv) deschide și închide porțile de transmisie după un program repetitiv, rezervând ferestre pentru traficul critic [92]. Preempțiunea cadrelor (*frame preemption*) reduce blocajul produs de cadrele lungi cu prioritate inferioară. IEEE 802.1CB permite replicarea cadrelor pe trasee distincte și eliminarea duplicatelor la destinație [91], [92], [93], [94].

EtherCAT și TSN nu trebuie privite ca înlocuitori direcți. EtherCAT este optimizat pentru comunicația ciclică la nivel de teren; TSN urmărește convergența mai multor clase de trafic într-o infrastructură Ethernet comutată. O instalație poate folosi EtherCAT pentru sincronizarea axelor și TSN pentru conectarea celulelor și a serviciilor superioare. OPC UA PubSub și UA FX definesc schimbul și semnificația informației, iar TSN poate oferi transportul cu proprietăți temporale controlate.

Peisajul comunicațiilor deterministe la nivel de fabrică este completat de alte protocoale Ethernet industrial consacrate:

- **PROFINET:** Distinge trei clase de performanță: PROFINET NRT (*non-real-time*, bazat pe TCP/IP standard), PROFINET RT (*Real-Time*, care ocolește stiva de transport pentru a atinge latențe de ordinul a 1–10 ms) și PROFINET IRT (*Isochronous Real-Time*). IRT utilizează comutatoare Ethernet cu suport hardware dedicat și o schemă strictă de divizare în timp (*time-slice*), garantând cicluri sub 1 ms și jitter sub 1 μs, fiind optimizat pentru controlul coordonat al mișcării (*motion control*) [95].
- **EtherNet/IP:** Utilizează protocolul comun de aplicare CIP (*Common Industrial Protocol*) peste stiva standard TCP/IP și UDP/IP. Determinismul și sincronizarea axelor sunt obținute prin extensia *CIP Sync*, care implementează profilul de sincronizare temporală de precizie IEEE 1588 / PTP (*Precision Time Protocol*) fără a necesita echipamente de rețea modificate proprietar [95].
- **IO-Link (IEC 61131-9):** Reprezintă standardul de comunicație punct-la-punct (pe 3 fire, neecranat) pentru nivelul cel mai de jos al ierarhiei (senzori și actuatori inteligenți). Deși nu este o magistrală de câmp propriu-zisă (*fieldbus*), IO-Link elimină conversiile analogice intermediare, transmițând digital datele de proces, metadatele de diagnosticare și parametrii de configurare către un *master IO-Link*, care le mapează mai departe transparent în rețeaua PROFINET, EtherCAT sau EtherNet/IP a fabricii [96].

Tabelul 3.4 sintetizează și compară principalele tehnologii de comunicație deterministă și de câmp, evidențiind standardele de referință, mecanismele de obținere a determinismului, timpii de ciclu caracteristici și nivelul arhitectural de aplicare [95], [96], [97].

Tabelul 3.4. Comparație a tehnologiilor de comunicație deterministă și de câmp

Tehnologie	Standard de referință	Mecanism de determinism	Timp de ciclu tipic	Nivel architectural / Aplicație
EtherCAT	IEC 61158 / IEC 61784	Procesare cadru on-the-fly + Distributed Clocks	50 μ s–1 ms	Câmp / I/O rapide, servo-acționări multi-ax
PROFINET IRT	IEC 61158 / IEC 61784	Divizare temporală hardware (<i>time-slice</i>) dedicată	250 μ s–1 ms	Nivel mașină / Robotică, linii rapide de asamblare
EtherNet/IP (CIP Sync)	IEC 61158 / ODVA CIP	Sincronizare distribuită IEEE 1588 (PTP) peste UDP	1 ms–10 ms	Nivel celulă și proces / Control logic și acționări
TSN (IEEE 802.1)	Standarde IEEE 802.1 (Qbv, AS, CB)	Time-Aware Shaper + preempțiune cadre L2	<1 ms (convergent)	Backbone convergent unificat IT/OT
IO-Link	IEC 61131-9	Comunicație master-slave serială punct-la-punct (UART)	2 ms–20 ms	Nivel senzor/actuator, diagnosticare

3.6 Rețele private 5G și comunicații industriale mobile

5G poate susține aplicații cu lățime de bandă mare, densitate ridicată de dispozitive, mobilitate și cerințe reduse de latență. Categoriile eMBB (*enhanced Mobile Broadband*), URLLC (*Ultra-Reliable Low-Latency Communication*) și mMTC (*massive Machine-Type Communication*) descriu familii de capabilități, însă implementarea concretă depinde de terminale, spectru, topologia rețelei și configurația serviciilor. O rețea privată 5G oferă control asupra acoperirii, identităților, politicilor de acces și integrării cu procesarea la edge. Aplicațiile tipice includ AMR-uri, camere, interfețe HMI mobile și urmărirea activelor [81], [98]

Segmentarea logică a rețelei (*network slicing*) permite definirea unor servicii cu politici distincte, însă segmentele logice pot partaja componente fizice și nu garantează automat independența. Procesarea multi-acces la edge (*Multi-access Edge Computing*, MEC) scurtează drumul până la aplicație, dar latența totală continuă să includă procesarea terminalului, planificarea radio și transportul.

5G nu justifică mutarea tuturor buclelor de control pe comunicații fără fir. Transferul unei funcții este acceptabil numai după ce latența, disponibilitatea și comportamentul la pierderea legăturii au fost demonstrate în mediul real al instalației. RedCap și eRedCap extind utilizarea 5G către terminale mai simple și mai economice, dar nu schimbă principiul fundamental: funcțiile critice impun validarea întregului lanț de comunicație și control.

3.7 Echipamente de interconectare, instalații existente și administrarea ciclului de viață

În realitatea industrială, tehnologiile noi nu le înlocuiesc automat pe cele vechi. Instalațiile existente combină frecvent RS-485, Modbus, PROFIBUS, CAN, Ethernet industrial, OPC UA și MQTT. Protocoalele vechi nu dispar odată cu instalarea unei platforme IIoT, deoarece înlocuirea echipamentelor poate fi costisitoare sau imposibilă fără oprirea producției.

Echipamentul industrial de interconectare leagă aceste domenii și poate realiza conversia de protocoale, contextualizarea, stocarea temporară, filtrarea și aplicarea politicilor de acces. Conversia

trebuie să trateze tipurile de date, ordinea octeților, factorii de scalare, unitățile, codurile de stare și drepturile de scriere. Transferul corect al unui număr nu garantează că valoarea este actuală, plauzibilă fizic sau interpretată în aceeași unitate de măsură.

Modbus rămâne răspândit datorită simplității modelului său de date, bazat pe bobine (*coils*), intrări discrete, registre de intrare și registre de păstrare. Aceste categorii nu definesc unitatea, scalarea sau ordinea octeților. Protocoalele clasice nu oferă nativ modele semantice sau autentificare puternică; de aceea, dispozitivele care le folosesc trebuie segmentate și accesate prin echipamente de interconectare care limitează scrierile și marchează valorile neactualizate [99], [100], [101].

Echipamentul de interconectare constituie o frontieră între dispozitivele vechi și infrastructura modernă. El trebuie să limiteze scrierile, să marcheze datele neactualizate și să valideze local prospețimea și compatibilitatea comenzilor cu starea procesului. Cerințele generale privind segmentarea, identitățile și jurnalizarea sunt prezentate în secțiunea 8.3 [3], [4].

Nodurile edge, echipamentele de interconectare și dispozitivele IIoT trebuie administrate ca active industriale. Ciclul lor de viață include înregistrarea, configurarea, monitorizarea, actualizarea, revenirea la o versiune anterioară și dezafectarea. Actualizările se distribuie gradual, iar configurația nouă se verifică atât la nivel software, cât și în raport cu comunicațiile și procesul deservit. Monitorizarea operațională trebuie să coreleze starea componentelor fizice, utilizarea resurselor, conectivitatea și calitatea datelor. Administrarea aplicațiilor pe flote eterogene este dezvoltată în secțiunea 11.4.

3.8 Inteligență artificială la edge: optimizare, distribuire și exploatare

Pe infrastructura descrisă în secțiunile anterioare, inteligența artificială la edge permite execuția inferenței și, în anumite situații, adaptarea controlată a unui model în apropierea sursei de date. Aplicațiile includ inspecția vizuală, detectarea anomaliilor, estimarea calității și validarea senzorilor. Procesarea locală poate reduce latența și transferul de date sensibile, însă modelul trebuie evaluat pe platforma finală, sub încărcări reprezentative și în condițiile termice reale de exploatare. Acuratețea obținută pe o stație de dezvoltare nu dovedește respectarea termenului-limită pe un calculator industrial [25], [102].

Modelele de inteligență artificială dezvoltate pe stații puternice de calcul impun o optimizare riguroasă înainte de a fi distribuite pe dispozitive industriale edge cu resurse de memorie și calcul limitate (TinyML) [103]. Această optimizare se bazează pe trei mecanisme principale:

- Cuantizarea (*Quantization*): reduce precizia reprezentării numerice a parametrilor (de exemplu, trecerea de la virgulă mobilă FP32 la întregi INT8), scăzând amprenta de memorie de până la 4 ori și accelerând inferența pe procesoare industriale fără nuclee grafice dedicate;
- Tăierea structurală (*Pruning*): identifică și elimină conexiunile, neuronii sau canalele de convoluție a căror contribuție la rezultatul final este neglijabilă;
- Distilarea cunoașterii (*Knowledge Distillation*): transferă acuratețea și capacitatea de generalizare a unui model masiv („profesor”) către o rețea mult mai compactă („student”).

Pentru accelerarea hardware a inferenței pot fi utilizate GPU-uri industriale, unități de procesare neuronală (NPU) sau FPGA-uri. Selecția platformei nu se face însă exclusiv pe baza numărului mediu de cadre sau inferențe pe secundă: sunt critice timpul maxim de execuție (*Worst-Case Execution Time*), consumul energetic și stabilitatea termică în carcasă închisă. Pe microcontrolere și senzori inteligenți (TinyML), un model compact devine justificat doar atunci când variabilitatea semnalului

depășește posibilitățile programării deterministe; pentru relații simple și stabile, o regulă clasică rămâne mai sigură, mai transparentă și mai ușor de verificat [103].

Într-o arhitectură consacrată, antrenarea se realizează central, iar inferența rulează pe nodul edge. Pachetul distribuit trebuie să includă modelul, preprocesarea, pragurile, bibliotecile și cerințele privind procesorul sau acceleratorul. Compatibilitatea se verifică pe platforma finală, iar introducerea graduală permite compararea performanțelor înainte de extindere. Principiile generale de administrare a versiunilor și monitorizare a derivei sunt prezentate în secțiunea 5.4.

Învățarea federată (*federated learning*) permite mai multor noduri să contribuie la un model comun, fără a transfera sistematic toate datele brute [104]. Fiecare participant calculează local actualizări, pe care un coordonator le agregă. Metoda poate reduce centralizarea datelor, însă nu garantează automat confidențialitatea sau echitatea performanței între locații. Trebuie evaluate riscul compromiterii modelului prin actualizări ostile și degradarea locală mascată de o îmbunătățire medie globală. Modelul agregat se validează separat pentru fiecare locație, produs și regim relevant [105].

În exploatare, nodul edge trebuie monitorizat simultan ca platformă de calcul și ca parte a procesului. Se urmăresc timpul maxim de inferență, utilizarea resurselor, temperatura, starea aplicației și disponibilitatea mecanismului de rezervă. Supraîncărcarea sau degradarea termică pot invalida funcția chiar dacă modelul nu s-a modificat. Un model executat la edge nu preia implicit o funcție de siguranță; rezultatele sale trebuie încadrate prin validări și protecții locale. Validarea generală a modelelor este tratată în secțiunea 5.4, iar autoritatea asupra procesului în secțiunea 8.5. Distribuirea și actualizarea acestor modele pe flote de dispozitive edge beneficiază de pe urma inițiativelor de orchestrare standardizată, cum este Margo, care separă descrierea aplicației de platforma de execuție și permit administrarea unitară a ciclului de viață.

Edge, IIoT și comunicațiile industriale trebuie evaluate ca un singur sistem complet. Un accelerator rapid nu compensează o rețea imprevizibilă, un protocol semantic nu suplinește un senzor necalibrat, iar o conexiune criptată nu dovedește că mesajul este actual sau plauzibil fizic. Arhitectura trebuie să lege măsurarea, comunicația, calculul și acțiunea prin cerințe verificabile referitoare la timp, la calitatea informației, la disponibilitate și la autoritate.

Partea a III-a. Inteligență și control

4. Date industriale și procesarea semnalelor

Datele industriale cuprind măsurări, stări discrete, evenimente, alarme, imagini, înregistrări audio și informații operaționale despre produse, procese și resurse. Valoarea lor nu stă în volum sau în viteza de generare, ci în capacitatea de a susține, în condiții bine definite, controlul, supravegherea, analiza și luarea deciziilor [106], [107].

Capitolul este structurat în două secvențe complementare. Prima urmărește lanțul de date industriale – de la context, transport și procesare până la gestionarea datelor, identificarea sursei acestora și modul în care acestea se modifică în timp. A doua urmărește lanțul semnalului – de la achiziție și preprocesare până la filtrare, reprezentări timp-frecvență și extragerea caracteristicilor.

4.1 Contextul și ciclul de viață al datelor industriale

Datele industriale sunt adesea descrise prin volum, viteză, varietate, veridicitate și valoare [106]. Aceste caracteristici devin însă utile numai raportate la procesul fizic și la decizia pe care o sprijină. Volumul poate proveni din semnale eșantionate la frecvență înaltă, camere industriale, variabile din sistemul de arhivare sau perioade lungi de păstrare. Viteza trebuie analizată distinct la nivelul eșantionării, transmisiei, stocării și utilizării: un senzor poate achiziționa vibrații la zeci de kilohertzi și poate transmite rezultatul o dată pe secundă, iar termenul-limită al deciziei poate fi diferit de toate aceste frecvențe. Varietatea provine din coexistența valorilor scalare, a seriilor temporale, a evenimentelor, a formelor de undă și a imaginilor. Veridicitatea este afectată de zgomot, eroare sistematică, derivă, valori lipsă, duplicate și erori de sincronizare. Valoarea apare atunci când informația susține o acțiune: reglarea procesului, reducerea consumului, detectarea degradării sau asigurarea trasabilității [106], [107], [108].

Fiecare observație trebuie însoțită de un context operațional suficient pentru interpretare: identitatea echipamentului și a senzorului, momentul măsurării, unitatea inginerescă, calitatea valorii, informațiile de calibrare, versiunea configurației, modul de funcționare și, după caz, produsul, lotul sau rețeta. Marcajul temporal trebuie să reflecte, pe cât posibil, momentul observației fizice, nu doar momentul recepției sau al scrierii într-o bază de date. Indicatorul de calitate trebuie să deosebească o valoare măsurată validă de una incertă, estimată, substituită sau neactualizată. Fără acest context, două valori numerice identice pot avea semnificații complet diferite, iar un model poate interpreta variațiile legitime între regimuri drept degradări ale procesului. Ciclul de viață al datelor începe cu achiziția, unde o mărime fizică este convertită într-o reprezentare digitală; aici trebuie păstrate identitatea sursei, momentul observației, unitatea de măsură, starea senzorului și informațiile necesare interpretării calității, aspecte ale căror detalii tehnice sunt dezvoltate în secțiunea 4.4.

Transportul poate folosi magistrale de teren, Ethernet industrial, OPC UA, MQTT, API-uri (*Application Programming Interface*) și platforme de procesare a fluxurilor. El trebuie să păstreze identitatea sursei, marcajul temporal, starea de calitate și ordinea relevantă a evenimentelor. Simplul succes al transmisiei nu dovedește că valoarea este actuală sau corectă fizic: un echipament de interconectare poate retransmite ultima valoare cunoscută după pierderea dispozitivului, iar aplicația o poate interpreta drept măsurare curentă dacă vechimea nu este marcată. Numerele de secvență sau identificatorii de eveniment permit detectarea pierderilor, duplicatelor și reordonărilor. Comportamentul la întreruperea comunicației trebuie definit prin capacitatea memoriei tampon,

perioada maximă de păstrare, politica de retransmitere și tratarea datelor sosite după expirarea intervalului operațional.

Validarea verifică schema, tipul, domeniul fizic, marcajul temporal, numărul de secvență, duplicatele și coerența cu alte variabile. Regulile nu pot fi pur numerice: un debit negativ poate fi legitim într-o instalație bidirecțională și imposibil în alta; o temperatură constantă poate indica fie un proces stabil, fie un senzor blocat. Eliminarea automată a tuturor valorilor extreme poate ascunde tocmai evenimentele pe care sistemul trebuie să le detecteze. În multe aplicații, valoarea este păstrată împreună cu un indicator de calitate, iar decizia este amânată până la analiza contextului.

Contextualizarea transformă o variabilă de proces izolată în informație operațională, asociind-o cu echipamentul, produsul, lotul, operația și starea procesului. Sistemul de arhivare, MES-ul și aplicația de mentenanță oferă perspective complementare care trebuie corelate prin identificatori stabili. Denumirile locale ale variabilelor se pot modifica, însă identitatea activului și relațiile dintre componente trebuie menținute prin modele administrate.

Stocarea se alege în funcție de tipul datelor și de cerințele de audit. Sistemul de arhivare a procesului este optimizat pentru serii temporale; stocarea de obiecte se potrivește pentru imagini și forme de undă; lacul de date păstrează date brute pentru explorare; depozitul de date conține informații curate, destinate raportării. Aceste sisteme pot coexista, dar flexibilitatea unui lac de date presupune catalogare, atribuirea responsabilității și politici de calitate. Perioada de păstrare se stabilește în funcție de scop: telemetria agregată poate fi păstrată ani de zile, în timp ce semnalele brute de înaltă frecvență sunt reținute doar în jurul evenimentelor relevante.

Prelucrarea locală poate reduce traficul și latența; platformele centrale pot susține analiza istorică, comparația între active și antrenarea modelelor. Principiile de repartizare între edge și cloud au fost prezentate în secțiunea 3.1. În lanțul de date, trebuie stabilite informațiile păstrate pentru verificare, legătura dintre rezultatul calculat și datele-sursă, durata de păstrare și condițiile în care datele brute sunt transferate sau eliminate.

4.2 Arhitecturi de procesare pe loturi, continuă și orientată pe evenimente

Procesul de extragere, transformare și încărcare (ETL-*Extract, Transform, Load*) presupune prelucrarea datelor înainte încărcării în sistemul țintă, fiind potrivit când schema este stabilă și informația trebuie validată înainte de utilizare [109]. Procesul de extragere, încărcare și transformare (ELT-*Extract, Load, Transform*) păstrează datele brute și aplică transformările ulterior, fiind indicat pentru volume mari și explorare. Industria folosește frecvent o arhitectură hibridă: validările critice se execută înainte încărcării, o copie brută este păstrată, iar transformările suplimentare au loc în platforma analitică [109], [110].

De exemplu, datele unui sistem de monitorizare a motoarelor pot fi validate local din punct de vedere al domeniului fizic și al marcajului temporal, apoi încărcate într-un sistem de arhivare. Indicatorii calculați pot fi folosiți imediat pentru supraveghere, în timp ce datele brute sunt procesate ulterior. ETL și ELT sunt decizii legate de momentul și locul transformării: validările care protejează procesul trebuie aplicate devreme; transformările exploratorii pot fi realizate mai târziu.

Procesarea pe loturi grupează datele pe intervale și este potrivită pentru raportare, reconciliere și antrenarea modelelor. Procesarea continuă tratează neîntrerupt evenimente sau ferestre mici de date și poate susține monitorizarea prospețimii și generarea de alarme [111]. Aceeași arhitectură poate produce un rezultat provizoriu și îl poate corecta ulterior, după sosirea unor date întârziate. Procesarea continuă nu trebuie confundată cu funcționarea strictă în timp real: prelucrarea permanentă a fluxului nu garantează o limită deterministă a latenței [111]. În acest context, într-o conductă de procesare

continuă trebuie deosebite momentul producerii evenimentului, momentul intrării în platformă și momentul procesării; când se urmărește succesiunea fizică, calculele se raportează la momentul producerii evenimentului. Discuția despre semnificația marcajelor temporale și vechimea informației a fost dezvoltată în secțiunea 2.2 [2], [111].

Ferestrele temporare grupează evenimentele pentru calculul indicatorilor. Ferestrele consecutive împart fluxul în intervale distincte; ferestrele glisante se suprapun; ferestrele de sesiune sunt delimitate de perioade de activitate. Alegerea ferestrei trebuie să urmeze fenomenul fizic: o medie pe cinci minute poate fi potrivită pentru temperatură, dar poate ascunde un șoc de vibrație. Într-un proces pe loturi, o delimitare prin starea procesului este mai potrivită decât o fereastră cu durată fixă. Lungimea, suprapunerea și condițiile de închidere fac parte din definiția indicatorului și se administrează prin versiuni.

Datele întârziate impun o politică explicită. Un raport poate fi recalculat, un KPI (*Key Performance Indicator*) poate fi marcat drept provizoriu, iar o înregistrare întârziată poate fi dirijată către un flux separat. Pentru o alarmă sincronizată cu procesul, informația sosită după termenul-limită poate să nu mai aibă valoare operațională. Rezultatul disponibil la momentul deciziei trebuie diferențiat de cel corectat ulterior.

Sistemele orientate pe evenimente trebuie proiectate pentru apariția duplicatelor, livrarea neordonată și repetarea procesării. O operație idempotentă poate fi executată de mai multe ori fără a-și multiplica efectul – proprietate critică pentru comenzile cu efect fizic sau tranzacțional. Identificatorii de eveniment și numerele de secvență ajută la eliminarea duplicatelor, însă procesarea efectivă „o singură dată” trebuie privită ca o proprietate a întregului lanț aplicativ.

Când rata de intrare depășește capacitatea de procesare, sistemul poate încetini sursa, poate extinde temporar memoria tampon, poate elimina date sau poate acorda prioritate anumitor clase. Stocarea temporară nelimitată prefăce suprasarcina în latență. O alarmă asociată unui defect trebuie să aibă prioritate față de telemetria obișnuită. Pragurile cozielor, politica de eliminare și comportamentul la saturare se definesc înainte de intrarea în exploatare.

4.3 Calitatea, proveniența, guvernanta și evoluția schemelor

Guvernanta datelor definește responsabilitățile legate de sensul, calitatea, accesul, durata de păstrare și modificarea informației. Responsabilitatea tehnică și cea operațională pot reveni unor echipe diferite: IT-ul administrează baza de date, ingineria de proces stabilește semnificația măsurării, iar departamentul de calitate deține regulile de acceptare. Pentru fiecare set important de date, trebuie precizați proprietarul operațional, consumatorii autorizați, criteriile de calitate și perioada de păstrare. Lipsa acestor responsabilități duce la variabile fără documentație și indicatori calculați cu definiții incompatibile [107], [108].

Trasabilitatea datelor descrie modul în care informația a fost achiziționată, filtrată, agregată și corectată. Un rezultat trebuie legat de datele-sursă, de intervalul analizat, de versiunile schemelor și de parametrii de preprocesare. Un indicator calculat dintr-o serie reeșantionată nu este echivalent cu același indicator calculat direct din date brute. Dacă preprocesarea se modifică, trebuie modificată și versiunea rezultatului. Trasabilitatea este esențială și pentru modelele de învățare automată, care trebuie legate de surse, perioade, transformări și etichete.

Fluxul de date trebuie monitorizat ca orice sistem operațional. Observabilitatea sa include ratele de intrare și procesare, adâncimea cozielor, latența, prospețimea, încălcările de schemă și starea surselor. Aceste informații permit să se facă distincția între oprirea unui senzor, indisponibilitatea unui

echipament de interconectare sau o modificare neanunțată a schemei. Fără această diferențiere, un model poate continua să producă rezultate aparent valide, deși nu mai primește date actualizate.

Datele industriale pot conține rețete, parametri de proces și proprietate intelectuală; de aceea, accesul și configurațiile conductei de date trebuie protejate. Copia de siguranță trebuie să cuprindă nu doar valorile, ci și metadatele, schemele, cataloagele și regulile de calitate. Restaurarea unui volum de date fără catalog poate lăsa un sistem disponibil tehnic, însă inutil operațional [3].

Formatul definește reprezentarea datelor; schema definește câmpurile, tipurile și relațiile dintre ele. Formatele text precum CSV, JSON și XML sunt ușor de inspectat, dar pot introduce ambiguități. Formatele binare – CBOR (*Concise Binary Object Representation*), Protocol Buffers [112], Avro – reduc volumul și timpul de parsare, însă impun o administrare riguroasă a versiunilor. CBOR oferă o reprezentare binară compactă inspirată din modelul JSON [113].

Schema sintactică nu este suficientă pentru interpretare. Un contract de date completează schema cu unități, domenii valide, calitate, frecvență așteptată, latență și responsabilitate. El trebuie să precizeze dacă valorile sunt măsurate, interpolate, estimate sau generate de un model. Contractul fixează și responsabilitățile: producătorul respectă schema și frecvența declarată, iar consumatorul gestionează explicit absența, întârzierea și modificările compatibile.

Evoluția schemelor este inevitabilă pe măsură ce firmware-ul și aplicațiile se modifică. Compatibilitatea cu versiunile anterioare permite noilor consumatori să citească date vechi; compatibilitatea cu versiunile următoare permite consumatorilor existenți să accepte date produse de schema nouă. Apache Avro oferă mecanisme de rezolvare a schemelor pentru a citi înregistrări produse cu versiuni diferite [114]. O modificare aparent minoră – de exemplu, trecerea de la milimetri la metri fără versionare – poate produce rezultate plauzibile numeric, dar greșite fizic. Datele brute, validate, suspecte și agregate se păstrează separat sau se diferențiază prin informații de trasabilitate, astfel încât sursa și transformările să poată fi reconstituite.

După ce datele au fost structurate, catalogate și securizate la nivelul platformelor software (așa cum s-a detaliat în secțiunile 4.1–4.3), devine esențial să analizăm cum ia naștere această informație la sursă, prin conversia fenomenelor fizice în date digitale.

4.4 Achiziția și preprocesarea semnalelor industriale

În context industrial, semnalul reprezintă expresia electrică a unei mărimi fizice dinamice asociate procesului: vibrația lagărelor, curentul absorbit de motoare, presiunea hidraulică, debitul sau temperatura. Lanțul de achiziție cuprinde traductorul, condiționarea analogică, filtrarea antialiasing, eșantionarea, cuantizarea, marcarea temporală și validarea. Fiecare etapă poate modifica informația disponibilă, iar erorile introduse înainte de conversie nu pot fi întotdeauna corectate prin procesare ulterioară [115].

Traductorul convertește o mărime fizică într-un semnal electric și introduce sensibilitate, domeniu de măsurare, liniaritate și zgomot propriu. Condiționarea analogică adaptează nivelul semnalului, îndepărtează componenta continuă nedorită și limitează banda. Filtrul antialiasing, plasat înaintea eșantionării, elimină componentele de frecvență care depășesc jumătatea frecvenței de eșantionare, prevenind replierea spectrală.

Figura 4.1 ilustrează lanțul complet de achiziție și preprocesare, de la traductor până la extragerea caracteristicilor.

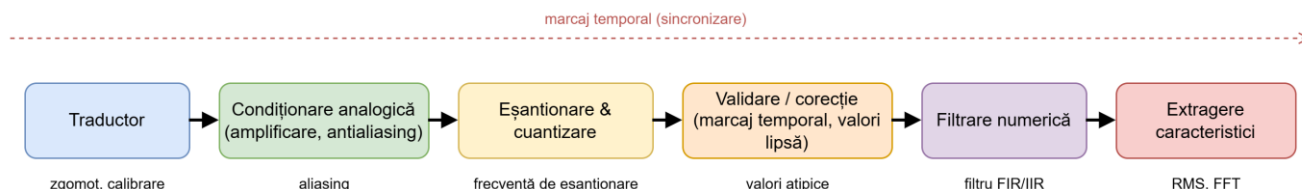


Figura 4.1. Lanțul de achiziție și preprocesare a semnalului industrial

Frecvența de eșantionare se alege în funcție de dinamica de interes. O eșantionare prea lentă produce repliere spectrală (*aliasing*): componentele de frecvență înaltă apar eronat la frecvențe mai joase. Teorema Nyquist oferă o limită teoretică, dar proiectarea are nevoie de o marjă suplimentară pentru filtrul antialiasing. În practică, frecvența de eșantionare se alege de 2.5 până la 5 ori mai mare decât frecvența maximă de interes, pentru a oferi suficientă atenuare în banda de tranziție a filtrului. Eșantionarea excesivă mărește volumul fără a aduce informație utilă, însă poate fi avantajoasă atunci când se urmărește o rezoluție temporală foarte fină sau când sunt necesare calcule de corelație precisă.

Cuantizarea convertește amplitudinea eșantionată într-o valoare numerică discretă. Rezoluția convertorului analog-numeric (de obicei 12, 16 sau 24 de biți) determină eroarea de cuantizare. Un convertor pe 16 biți oferă o gamă dinamică teoretică de aproximativ 96 dB, suficientă pentru majoritatea aplicațiilor industriale. La semnalele multicanal, sincronizarea este necesară pentru analiza de fază, corelație și fuziunea senzorilor. Un marcaj temporal pus la recepție poate fi insuficient dacă semnalul a fost eșantionat mai devreme, conform cerințelor de sincronizare din IEEE 1588 [93]. Eroarea de sincronizare între canale trebuie cunoscută, deoarece o diferență de câteva microsecunde poate introduce defazaje semnificative la frecvențe înalte.

Preprocesarea poate include validarea domeniului, eliminarea duplicatelor, tratarea valorilor lipsă, sincronizarea, reeșantionarea, filtrarea și scalarea. Fiecare transformare trebuie documentată și versionată, asociind rezultatul cu datele-sursă, intervalul procesat și versiunea aplicației. Metodele de extragere a comportamentului dispozitivelor din date eterogene au fost explorate și în contexte rezidențiale, oferind perspective utile pentru monitorizarea industrială [116].

Valorile lipsă pot fi păstrate explicit, interpolate, estimate sau pot constitui motivul pentru care o fereastră este eliminată. Completarea cu ultima valoare cunoscută poate fi adecvată pentru o stare discretă și periculoasă pentru un semnal dinamic. Durata absenței trebuie reținută, iar aplicația trebuie să deosebească o valoare măsurată de una estimată. Similar, valorile extreme nu se elimină automat: o valoare atipică poate proveni de la un senzor defect, o eroare de comunicație sau o tranziție reală. În multe situații, este de preferat ca valoarea să fie marcată ca suspectă și păstrată până când contextul permite interpretarea ei.

Sursele industriale pot folosi frecvențe de eșantionare și ceasuri diferite. Reeșantionarea creează o bază comună, dar poate introduce interpolare. Valorile cumulative se tratează altfel decât cele instantanee. Alinierea la același marcaj temporal nu dovedește o relație cauzală – trebuie luate în calcul și întârzierile fizice și de comunicație.

Standardizarea și normalizarea pot fi necesare pentru modelele statistice. Parametrii transformării se versionează împreună cu modelul. Regulile privind calcularea parametrilor exclusiv pe datele de dezvoltare și prevenirea scurgerii de informații sunt prezentate în secțiunea 5.4.

4.5 Filtrarea și reprezentările timp–frecvență

Filtrarea urmărește reducerea componentelor nedorite sau izolarea informației relevante. Media mobilă reduce variațiile rapide, dar introduce întârziere. Filtrul median este eficace împotriva impulsurilor.

Filtrele FIR (*Finite Impulse Response*) pot oferi fază liniară; filtrele IIR (*Infinite Impulse Response*) pot obține o tranziție mai abruptă cu un ordin mai mic, cu prețul unei faze neliniare. Filtrul se alege în funcție de fenomenul urmărit; o configurație potrivită pentru vizualizare poate elimina exact componentele de care diagnosticarea are nevoie. Frecvența de tăiere, ordinul și faza se administrează prin versiuni.

Transformata Fourier descompune un semnal în componente de frecvență, iar transformata Fourier rapidă (FFT – *Fast Fourier Transform*) permite calculul eficient al spectrului [115]. Pentru semnale aproximativ staționare, analiza poate evidenția armonici, benzi laterale și frecvențe caracteristice. Rezoluția spectrală a unei FFT este dată de raportul dintre frecvența de eșantionare și numărul de puncte: de exemplu, un semnal eșantionat la 10 kHz și analizat pe 4096 de puncte oferă o rezoluție de aproximativ 2.44 Hz. Scurgerea spectrală (*leakage*) apare când intervalul nu conține un număr întreg de perioade; aplicarea unei ferestre o reduce, dar modifică rezoluția și amplitudinea aparentă a componentelor [115]. Un singur spectru poate ascunde evoluția unui fenomen tranzitoriu.

Transformata Fourier pe termen scurt (STFT) calculează spectrul pe ferestre succesive și produce o spectrogramă, permițând observarea evoluției temporale a conținutului spectral. O fereastră scurtă oferă o localizare temporală mai bună; o fereastră lungă, o rezoluție spectrală mai bună. Compromisul este guvernat de principiul de incertitudine timp–frecvență: nu se poate obține simultan o localizare temporală arbitrar de fină și o rezoluție spectrală arbitrar de mare.

Transformata wavelet răspunde acestei limitări prin utilizarea unor funcții de bază cu suport variabil: ferestrele de analiză sunt mai scurte la frecvențe înalte (pentru a capta tranziții rapide) și mai lungi la frecvențe joase (pentru a rezolva componentele lent variabile). Transformata wavelet discretă descompune semnalul în coeficienți de aproximare și de detaliu pe mai multe niveluri succesive, permițând analiza multi-rezoluție. Este utilă pentru impulsuri, șocuri, descărcări parțiale și fenomene nestaționare. De exemplu, în diagnosticarea rulmenților, un șoc produs de o fisură pe calea de rulare produce un impuls de scurtă durată, care poate fi localizat mai precis cu ajutorul unei transformate wavelet decât cu o spectrogramă convențională.

Alegerea reprezentării trebuie justificată de mecanismul fizic, nu de comoditatea utilizării unei rețele neuronale antrenate pe imagini. Reprezentarea aleasă trebuie să păstreze informația relevantă și să poată fi calculată cu resursele platformei finale. Detaliile algoritmice pot fi dezvoltate în lucrarea de laborator dedicată monitorizării stării echipamentelor.

4.6 Extragerea caracteristicilor și reprezentarea temporală a datelor

După ce semnalele industriale au fost preprocesate și filtrate pentru eliminarea zgomotului (așa cum s-a prezentat în secțiunile 4.4–4.5), volumul masiv de eșantioane brute trebuie comprimat într-un set de indicatori sintetici relevanți pentru algoritmi de diagnosticare. Extragerea caracteristicilor (*feature extraction*) transformă fiecare fereastră temporală de date într-un vector compact de descriptori:

- În domeniul timp: se calculează descriptori statistici precum valoarea medie, valoarea eficace (RMS - *Root Mean Square* – indicator direct al energiei vibrațiilor), amplitudinea vârf-la-vârf, factorul de creastă, asimetria (*skewness*) și curtoza (*kurtosis* – deosebit de sensibilă la șocurile mecanice produse de primele fisuri ale rulmenților);
- În domeniul frecvență și timp–frecvență: se urmăresc frecvențele fundamentale de rotație, armonicile specifice componentelor defecte, energia integrată pe benzi spectrale critice și benzile laterale de modulație.

Corelarea corectă a acestor indicatori cu fizica procesului este cheia succesului: în timp ce RMS-ul indică o degradare generală a mașinii, distribuția armonicilor permite localizarea exactă a piesei defecte (inel interior, inel exterior sau corp de rostogolire).

Lungimea ferestrei, suprapunerea, filtrarea, rata de eșantionare și metoda de calcul fac parte din definiția caracteristicii. Aceste elemente trebuie menținute identic între dezvoltare și exploatare, fiindcă modificarea lor produce o reprezentare diferită. Caracteristicile se evaluează prin stabilitate, sensibilitate la fenomenul țintă, robustețe la variațiile normale și posibilitatea de a fi calculate cu resursele platformei finale [115].

Modelele temporare folosesc ordinea observațiilor pentru a surprinde dependențele între valoarea curentă, valorile trecute și intrările externe. Pot fi modele autoregresive, modele cu intrări exogene, reprezentări în spațiul stărilor sau arhitecturi de învățare automată pentru secvențe. Alegerea depinde de scop: prognoza, detectarea schimbării, simularea sau controlul reclamă criterii de validare distincte. Un model care oferă o predicție bună la un pas nu este automat adecvat pentru simularea pe termen lung [117], [118].

Caracteristicile și reprezentările temporale descrise în acest capitol constituie intrări pentru detecția anomaliilor, estimarea stării și prognoza degradării – teme dezvoltate în capitolul 5. Când obiectivul este reprezentarea dinamicii procesului în vederea predicției, simulării sau controlului, structurile modelelor, estimarea parametrilor și analiza reziduurilor sunt tratate în capitolul 7.

5. Detectarea anomaliilor și mentenanța predictivă

Detectarea anomaliilor și mentenanța predictivă utilizează datele despre proces și echipamente pentru a estima starea, a identifica degradarea și a susține intervențiile de mentenanță. Scopul nu este un scor algoritmic izolat, ci o informație suficient de timpurie și de credibilă pentru a decide continuarea funcționării, verificarea echipamentului, reducerea sarcinii, planificarea unei intervenții sau oprirea controlată [119], [120].

Capitolul prezintă definirea normalității, familiile de metode pentru detectarea anomaliilor, prognosticul și estimarea duratei de viață rămase, validarea modelelor, modalitățile de monitorizare și integrarea rezultatelor în procesul de mentenanță. Preprocesarea semnalelor și extragerea caracteristicilor au fost tratate în capitolul anterior.

5.1 Normalitatea, anomaliile și strategiile de mentenanță

Detectarea anomaliilor identifică observații, tipare sau secvențe care se abat de la comportamentul considerat normal pentru sistemul analizat [121], [122]. Normalitatea nu poate fi definită printr-un singur interval numeric atunci când instalația funcționează în mai multe regimuri. Aceeași temperatură, vibrație sau valoare a curentului poate fi normală la sarcină ridicată și suspectă la ralanti. Contextul include viteza, sarcina, produsul, rețeta, temperatura ambientală, etapa ciclului și modul de operare. Un model expus doar la funcționarea nominală poate interpreta pornirile, opririle și schimbările de produs drept defecte – caz în care problema nu este algoritmul, ci definirea incompletă a normalității [121].

O anomalie punctuală este o observație individuală neobișnuită – un vârf de temperatură, un impuls de vibrație sau un timp de ciclu mult mai mare decât precedentul. Ea poate corespunde unui fenomen real, unui senzor defect, unei erori de transmisie sau unei conversii numerice greșite. O anomalie contextuală este neobișnuită numai în raport cu regimul curent. O anomalie colectivă devine relevantă prin evoluția unei secvențe: creșterea lentă a temperaturii, apariția repetată a impulsurilor sau modificarea progresivă a spectrului. Valorile individuale pot rămâne în limite acceptate, deși tendința semnalează degradarea. Eliminarea automată a valorilor extreme poate ascunde tocmai evenimentele urmărite; în multe aplicații, este preferabil ca valoarea să fie păstrată împreună cu un indicator de calitate, până la analiza contextului.

Domeniul funcționării normale trebuie să cuprindă variațiile legitime de sarcină, produs, viteză și secvență. El poate fi reprezentat prin limite ingineresti, modele statistice, metode de grupare, modele probabilistice sau reziduurile unui model fizic. Definiția se revizuieste la înlocuirea senzorilor sau modificarea produsului, dar fără a permite ca o degradare lentă să fie absorbită automat în noua normalitate.

Indicatorii de stare se selectează în funcție de mecanismul de degradare, de regimul de funcționare și de timpul de avertizare necesar. Ei trebuie să fie sensibili la fenomenul urmărit, insensibili la variațiile normale și calculabili cu resursele disponibile. Caracteristicile statistice și spectrale au fost prezentate în secțiunile 4.5–4.6; în mentenanță, relevanța lor trebuie dovedită prin asocierea cu un mod de defectare și prin confirmarea obținută în urma inspecțiilor. Un singur indicator descrie rareori toate modurile de defectare, iar o corelație observată într-un caz nu demonstrează o relație fizică generală.

Rolul predicției se raportează la strategia de mentenanță [123]. Mentenanța corectivă, executată după pierderea funcției, se justifică pentru componente ieftine, necritice sau redundante. Mentenanța preventivă, bazată pe intervale calendaristice sau ore de funcționare, este simplă dar poate duce la înlocuiri premature sau poate rata defecte apărute înainte de termen. Mentenanța bazată pe stare declanșează intervenția când indicatorii depășesc criterii definite; presupune praguri validate și

proceduri de confirmare [119]. Mentenanța predictivă estimează evoluția probabilă, riscul într-un interval sau durata de viață rămasă (RUL-*Remaining Useful Life*). Mentenanța prescriptivă combină această estimare cu planul de producție, disponibilitatea pieselor și ferestrele de oprire. O predicție corectă nu conduce automat la aceeași intervenție: un activ redundant poate continua la sarcină redusă, în timp ce o anomalie minoră într-o componentă fără redundanță poate necesita prioritate maximă [119].

5.2 Familii de metode pentru detectarea anomaliilor

În multe aplicații industriale avem numeroase măsurări din funcționare normală, dar puține exemple confirmate pentru fiecare tip de defect. De aceea, nu putem întotdeauna antrena un model care să recunoască toate defecțiunile. Sunt utile metodele care pornesc de la comportamentul obișnuit al utilajului, inclusiv abordări nesupervizate sau semi-supervizate. Ele semnalează abaterile, dar o abatere nu înseamnă automat un defect: poate apărea și după schimbarea sarcinii, înlocuirea unui senzor sau modificarea produsului. O alarmă devine utilă când abaterea este verificată în raport cu regimul real al mașinii.

Tabelul 5.1 sintetizează și compară principalele familii de metode utilizate în detecția anomaliilor industriale.

Tabelul 5.1. Familii de metode pentru detectarea anomaliilor

Familie	Metode reprezentative	Situații potrivite	Limitări și moduri frecvente de eșec
Praguri, reguli și metode statistice	Praguri fixe/adaptive, EWMA, Shewhart, CUSUM, deviație mediană	Limite fizice cunoscute, regim staționar, date puține	Pragurile fixe ignoră schimbarea regimului; cele adaptive pot masca degradarea lentă
Învățare automată clasică	Regresie logistică, Random Forest, Gradient Boosting, SVM, k -NN, LOF (<i>Local Outlier Factor</i>), One-Class SVM	Caracteristici extrase clar, volume moderate de date	Sensibile la scalare și dezechilibru de clase; degradare în spații cu multe dimensiuni
Modele temporale și învățare profundă	Autoencodere, LSTM (<i>Long Short-Term Memory</i>), rețele convoluționale temporale (TCN), Transformer	Semnale complexe, dinamici temporale lungi, volume mari neetichetate	Cost computațional ridicat; explicabilitate redusă; risc de a reconstrui și anomaliile
Metode hibride fizică–date	Reziduuri pe modele fizice, cutie gri, constrângeri fizice în ML	Cunoaștere parțială a procesului, fenomene secundare greu de modelat	Erorile modelului fizic pot fi interpretate eronat ca defecte; integrare mai complexă

Metode mai recente, cum ar fi detectarea anomaliilor bazată pe arhitecturi de tip transformer sau pe rețele de atenție pe grafuri, extind capabilitățile modelelor temporale și ale învățării profunde, însă rămân dependente de volumul și calitatea datelor disponibile în context industrial.

Pragurile fixe pot fi exprimate prin condiții de tipul:

$$x(t) > P_s$$

pentru depășirea pragului superior P_s , respectiv:

$$x(t) < P_t$$

pentru atingerea unei valori sub pragul inferior P_i .

Pragurile P_s și P_i pot reprezenta limite fizice, recomandări ale producătorului, valori stabilite prin standarde sau niveluri derivate din date – semnificații care nu trebuie confundate. Pragurile adaptive pot reduce numărul alarmelor false, însă adaptarea trebuie raportată și limitată printr-o referință stabilă; în caz contrar, sistemul poate ajunge să considere degradarea progresivă drept stare normală.

Metodele statistice urmăresc schimbările de distribuție sau acumularea abaterilor mici. O fereastră scurtă reacționează rapid dar generează alarme false; o fereastră lungă este stabilă dar detectează mai târziu. CUSUM (*Cumulative Sum*) și EWMA (*Exponentially Weighted Moving Average*) pot identifica modificări persistente mai devreme decât un prag instantaneu, dar necesită o referință coerentă. Metodele bazate pe distanță și densitate compară observația cu vecinii sau cu distribuția de referință; scalarea este esențială, iar distanța Mahalanobis – care măsoară distanța unui punct față de o distribuție, ținând cont de covarianța variabilelor – tratează corelațiile dar devine instabilă când datele sunt puține.

Învățarea supervizată poate clasifica explicit modurile de defectare, dar depinde de calitatea etichetelor. Înregistrările de mentenanță descriu adesea simptome și acțiuni, nu cauza confirmată. Etichetele trebuie să distingă simptomul observat, defectul suspectat, cauza confirmată și rezultatul verificării. În practică, exemplele de defecte autentice sunt mult mai puține decât cele normale, producând un dezechilibru între clase; un model care favorizează clasa normală poate rata aproape toate evenimentele importante [124].

Când există date normale abundente și puține defecte confirmate, se pot folosi metode de grupare, modele de densitate, SVM (*Support Vector Machine*) cu o singură clasă și autoencodere. Acestea învață distribuția normală și semnalează abaterile, putând detecta tipuri de defecte absente din antrenare. Totuși, schimbarea produsului sau înlocuirea unui senzor poate genera scoruri ridicate fără un defect real. Un autoencoder prea încâpător poate reconstrui și anomalii; unul prea restrictiv poate respinge variațiile normale. Pragul se stabilește pe date independente și se evaluează prin alarme false, evenimente ratate și confirmare fizică.

Modelele temporale exploatează ordinea eșantioanelor și pot detecta schimbări ale dinamicii chiar și când valorile individuale rămân normale. Rețelele recurente, convoluționale temporale și arhitecturile Transformer pot surprinde dependențe pe termen lung, dar costul de calcul trebuie justificat. Lungimea ferestrei și rata de eșantionare trebuie păstrate identice între antrenare și exploatare.

Metodele hibride combină modele fizice, reguli și reprezentări învățate. De exemplu, un model termic estimează temperatura așteptată, iar detectorul analizează reziduul. Regulile pot dezactiva alarma în timpul pornirii sau pot impune o limită absolută. Soluțiile hibride păstrează o legătură mai clară cu procesul, dar ambele componente trebuie versionate și validate [59], [121].

5.3 PHM, estimarea RUL și incertitudinea

Prognosticul și managementul stării de sănătate PHM (*Prognostics and Health Management*) integrează monitorizarea, diagnosticul, prognosticul și sprijinul decizional. Diagnosticul urmărește starea actuală și cauza probabilă; prognosticul estimează evoluția viitoare. PHM leagă aceste rezultate de obiectivele mentenanței [119], [120].

Durata de viață utilă rămasă RUL (*Remaining Useful Life*) reprezintă timpul sau utilizările rămase până la pierderea funcției. RUL nu este o proprietate absolută: depinde de definiția defectului, de sarcina viitoare și de condițiile de operare. Același rulment poate avea durate rămase diferite după cum utilajul funcționează la sarcină nominală sau redusă [120], [125].

Modelele de prognostic pot fi bazate pe fizică, pe cunoaștere sau pe date. Modelele fizice (propagarea fisurilor, uzura, coroziunea) sunt interpretabile dar necesită parametri preciși. Modelele bazate pe cunoaștere folosesc reguli experte și arbori de defectare; sunt utile când experiența există, dar întreținerea regulilor devine dificilă la modificarea procesului. Modelele pe bază de date utilizează regresia, procese gaussiene sau analiza supraviețuirii și necesită exemple reprezentative ale degradării. Totuși, secvențele complete de degradare până la defectare se întâlnesc rar în exploatare, deoarece componentele sunt adesea înlocuite preventiv, producând date cenzurate. Analiza supraviețuirii modelează timpul până la eveniment și poate utiliza astfel de cazuri, estimând probabilitatea de supraviețuire sau rata instantanee de defectare [126].

Modelele hibride combină un model fizic pentru tendința generală cu un model pe bază de date pentru efectele sarcinii și mediului. Incertitudinea ambelor componente trebuie estimată și propagată către rezultat.

O predicție exprimată printr-o valoare punctuală deterministă (de tipul $RUL = 120$ ore) creează o impresie falsă de precizie și este riscantă operațional, deoarece ignoră variabilitatea procesului de degradare. O formulare corectă și robustă este cea probabilistică, exprimată printr-un interval de încredere de exemplu: $RUL_{median} = 120$ ore, cu intervalul de încredere $I = [80, 170]$ ore la un nivel de 90%). Această abordare permite inginerului de mentenanță să ia decizii conservatoare bazate pe limita inferioară a intervalului (80 de ore). Incertitudinea provine din zgomotul de măsurare, parametrii modelului, sarcina viitoare necunoscută și mecanismele fizice neobservate direct.

Evaluarea prognosticului utilizează MAE, RMSE, eroarea relativă și orizontul prognostic – momentul anterior defectului la care predicția devine suficient de stabilă pentru a susține o intervenție [127]. Costurile erorilor sunt asimetrice: o predicție optimistă poate permite avaria; una conservatoare duce la înlocuiri premature. Probabilitățile trebuie calibrate: dacă un model estimează o probabilitate de 0,8 pentru defectare, aproximativ 80% din cazuri ar trebui să se confirme. Calibrarea se verifică separat pentru active, regimuri și perioade relevante.

Provocarea centrală în estimarea RUL nu este doar precizia punctuală, ci consistența predicției pe măsură ce date noi devin disponibile. Pe măsură ce un defect progresează, semnalul purtător de informație (de exemplu, amplitudinea unei armonici, energia dintr-o bandă spectrală) devine mai evident, iar incertitudinea scade. O predicție utilă din punct de vedere operațional este cea care atinge o stabilitate suficientă cu suficient timp înainte de defectare pentru a permite planificarea intervenției — acesta este conceptul de orizont prognostic, definit ca momentul în care eroarea de predicție intră și se menține într-un interval acceptabil [127].

Modelele de supraviețuire (*survival analysis*) oferă un cadru natural pentru estimarea RUL atunci când datele conțin observații cenzurate — componente care nu au atins încă defectarea la momentul analizei. Estimarea Kaplan-Meier calculează probabilitatea empirică de supraviețuire dintr-un set de date eterogen; modelul Cox (*proportional hazards*) leagă această probabilitate de covariate (caracteristici extrase, condiții de operare), permițând estimarea riscului instantaneu de defectare în funcție de starea observată a echipamentului [126]. Deși aceste modele nu oferă o curbă continuă de degradare, ele sunt valoroase atunci când istoricul conține multe echipamente similare cu durate de viață și condiții de operare diferite.

Modelele bazate pe procese stochastice (proces Wiener, Gamma, Markov) descriu evoluția aleatoare a indicatorului de degradare și permit actualizarea predicției pe măsură ce noi măsurări devin disponibile, de obicei prin filtrare bayesiană. Sunt potrivite atunci când degradarea este graduală și măsurătorile sunt bruiate, dar necesită specificarea unei legi de evoluție și a unui prag de defectare.

Un exemplu orientativ: pentru un rulment monitorizat prin vibrații, RMS-ul accelerației filtrate în banda 2–10 kHz poate servi ca indicator de degradare. Primele 100 de ore de funcționare definesc referința normală. O creștere de 6 dB față de referință, confirmată pe trei măsurători consecutive, declanșează o alarmă de avertizare. Un model de regresie exponențială ajustat pe datele istorice de la rulmenți similari estimează RUL pe baza pantei curente a indicatorului, cu un interval de încredere de 80%. Valoarea operațională apare atunci când intervalul inferior al predicției depășește timpul necesar pentru comandarea piesei și programarea intervenției, nu doar când estimarea punctuală pare alarmantă.

5.4 Validarea și exploatarea de încredere a modelelor

Principiile acestei secțiuni se aplică tuturor modelelor bazate pe date, indiferent de rezultatul furnizat. Particularitățile aplicației pot reclama metrici suplimentare, dar nu înlătură obligația separării corecte a datelor, prevenirii scurgerilor de informații și monitorizării derivei.

Defectele industriale apar cu o frecvență foarte redusă, iar acuratețea poate fi înșelătoare: dacă doar 0.1% din ferestre conțin un defect, un model care prezice mereu „normal” atinge o acuratețe de 99.9% fără să detecteze nimic. Precizia, sensibilitatea și scorul F1 descriu performanța clasificării, dar sunt insuficiente pentru evaluarea operațională. Trebuie urmărite alarmele false per activ și per lună, evenimentele ratate, timpul de avertizare și proporția alarmelor care conduc la o inspecție confirmată [124].

Pentru un defect produs la momentul t_f și detectat de model la momentul t_d , timpul util de avertizare (*lead time*, $L = t_f - t_d$) reprezintă intervalul disponibil pentru reacție. Acest interval trebuie să depășească durata necesară aprovizionării cu piese de schimb și planificării echipei de intervenție. O alarmă emisă după ultimul moment util nu mai are valoare operațională, în timp ce o alertă declanșată mult prea devreme riscă să fie ignorată dacă rămâne activă luni de zile cu o incertitudine ridicată.

Evaluarea trebuie să îmbine criterii statistice și operaționale. Cele statistice includ precizia, sensibilitatea, scorul F1, calibrarea probabilităților și eroarea de predicție. Cele operaționale includ numărul de alarme false per activ și per interval de timp, evenimentele importante nedetectate, timpul de avertizare, proporția alarmelor care duc la o verificare, proporția verificărilor care confirmă problema, volumul de rezultate pe care echipa îl poate analiza și efectul asupra opririlor, costurilor și intervențiilor inutile. Nicio măsură nu se interpretează izolat: un model poate avea o sensibilitate ridicată și poate fi inutilizabil dacă generează mai multe alarme decât poate organizația să proceseze.

Validarea trebuie să reproducă situația reală de utilizare [32]. Principii similare de separare riguroasă a datelor, inclusiv separarea temporală, sunt subliniate și în ghidurile de validare din literatura medicală [128]. Împărțirea aleatorie a ferestrelor este adesea nevalidă, deoarece segmente din același ciclu pot ajunge atât în antrenare cât și în testare, producând scurgeri de informații. Separarea se face după timp, activ, produs, campanie sau eveniment de defectare. Normalizarea și ajustarea pragurilor se efectuează exclusiv pe datele de antrenare și validare.

Deriva conceptului descrie modificarea în timp a relației intrare–ieșire [129]. Deriva datelor se referă la schimbarea distribuției intrărilor; deriva etichetelor, la schimbarea frecvenței rezultatelor. Monitorizarea trebuie să urmărească distribuția indicatorilor, frecvența alarmelor, incertitudinea și performanța confirmată. Detectarea derivei nu înseamnă automat că modelul este greșit, ci că ipotezele trebuie reverificate. [129].

Reantrenarea automată necontrolată poate transforma degradarea în comportament normal. Datele de actualizare trebuie validate, iar versiunea rezultată testată înainte de introducerea în exploatare. Pragurile de alarmă se versionează împreună cu modelul: valoarea anterioară, valoarea nouă, motivul

modificării și aprobarea se păstrează. Modelul este o componentă operațională supusă guvernantei și retragerii controlate [32].

Explicabilitatea se adaptează utilizatorului. Tehnicianul are nevoie de semnalele contributive, tendința observată, regimul de funcționare și nivelul de încredere. Importanța caracteristicilor nu dovedește cauzalitatea, iar explicația sprijină verificarea fizică, nu o înlocuiește.

Autoritatea se stabilește independent de performanța statistică. Un model consultativ prezintă informația; unul de supraveghere poate solicita inspecții. Acționarea directă asupra procesului impune cerințe mult mai stricte. Mecanismul de rezervă se definește pentru situații cu încredere scăzută, senzori indisponibili sau depășire a timpului de inferență. Ultima predicție nu se reutilizează nelimitat; rezultatul trebuie să aibă o perioadă de valabilitate definită.

5.5 Modalități de monitorizare a stării echipamentelor

Starea unui echipament poate fi observată prin semnale mecanice, termice, electrice, chimice și acustice, precum și prin indicatorii de proces [130]. Alegerea depinde de mecanismul de degradare, de accesul la componentă, de cost și de timpul de avertizare necesar. Principalele modalități sunt sintetizate în Tabelul 5.2.

În monitorizarea vibrațiilor, indicatorii trebuie interpretați în raport cu turația, sarcina și calea de propagare. Urmărirea ordinelor de rotație și analiza anvelopei pot evidenția defecte specifice ale rulmenților sau angrenajelor. Temperatura confirmă frecarea sau suprasarcina, semnăturile electrice reflectă efecte ale motorului și sarcinii, iar analiza lubrifiantului furnizează dovezi despre contaminare și uzură. O variație observată printr-o singură modalitate nu constituie, de regulă, o confirmare suficientă a cauzei.

Fuziunea multimodală combină sensibilități complementare – vibrația poate indica un tipar, temperatura confirmă frecarea, curentul evidențiază sarcina, lubrifiantul atestă uzura. Fuziunea se poate face la nivel de date brute, caracteristici sau decizii. Mai mulți senzori nu garantează o performanță mai bună dacă o sursă este slabă sau nesincronizată [119], [120].

Tabelul 5.2 enumeră modalitățile de monitorizare; utilitatea lor practică depinde de asocierea cu indicatori concreți, calculabili și interpretabili. Câteva exemple orientative:

- **Rulment cu elemente rulante:** valoarea efectivă (RMS) a vibrației în banda 500–5000 Hz, raportul dintre valoarea de vârf și valoarea efectivă (factor de creastă), gradul de aplatizare a distribuției amplitudinilor (curtoză) și frecvențele caracteristice defectelor — trecerea bilelor peste inelul exterior, trecerea peste inelul interior și frecvența de rotire a corpurilor de rulare — extrase din anvelopa semnalului.
- **Angrenaj:** indicatorul FM4 (derivat din semnalul diferență), energia benzilor laterale din jurul frecvenței de angrenare și raportul dintre amplitudinea benzilor laterale și cea a frecvenței de angrenare.
- **Motor electric asincron:** amplitudinea componentelor la frecvența de alunecare și la frecvențele armonicilor de bobinaj, dezechilibrul curenților de fază și spectrul cuplului electromagnetic estimat.
- **Pompă centrifugă:** presiunea diferențială, puterea absorbită, spectrul vibrațiilor la frecvența de trecere a paletelor și analiza anvelopei pentru defecte de rulment.
- **Transformator de putere:** analiza gazelor dizolvate (DGA), măsurarea descărcărilor parțiale și răspunsul dielectric în domeniul frecvență.

Acești indicatori trebuie interpretați împreună cu regimul de funcționare (sarcină, turație, temperatură), iar tendința lor temporală este adesea mai informativă decât valoarea instantanee.

Tabelul 5.2. Modalități uzuale de monitorizare a stării echipamentelor

Modalitate	Fenomene observate	Avantaje	Limitări și context necesar
Vibrații	Dezechilibru, nealiniere, jocuri mecanice, defecte de rulmenți și angrenaje, rezonanță	Sensibilitate ridicată la mașini rotative; permite analiză spectrală (FFT), anvelopă și urmărire de ordine	Depinde de turație, sarcină, montaj și calea de propagare; amplitudinea mare nu indică singură cauza
Temperatură și termografie	Frecare, suprasarcină, răcire insuficientă, ungere deficitară, rezistențe de contact	Inspecție fără contact; confirmă direct disiparea termică anormală	Răspuns lent (inerție termică); influențată de emisivitate, reflexii, distanță și mediu
Semnături electrice	Asimetrii de fază, bare rotorice rupte, izolație degradată, fluctuații de sarcină	Folosește traductoarele existente din tablou; evită montarea senzorilor pe mașină	Necesită decuplarea perturbațiilor din rețea, convertizor, motor și proces
Lubrifiant și particule	Contaminare cu apă/praf, oxidare, degradare aditivi, particule de uzură	Compoziția și morfologia particulelor identifică exact componenta uzată	Prelevare periodică discontinuă; depinde strict de punctul și procedura de eșantionare
Emisie acustică și ultrasunete	Microfisurare incipientă, frecare uscată, scurgeri de presiune, descărcări parțiale	Detectează fenomene tranzitorii de înaltă frecvență și viteze foarte lente	Sensibilitate mare la zgomotul de fond și la poziționarea traductorului pe carcasă
Indicatorii procesului	Timp de ciclu, consum energetic specific, variație debit/presiune, rebuturi	Măsurări nativ fără senzori dedicați; reflectă direct impactul operațional	Influențați simultan de rețetă, sarcină, uzură, calitatea materiei prime și operator

5.6 Strategii de monitorizare și implementarea distribuită

Monitorizarea continuă colectează date permanent și permite detecția timpurie și generarea automată de alarme [119], [120]. Conceptul de monitorizare la distanță a parametrilor critici este folosit pe scară largă și în domeniul medical [131], [132], [133]. Costurile includ senzorii, infrastructura de comunicație, stocarea și mentenanța sistemului. Monitorizarea periodică pe trasee, cu instrumente portabile, poate fi suficientă pentru active cu risc redus. O strategie hibridă alocă monitorizare continuă activelor critice și măsurări periodice celor cu risc mediu. Frecvența de măsurare trebuie corelată cu viteza de evoluție a mecanismului.

În implementarea distribuită, rezultatul transmis platformei centrale trebuie asociat cu fereastra diagnostică, regimul de funcționare, marcasele temporale și versiunea modelului, conform mecanismului memoriei tampon circulare și repartizării funcțiilor între edge și cloud prezentate în secțiunea 3.1. Pierderea comunicației nu trebuie să întrerupă monitorizarea locală și nici să ducă la reutilizarea nelimitată a ultimei estimări; se definesc, așadar, durata de păstrare locală, prioritățile alarmelor și modul de reconciliere după reconectare. Totodată, implementarea trebuie să colecteze confirmarea într-o formă utilizabilă: ordinul de lucru trebuie să înregistreze simptomul, componenta inspectată, cauza confirmată și intervenția realizată, iar aceste informații se asociază cu alarma și datele care au generat-o.

Obiectivul predicției trebuie formulat cu precizie: evenimentul, orizontul și acțiunea pe care o face posibilă. „Predicția defectului” este o formulare prea vagă. Un obiectiv operațional poate fi estimarea

probabilității de defectare a unui rulment în următoarele 14 zile, pentru a comanda piesa și a programa intervenția. Capacitatea operațională se analizează înainte de introducerea în exploatare: frecvența estimată a alarmelor se compară cu resursele disponibile pentru verificare.

5.7 Integrarea cu procesul de mentenanță și verificarea rezultatului

Alarma trebuie integrată cu fluxul de planificare și execuție a intervențiilor. Ea se asociază cu activul, componenta, prioritatea, riscul și acțiunea recomandată. O notificare generică de tipul „anomalie detectată” are valoare redusă; mesajul trebuie să indice ce s-a modificat, în ce regim, cu ce nivel de încredere și ce verificare se recomandă.

Sistemul CMMS (*Computerized Maintenance Management System*) poate genera ordinul de lucru și asocia piesele; MES-ul furnizează programul producției și ferestrele de oprire; ERP-ul oferă costul și disponibilitatea pieselor. Standarde precum FHIR, din domeniul medical, oferă lecții utile pentru interoperabilitatea semantică în mentenanță [134]. Integrarea nu presupune că fiecare alarmă generează automat o intervenție.

Prioritizarea automată a ordinelor de lucru în sistemul CMMS nu se bazează pe o simplă regulă statică, ci pe evaluarea dinamică a cinci factori operaționali:

1. Probabilitatea de defectare (P_d): severitatea degradării estimate de modelul PHM;
2. Criticitatea utilajului (C_p): impactul opririi asupra fluxului general de fabricație și riscul de securitate;
3. Timpul util rămas (R): orizontul până la defectarea funcțională și existența unei linii paralele redundante;
4. Durata estimată a reparației (T_i): timpul necesar pregătirii, izolării și intervenției tehnice;
5. Fereastra de oportunitate (F_o): sincronizarea reparației cu pauzele de schimb sau cu opririle tehnologice deja planificate în MES.

Evaluarea coroborată a acestor parametri permite planificarea optimă a intervențiilor, fără a perturba inutil comenzile aflate în execuție. Analiza detaliată a riscului și a funcționării în regim degradat este prezentată în secțiunile 8.1–8.2.

Procesul trebuie să încheie bucla operațională prezentată în Figura 5.1.

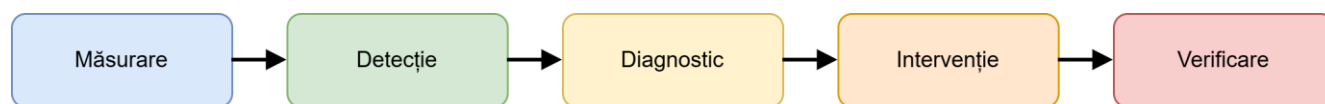


Figura 5.1. Bucla operațională a monitorizării stării și a mentenanței predictive

După reparație se verifică dacă indicatorii au revenit la valorile așteptate. Dacă alarma persistă, cauza poate fi alta, intervenția poate fi incompletă sau modelul poate fi incorect. Rezultatul verificării și evoluția indicatorilor se asociază înregistrării operaționale definite în secțiunea 5.6; astfel se poate stabili dacă alarma a fost corectă și dacă intervenția a produs efectul scontat. Rezultatele se compară cu strategia anterioară sau cu o referință operațională echivalentă, utilizând criteriile din secțiunea 5.4, fără a atribui automat efectele modelului atunci când, în aceeași perioadă, s-au modificat politica de mentenanță, piesele utilizate, furnizorii, personalul sau regimul de producție. Abia atunci când rezultatul intervenției este verificat și folosit pentru confirmarea diagnosticului și pentru îmbunătățirea modelelor, a pragurilor și a procedurilor, valoarea mentenanței predictive este pe deplin demonstrată.

6. Viziune artificială și inspecție optică industrială

Inspecția optică industrială folosește imagini pentru a verifica prezența, poziția, geometria, aspectul și conformitatea produselor. Pentru a decide dacă o piesă este conformă, stația de inspecție trebuie mai întâi să obțină o imagine în care defectul căutat poate fi văzut. De aceea, înainte de a alege algoritmul, stabilim ce măsurăm, cum iluminăm piesa, unde așezăm camera și cât timp avem pentru decizie. Capitolul urmărește întregul traseu: formarea imaginii, analiza ei, validarea rezultatului și comanda de respingere a piesei. O schimbare de iluminare, poziție sau expunere poate modifica imaginea fără ca piesa să se fi schimbat. Evaluăm astfel algoritmul împreună cu stabilitatea condițiilor în care este făcută măsurarea [135].

6.1 Arhitectura stației și formularea cerinței de inspecție

Lanțul funcțional al unei stații de inspecție optică automată (AOI - *Automated Optical Inspection*) este ilustrat în Figura 6.1.

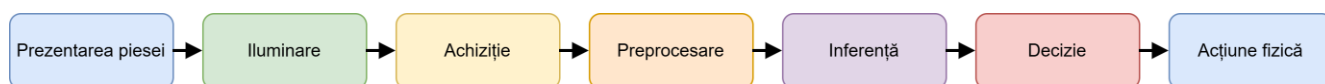


Figura 6.1. Lanțul funcțional al unei stații de inspecție optică automată

Figura 6.2 detaliază componentele fizice ale stației și fluxul de semnale dintre achiziție, procesare și acțiunea de respingere.

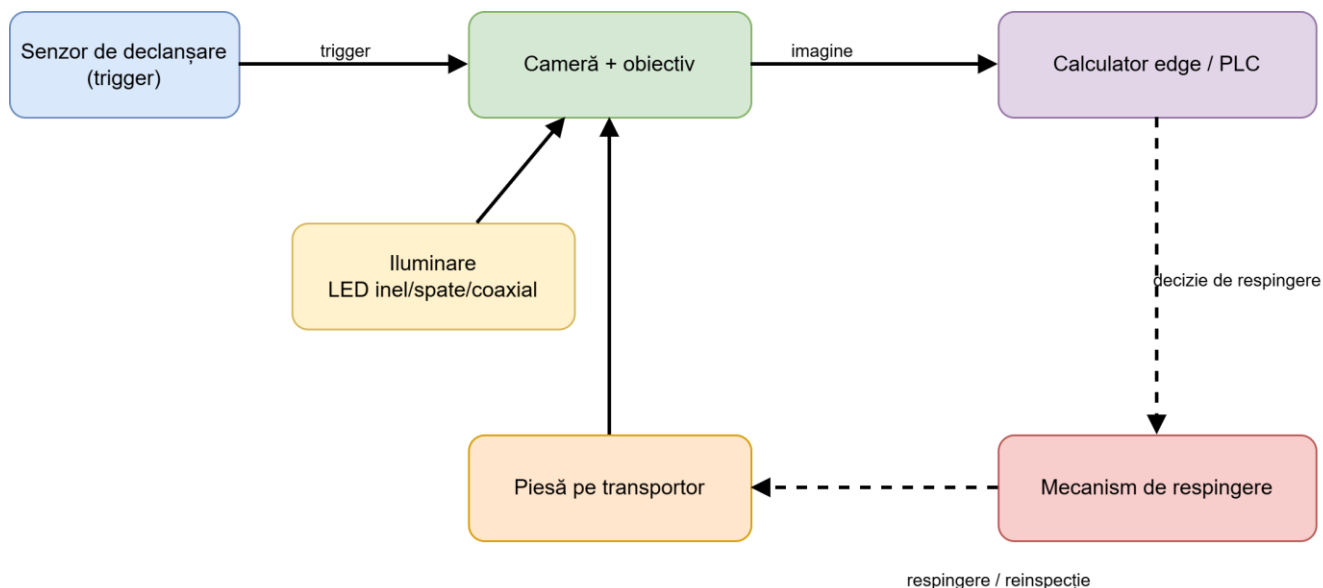


Figura 6.2. Schema unei stații de inspecție optică automată (AOI)

Lanțul se completează cu identificarea produsului, selectarea rețetei, urmărirea piesei, trasabilitatea imaginilor și verificarea periodică a funcției. Dacă piesa este prezentată ambiguu sau imaginea este neclară, rezultatul poate deveni instabil, indiferent de complexitatea algoritmului.

Proiectarea începe cu formularea exactă a cerinței: tipurile de defect, dimensiunea și contrastul minim, rata acceptabilă de erori, timpul maxim de decizie și acțiunea asociată. Cerința poate viza prezența unei componente, citirea unui cod, verificarea unei dimensiuni sau deformarea tridimensională. Criteriul

trebuie legat de efectul industrial: un defect vizibil dar fără impact funcțional poate să nu justifice respingerea; o discontinuitate foarte mică poate fi critică pentru etanșare sau rezistență structurală.

Dimensiunea minimă se exprimă în unități fizice și se transformă într-o cerință de rezoluție. Un defect care ocupă teoretic un singur pixel nu poate fi separat robust de zgomot și neuniformitatea iluminării. Cerința trebuie să precizeze și comportamentul când inspecția nu poate fi realizată: rezultatul poate impune repetarea achiziției, reinspecția sau respingerea preventivă. Prezentarea obiectului trebuie, de asemenea, să fie vizibilă și repetabilă. Poziționarea poate recurge la ghidaje, dispozitive de fixare, transportoare sincronizate sau localizare vizuală. Un dispozitiv mecanic simplu și repetabil este adesea mai robust decât antrenarea modelului pentru variații care puteau fi eliminate fizic. Dacă produsul poate apărea în mai multe orientări, stația fie îl poziționează mecanic, fie identifică orientarea înaintea inspecției.

Alegerea formulării algoritmice depinde de rezultatul necesar, nu de popularitatea unei arhitecturi. Tabelul 6.1 sintetizează principalele formulări și limitările lor. Clasificarea este potrivită când piesa este localizată și o decizie globală este suficientă. Detecția este utilă când pot exista mai multe regiuni sau componente. Segmentarea permite aplicarea de reguli dimensionale. Detectarea anomaliilor este relevantă când există multe produse conforme și puține defecte. Inspecția tridimensională se justifică atunci când geometria, nu aspectul superficial, constituie proprietatea de interes.

Tabelul 6.1. Formulări uzuale ale problemelor de inspecție

Formulare	Rezultat	Utilizare caracteristică	Limitare principală
Clasificare	Etichetă globală per imagine	Verificarea prezenței componentei sau a tipului de piesă	Nu localizează defectul în imagine
Detecție de obiecte	Dreptunghi de încadrare (Bounding box) + clasă	Identificarea componentelor pe PCB sau a defectelor locale	Aproximare grosieră a geometriei defectului
Segmentare semantică	Mască de pixeli clasificați	Măsurarea precisă a porilor, fisurilor sau cordoanelor	Cost ridicat de etichetare și calcul
Detecție de anomalii	Scor de anomalie și hartă de abateri	Inspecție de suprafețe complexe cu defecte rare/imprevizibile	Sensibilă la variații normale de iluminare și textură
Metrologie 3D	Coordonate spațiale, profil de înălțime	Verificarea coplanarității, volumului și geometriei piesei	Cost hardware ridicat; timp de achiziție mai mare

6.2 Iluminarea, optica, aparatul de achiziție și calibrarea

Calitatea imaginii trebuie proiectată, nu presupusă. În numeroase aplicații, alegerea corectă a iluminării și a opticii aduce un câștig mai mare decât înlocuirea modelului de învățare automată. Iluminarea se selectează pentru a pune în evidență proprietatea inspectată, nu pentru a obține o imagine plăcută vizual.

Iluminarea din spate convine pentru siluete și măsurări dimensionale. Iluminarea în câmp luminos evidențiază suprafețele uniforme; cea în câmp întunecat face vizibile zgârieturile și muchiile. Iluminarea coaxială este utilă pentru suprafețe reflectante, iar cea difuză de tip cupolă reduce reflexiile pe obiecte curbe. Polarizarea diminuează reflexiile speculare; iluminarea ultravioletă, infraroșie sau multispectrală evidențiază caracteristici greu vizibile în spectrul obișnuit. Lumina structurată și profilarea cu laser se folosesc pentru recuperarea geometriei tridimensionale [135].

Intensitatea, uniformitatea, unghiul și degradarea surselor trebuie controlate. Scăderea intensității LED-urilor sau depunerile pe suprafețele de protecție pot schimba domeniul imaginilor fără ca produsul să se fi modificat. Pentru aplicațiile critice se pot folosi referințe interne și verificări periodice.

Alegerea camerei și a obiectivului depinde de câmpul vizual, dimensiunea minimă a defectului, rezoluția spațială, adâncimea de câmp și frecvența de achiziție. Obturatorul global este preferabil pentru obiecte în mișcare rapidă. Expunerea trebuie să limiteze neclaritatea de mișcare fără a produce subexpunere. Corecția software a distorsiunii nu recuperează rezoluția optică pierdută. Adâncimea de câmp se raportează la toleranța de poziție a obiectului; pentru măsurări precise pot fi necesare obiective telecentrice, iar selecția se face pe baza cerinței metrologice, nu doar a numărului de megapixeli.

Calibrarea include corecția distorsiunii, determinarea relației pixel–unitate fizică, alinierea multicameră și calibrarea cameră–robot. Se păstrează procedura, artefactul de calibrare, data și configurația optică. Calibrarea se repetă după modificarea camerei, obiectivului sau poziției mecanice. O calibrare geometrică validă nu garantează măsurări corecte dacă piesa nu se află în planul presupus sau focalizarea s-a degradat [135].

Imaginea trebuie achiziționată într-o poziție repetabilă. Semnalul de declanșare poate veni de la un senzor fotoelectric, encoder sau PLC. Întârzierea totală, de la detectarea piesei până la acțiunea fizică, este:

$$T_{\text{total}} = T_{\text{trigger}} + T_{\text{exposure}} + T_{\text{transfer}} + T_{\text{preprocess}} + T_{\text{inference}} + T_{\text{postprocess}} + T_{\text{communication}}$$

unde:

- T_{trigger} reprezintă întârzierea dintre detectarea piesei și inițierea achiziției;
- T_{exposure} reprezintă timpul de expunere;
- T_{transfer} reprezintă timpul necesar transferului imaginii;
- $T_{\text{preprocess}}$ reprezintă timpul necesar preprocesării imaginii;
- $T_{\text{inference}}$ reprezintă timpul de execuție al modelului;
- $T_{\text{postprocess}}$ reprezintă timp necesar postprocesării rezultatului;
- $T_{\text{communication}}$ reprezintă timpul necesar comunicării rezultatului către sistemul de control.

Trebuie evaluată limita maximă relevantă, nu doar media. Expunerea, transferul, alocarea de memorie, încărcarea acceleratorului și serviciile de fundal pot produce vârfuri rare de latență. Identitatea piesei se păstrează de la trigger până la punctul de separare, prin identificatori, poziții urmărite, registre FIFO, encoder sau integrarea cu sistemul de transport [2].

Preprocesarea include corecția distorsiunii, normalizarea intensității, reducerea zgomotului, decuparea și localizarea regiunii de interes. Transformările se versionează împreună cu modelul. Filtrarea excesivă poate elimina zgârieturi și muchii fine împreună cu zgomotul. Înaintea inferenței, se verifică validitatea imaginii: expunerea, saturarea, neclaritatea, poziția piesei și disponibilitatea iluminării. O imagine neclară nu este un defect al produsului; stația trebuie să distingă „produs neconform” de „inspecție invalidă”. Tratarea piesei asociate unei inspecții invalide este definită în secțiunea 6.6.

6.3 Metode clasice, învățare profundă și detectarea anomaliilor

Metodele clasice și cele bazate pe învățare profundă nu trebuie opuse ca generații succesive. O stație robustă le poate îmbina: o metodă geometrică localizează piesa, un model neuronal interpretează aspectul, iar o regulă dimensională validează rezultatul.

Filtrarea, operațiile morfologice, detectarea muchiiilor și potrivirea cu șabloane rămân adecvate când geometria și condițiile de achiziție sunt controlate. Aceste metode necesită puține date, permit toleranțe explicite și au un timp de execuție predictibil [135].

Rețelele neuronale convoluționale (CNN - *Convolutional Neural Network*) învață reprezentări ierarhice direct din imagini. Toleranța lor la variații depinde de reprezentarea acestora în datele de antrenare; modelul poate exploata asocieri parazite – fundalul, dispozitivul de fixare sau particularitățile unui lot. Detectoarele în două etape, precum Faster R-CNN, generează regiuni candidate înaintea clasificării; cele într-o singură etapă, precum YOLO (*You Only Look Once*), realizează direct predicția [136], [137].

Segmentarea atribuie o clasă sau o probabilitate fiecărui pixel. U-Net, cu arhitectura sa codificator-decodificator și cu legăturile între nivelurile de rezoluție, stă la baza multor aplicații de segmentare [138]. Metoda este potrivită pentru măsurarea suprafeței unui defect, a lungimii unei fisuri sau pentru aplicarea unor criterii dimensionale, însă reclamă adnotări detaliate. Contururile defectului pot fi ambigue, iar măștile produse de inspectori diferiți pot varia. Criteriile de etichetare trebuie formulate operațional; o mască nu reprezintă întotdeauna un adevăr absolut [138].

Arhitecturile de tip Transformer pentru viziune împart imaginea în fragmente și folosesc mecanisme de atenție pentru a modela relațiile globale [139]. Preantrenarea pe seturi mari poate furniza reprezentări transferabile, dar diferența dintre imaginile generice și cele industriale trebuie evaluată. Costul de calcul, volumul de date și timpul maxim de execuție pot face ca un CNN compact sau o metodă clasică să fie mai potrivite pentru o aplicație stabilă, cu termene-limită stricte. Un model mai mare nu se alege doar pentru că este mai nou [139].

În inspecția industrială, situația frecventă este aceea a numeroaselor imagini conforme și a foarte puținelor exemple cu defecte. Detectarea anomaliilor învață aspectul normal și produce un scor – sau o hartă – pentru abateri. Această abordare este adecvată atunci când exemplele de defecte sunt insuficiente pentru antrenare, când nu se cunosc toate tipurile posibile sau când adnotarea este dificilă. O abatere vizuală nu certifică însă un defect fizic: o variantă legitimă de material, un nou furnizor, o nuanță diferită ori înlocuirea iluminatorului pot genera scoruri mari [121].

Metodele bazate pe memorii de caracteristici extrag reprezentări locale din imaginile normale și compară imaginile noi cu această memorie. PatchCore utilizează distanțele dintre fragmentele de imagine pentru detecție și localizare și oferă rezultate bune fără exemple defecte; consumul de memorie și variațiile normale neincluse în antrenare pot însă limita utilizarea. Setul normal trebuie să acopere loturile, materialele, pozițiile și condițiile optice legitime [140].

Metodele de reconstrucție transpun în plan vizual principiul din secțiunea 5.2: imaginea normală este reconstruită, iar diferența față de intrare localizează anomalia [141]. DRAEM, de exemplu, învață reconstrucția și discriminarea regiunilor neobișnuite cu ajutorul unor anomalii sintetice. Datele sintetice trebuie să respecte materialul, iluminarea, geometria și mecanismul defectului; altminteri, modelul poate învăța marginile și texturile create de generator, nu fenomenul real [141]. Alte abordări, precum BlindSpotNet, exploatează mascarea regiunilor pentru a detecta abaterile de la normalitate [142].

Modelele de tip profesor-elev și reprezentările vizual-lingvistice, precum WinCLIP, pot reduce necesarul de date în regimuri cu puține exemple [143]. Învățarea semi-supervizată și prin transfer combină imagini normale, puține defecte etichetate și modele preantrenate [144]. Defectele reale pot servi la ajustarea pragurilor, la reglajul fin al modelului și la selecția exemplurilor negative dificile, cu condiția ca setul de test să rămână separat. Actualizarea incrementală pentru produse noi se verifică pe

toate variantele active, pentru a evita uitarea catastrofală a cunoștințelor anterioare. Colaborarea între fabrici poate folosi mecanismele de învățare federată descrise în capitolul 3, validând însă local fiecare produs, cameră, iluminare și configurație optică [105].

Inspecția tridimensională se justifică atunci când deformarea sau volumul nu pot fi evaluate prin intensitate și culoare. Se pot folosi profilarea laser, lumina structurată sau camere de adâncime. Datele tridimensionale presupun costuri mai mari, iar alegerea lor trebuie motivată de proprietatea inspectată [145]. În selecția unei metode, trebuie evaluat și costul computațional al inferenței pe platforma țintă: un model mai performant pe un set de referință poate depăși bugetul de timp disponibil într-o aplicație cu timp de ciclu redus, făcând preferabilă o abordare mai simplă, dar cu latență predictibilă.

6.4 Datele, validarea și evaluarea performanței

Seturile de referință permit compararea metodelor, dar nu echivalează cu acceptarea industrială. MVTec AD conține imagini normale și cu defecte, cu adnotări la nivel de pixel [144], [146]. Setul de date VisA oferă imagini industriale color, de rezoluție ridicată, acoperind o varietate mai largă de materiale și defecte fine [147]. MVTec 3D-AD include informație geometrică pentru evaluarea anomaliilor tridimensionale. Niciunul dintre aceste seturi nu reproduce automat materialul, camera, viteza liniei, iluminarea, murdăria, variantele de produs sau costul erorilor din instalația concretă [145], [146].

Principiile generale de separare a datelor și de prevenire a scurgerii de informații sunt prezentate în secțiunea 5.4. Pentru o stație AOI, împărțirea se face după unitatea fizică și configurația a cărei generalizare se verifică. Imaginile aceleiași piese, cadrele succesive din aceeași secvență sau imaginile aproape identice nu se distribuie între antrenare și testare.

Etichetele pot fi definite la nivel de imagine, de dreptunghi delimitator, de mască sau de categorie de defect. Termeni precum „zgârietură mică” sau „suprafață neuniformă” trebuie transformați în criterii măsurabile și corelați cu standardele de calitate. Așa cum s-a discutat în secțiunea 5.4, seturile industriale reale sunt puternic dezechilibrate, cu o proporție foarte mică de eșantioane defecte, iar evaluarea unor astfel de clase urmează principiile deja expuse. În inspecția optică, trebuie diferențiate explicit acceptarea falsă (un produs neconform este declarat conform) și respingerea falsă (un produs conform este eliminat sau trimis la reinspecție). Costurile celor două erori diferă și depind de produs: acceptarea falsă poate atrage reclamații, retrageri sau consecințe de siguranță; respingerea falsă mărește rebuturile, reinspecțiile și blocajele de flux.

Pentru detecție, se evaluează și precizia localizării; pentru segmentare, se pot folosi intersecția peste reuniune (IoU), coeficientul Dice și alte metrici la nivel de pixel. Metricile algoritmice sunt completate cu latența maximă, debitul, disponibilitatea și rata cazurilor trimise la verificare manuală. Rezultatele se analizează separat pe produse, loturi și regimuri.

Scorul modelului este transformat într-o decizie operațională prin pragurile validate conform secțiunii 5.4. Pentru o stație AOI, se poate defini o zonă intermediară: cazurile clare sunt acceptate sau respinse automat, iar cele ambigue sunt trimise la verificare manuală. Dimensiunea acestei zone trebuie corelată cu numărul real de inspectori și cu timpul disponibil; trimiterea unui volum excesiv de imagini la verificare manuală mută blocajul în alt punct. Stația trebuie să poată declara explicit stări precum „rezultat necunoscut”, „încredere redusă”, „imagine invalidă”, „verificare manuală” și „stație indisponibilă”, fără a forța fiecare imagine într-o clasă cunoscută. Pentru un scor de anomalie s , se poate folosi o zonă intermediară:

$$s < T_{\text{accept}} \Rightarrow \text{accept}$$

$$T_{\text{accept}} \leq s < T_{\text{reject}} \Rightarrow \text{review}$$

$$s \geq T_{\text{reject}} \Rightarrow \text{reject}$$

Scorul de încredere nu este automat o probabilitate calibrată; o valoare mare poate apărea și pentru o imagine din afara domeniului de antrenare. Interpretarea scorurilor și a pragurilor se verifică separat pentru produsele, tipurile de defect și condițiile optice relevante. Validarea periodică trebuie să verifice întregul lanț, nu doar modelul. Tabelul 6.2 detaliază elementele supuse verificării, metodele și rezultatele urmărite.

Tabelul 6.2. Validarea periodică a stației AOI

Element verificat	Metodă	Rezultat urmărit
Iluminare și optică	Ținte etalon pentru intensitate, uniformitate și claritate	Detectarea degradării LED-urilor, a murdăriei sau a defocalizării
Produse conforme	Eșantioane etalon (<i>Golden Samples</i>) și piese din producția curentă	Stabilitatea ratei de acceptare (>99.9%)
Defecte cunoscute	Mostre cu defecte reale la limita toleranței	Confirmarea detecției fără generarea de alarme false
Declanșarea și urmărirea piesei	Introducerea controlată a unor piese test pe flux	Verificarea sincronizării senzor–cameră–mecanism de sortare
Comunicația cu PLC-ul	Testarea semnalelor de handshake și a marcajelor temporale	Transmiterea sigură a deciziei în timpul alocat ciclului
Mecanismul de respingere	Confirmare fizică prin senzori de barieră optică pe linia de rebut	Garantarea eliminării fizice a fiecărei piese neconforme
Timpul de răspuns	Măsurarea bugetului temporal total (T_{total}) sub sarcină maximă	Încadrarea în timpul de ciclu al mașinii (<i>takt time</i>)
Versiuni și rețete	Verificarea hash-ului modelului AI și a parametrilor aprobați	Prevenirea utilizării unor rețete nevalidate sau modificate neautorizat

Eșantioanele etalon se protejează și se înlocuiesc când se degradează. O imagine injectată direct în software poate verifica modelul, dar nu verifică iluminarea, camera, declanșarea, comunicația și actuatorul. Frecvența validării depinde de criticitatea produsului pentru siguranță și calitate, de stabilitatea stației și de ritmul modificărilor.

6.5 Robustețea stației și administrarea schimbărilor

Robustețea unei stații AOI înseamnă menținerea performanței în domeniul de condiții pentru care a fost proiectată și validată [135]. Domeniul trebuie să precizeze variațiile admise de produs, material, culoare, textură, poziție, viteză, temperatură, lumină ambientală, iluminare controlată și configurație optică. Funcționarea în afara acestor limite se detectează și se tratează ca o condiție distinctă, nu se interpretează automat ca produs conform sau neconform.

Monitorizarea urmărește expunerea, uniformitatea iluminării, focalizarea, rata imaginilor invalide și distribuția scorurilor. Modificările de produs, furnizor sau material se corelează cu acești indicatori. Orice modificare a configurației optice sau mecanice este tratată ca o schimbare a stației și impune revalidarea. Mecanismele optice și geometrice care provoacă astfel de modificări sunt descrise în secțiunea 6.2. Orice modificare a iluminării, a opticii, a poziției mecanice, a camerei, a rețetei sau a variantei de produs este tratată ca o schimbare a configurației stației. Înaintea reluării producției, se aplică procedura de validare periodică și eșantioanele etalon definite în secțiunea 6.4 – inclusiv verificarea cazurilor conforme, a defectelor cunoscute și a situațiilor dificile.

Pentru o stație AOI, configurația operațională cuprinde modelul, preprocesarea, pragurile, rețeta, parametrii camerei, configurația iluminării și criteriile de acceptare. Aceste elemente se verifică împreună pentru fiecare produs activ. Principiile de distribuire a modelului la edge sunt expuse în secțiunea 3.8; validarea, monitorizarea derivei, versionarea și revenirea la o versiune anterioară sunt tratate în secțiunea 5.4.

Interfața de verificare trebuie să afișeze imaginea originală, regiunea de interes, rezultatul, rețeta activă și motivul pentru care acel caz necesită intervenția operatorului. Într-o stație AOI se pot defini trei niveluri de autoritate:

1. stația furnizează rezultatul și solicită verificarea manuală;
2. stația comandă respingerea automată a piesei identificate;
3. stația solicită oprirea controlată a liniei atunci când inspecția nu mai poate fi efectuată sau este detectată o condiție predefinită.

La ultimele două niveluri, PLC-ul trebuie să valideze identitatea piesei, termenul de valabilitate al rezultatului, modul de operare și interblocările înainte de a executa acțiunea. Principiile generale privind autoritatea automatizării, informațiile oferite operatorului și supravegherea umană sunt prezentate în secțiunea 8.5. Rețetele, imaginile, modelele și calea de scriere spre PLC trebuie protejate împotriva modificării sau utilizării neautorizate; schimbarea unui prag ori selectarea unei rețete greșite poate modifica decizia fără a schimba modelul. Cerințele generale privind identitățile, controlul accesului, actualizările, accesul la distanță și jurnalizarea sunt prezentate în secțiunea 8.3.

Pierderea camerei, iluminării sau comunicației trebuie să conducă la o stare explicită. O stație este disponibilă numai când imaginea este validă, rezultatul este furnizat la timp și acțiunea fizică poate fi executată. Comportamentul general în mod degradat este prezentat în secțiunea 8.2, iar decizia privind piesele aflate în flux, în secțiunea 6.6.

6.6 Integrarea cu PLC-ul, trasabilitatea și mecanismul de respingere

Integrarea cu PLC-ul se proiectează sub forma unui schimb de semnale și confirmări care separă declanșarea, identificarea produsului, rețeta și confirmarea preluării rezultatului. Mesajul poate conține decizia, codul defectului, scorul și un identificator unic al inspecției. În cadrul acestui schimb, confirmările trebuie să prevină atribuirea greșită a rezultatului, procesarea dublă și reutilizarea unei predicții învechite; rezultatul trebuie să aibă o perioadă de valabilitate, iar depășirea duratei maxime de așteptare nu se tratează implicit ca o conformitate.

Mecanismul de respingere trebuie inclus în diagnosticarea stației. Pentru aplicațiile importante, acțiunea fizică se confirmă cu un senzor în zona de respingere. Diferența dintre „comandă emisă” și „piesă eliminată” se păstrează în istoricul inspecției.

Trasabilitatea permite reconstituirea fiecărei decizii. Pentru fiecare inspecție se rețin: identificatorul produsului, marcajul temporal, imaginea relevantă, rețeta, versiunile de software și model, scorul, decizia transmisă și confirmarea acțiunii fizice. Nu este obligatorie arhivarea permanentă a tuturor imaginilor; politica poate prevedea păstrarea cazurilor neconforme și a unui eșantion de produse conforme.

Criteriul decisiv al acceptării industriale este ca decizia corectă să fie asociată piesei corecte, transmisă înainte de expirarea termenului și aplicată fizic, cu confirmarea rezultatului.

7. Identificarea sistemelor, control predictiv și gemeni digitali

Modelele industriale sunt utilizate pentru explicarea, estimarea, simularea, diagnosticarea și controlul proceselor fizice. Un model poate descrie relația dintre puterea unui încălzitor și temperatura produsului, dintre poziția unei valve și debit sau dintre programul de producție și consumul energetic. Valoarea sa nu este dată exclusiv de precizia numerică, ci de capacitatea de a reprezenta fenomenele relevante pentru scopul urmărit. Un model potrivit pentru afișarea unei tendințe poate fi insuficient pentru simulare; un model care prezice corect următorul eșantion poate acumula erori importante atunci când își utilizează propriile ieșiri.

Un model folosit doar pentru afișarea unei tendințe poate tolera mici erori. Într-un controler, aceeași eroare poate duce la o comandă nepotrivită: algoritmul caută tocmai valorile la care modelul promite cel mai bun rezultat, chiar dacă pentru acele valori avem puține măsurări. De aceea, înainte de a folosi modelul pentru optimizare, verificăm în ce regimuri a fost testat, cât de incerte sunt predicțiile și ce strategie de rezervă folosim dacă rezultatul nu mai este de încredere [118], [148].

Lațul ingineresc începe prin definirea obiectivului și înțelegerea procesului fizic, continuă cu proiectarea experimentului, achiziția datelor, alegerea structurii, estimarea parametrilor și validarea pe date independente și se extinde apoi către simulare, control predictiv sau geamăn digital. Un model identificat poate fi utilizat într-un simulator, simulatorul poate susține punerea în funcțiune virtuală, iar aceeași reprezentare poate deveni o componentă a unui controler predictiv ori a unui geamăn digital.

7.1 Scopul modelului, proiectarea experimentului și domeniul de validitate

Identificarea sistemelor urmărește obținerea unui model matematic pornind de la măsurări ale intrărilor și ieșirilor. Deși utilizează date, procesul nu poate fi separat de cunoașterea instalației. Înaintea colectării datelor, trebuie precizat dacă modelul este destinat predicției, controlului, simulării, diagnosticării sau construirii unui geamăn digital. Scopul determină variabilele necesare, orizontul de predicție, precizia solicitată, domeniul de funcționare și nivelul de interpretabilitate [118].

Un model pentru detectarea abaterilor poate fi simplu și sensibil la modificări. Un model pentru simulare pe termen lung trebuie să rămână stabil când folosește propriile ieșiri, iar un model pentru control predictiv (MPC - *Model Predictive Control*) trebuie să descrie dinamica pe întregul orizont de predicție. Dacă modelul este utilizat de un algoritm de optimizare, trebuie impuse limitele fizice relevante; altfel, algoritmul poate selecta un punct de operare imposibil sau nesigur.

Formularea diferențiază intrările manipulate, perturbațiile măsurate și nemăsurate, ieșirile observate și stările interne. Într-un cuptor industrial, puterea încălzitorului constituie intrarea manipulată, temperatura produsului ieșirea urmărită, iar debitul de aer o perturbație măsurată. Omiterea unei perturbații importante poate face ca modelul să îi atribuie efectul intrării sau unui parametru incorect. Includerea tuturor variabilelor disponibile nu se justifică automat, deoarece măsurările slab relevante pot reduce identificabilitatea parametrilor.

Procesul de identificare a sistemului presupune stimularea modurilor dinamice de interes. Aceasta poate utiliza trepte, impulsuri, semnale multisinus sau secvențe binare pseudoaleatoare. Atunci când experimentele dedicate nu sunt acceptabile, pot fi folosite variațiile naturale ale producției, cu condiția ca acestea să conțină suficientă informație relevantă. Experimentul trebuie să respecte limitele de siguranță și costul perturbării producției [118]. Semnalul de excitație trebuie, de asemenea, să acopere frecvențele și regimurile importante: datele colectate într-un singur punct de operare nu descriu automat comportamentul neliniar al întregii instalații, iar lipsa unei excitații suficiente poate produce parametri cu incertitudine ridicată și modele care eșuează în condiții diferite.

Datele industriale sunt colectate frecvent în buclă închisă, unde regulatorul modifică intrarea ca răspuns la ieșire. Comanda devine corelată cu perturbațiile, iar metodele de identificare trebuie să țină seama de această relație. Un regulator performant poate reduce variația ieșirii atât de mult încât datele să nu mai conțină suficientă informație despre dinamica procesului.

Perioada de eșantionare trebuie aleasă pentru a reprezenta dinamica relevantă, fără ca datele să devină dominate de zgomot. Marcajele temporale incorecte și canalele nesincronizate pot fi interpretate eronat drept dinamică a procesului. Pentru identificare, este esențial ca aceleași transformări să fie utilizate atât la estimare, cât și în exploatare [115], [118].

Domeniul de validitate include intervalele intrărilor și ieșirilor, regimurile de operare și condițiile ambientale. Un model identificat în jurul unui punct de funcționare este adesea valid numai local. Extrapolarea nu trebuie permisă doar pentru că software-ul acceptă numeric intrările; depășirea domeniului trebuie detectată și tratată prin limitare sau activarea unei strategii de rezervă.

7.2 Structura, estimarea și validarea modelului

Modelele matematice utilizate în informatica industrială se clasifică în funcție de ponderea cunoștințelor fizice a priori introduse în raport cu datele experimentale disponibile. Se disting trei mari categorii: modele de tip cutie albă (*white-box*, bazate exclusiv pe legi fizice), modele de tip cutie neagră (*black-box*, învățate integral din date) și modele de tip cutie gri (*grey-box*, care combină o structură fizică parțială cu identificarea parametrilor necunoscuți). Tabelul 7.1 compară aceste abordări.

Tabelul 7.1. Comparația principalelor categorii de modele

Caracteristică	Modele de tip cutie albă	Modele de tip cutie gri	Modele de tip cutie neagră
Cunoaștere fizică introdusă	Ridicată (ecuații fundamentale de bilanț)	Parțială (structură fizică asumată)	Redusă (relații pur statistice)
Necesar de date	Redus–mediu (pentru validare)	Mediu (pentru estimarea parametrilor)	Ridicat (acoperire completă de regimuri)
Interpretabilitate	Ridicată (coeficienții au sens fizic direct)	Ridicată–medie	Redusă–medie (rețele opace)
Extrapolare	Justificată în limitele legilor fizice	Bună în domenii controlate	Slabă în afara datelor antrenate
Introducerea constrângerilor	Naturală (limite fizice directe)	Relativ directă	Necesită penalizări matematice suplimentare
Costul modelării	Ridicat (analiză fizică detaliată)	Mediu	Redus la început, dependent de date
Risc caracteristic	Omiterea unor fenomene fizice secundare	Structură asumată incompletă	Instabilitate fizică la intrări neobservate

Un model clasic de tip cutie albă este reprezentat, de exemplu, de ecuația de bilanț a unui proces termic simplificat:

$$C \frac{dT}{dt} = P - k(T - T_a)$$

unde C este capacitatea termică a incintei, P este puterea electrică aplicată de rezistențe, k este coeficientul global de transfer termic spre exterior, iar T_a reprezintă temperatura mediului ambiant. Deși o astfel de structură este transparentă și permite impunerea directă a constrângerilor fizice, parametrii reali pot varia în timp (de exemplu, acumularea depunerilor sau îmbătrânirea izolației), iar fenomenele neliniare devin greu de formulat analitic.

În practică, modelele de tip cutie gri oferă cel mai echilibrat compromis pentru aplicațiile industriale [149]: structura de bază păstrează ecuațiile de conservare a masei și energiei, în timp ce coeficienții necunoscuți (frecări, randamente, rezistențe termice) sunt estimați pe baza datelor de proces.

Structura modelului trebuie să fie suficient de complexă pentru dinamica relevantă și suficient de simplă pentru o estimare robustă. O ordine prea mică produce erori sistematice; o ordine prea mare poate reproduce zgomotul și reduce generalizarea [117], [118].

Pentru compensarea statică a traductoarelor sau pentru construirea unor senzori virtuali (*soft sensors*), se utilizează frecvent regresia liniară multivariabilă regularizată. Tehnica Ridge (regularizare L_2) stabilizează estimarea în prezența variabilelor puternic corelate, în timp ce metoda Lasso (regularizare L_1) penalizează coeficienții nesemnificativi, forțându-i la zero și realizând astfel o selecție automată a celor mai relevanți parametri [150].

Atunci când procesul prezintă inerție, întârzieri de transport și dinamică temporală, descrierea comportamentului impune utilizarea unor modele discrete intrare-ieșire.

Modelul autoregresiv cu intrare exogenă (ARX) reprezintă structura fundamentală utilizată în identificarea sistemelor de control:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + e(k)$$

În această formulare:

- $y(k)$ reprezintă mărimea măsurată de ieșire la pasul curent de eșantionare k ;
- $u(k)$ este comanda aplicată la intrare;
- $e(k)$ este eroarea de predicție (reziduul) la pasul k , care înglobează perturbațiile și zgomotul de măsurare nemodelat;
- Simbolul q^{-1} reprezintă operatorul de întârziere discretă cu un eșantion ($q^{-1}y(k) = y(k - 1)$ și $q^{-2}y(k) = y(k - 2)$);
- Polinoamele dinamice au forma $A(q^{-1}) = 1 + a_1q^{-1} + \dots + a_{n_a}q^{-n_a}$ și $B(q^{-1}) = b_1q^{-1} + \dots + b_{n_b}q^{-n_b}$, unde ordinele n_a și n_b stabilesc complexitatea dinamicii.

Dacă perturbațiile care afectează procesul nu sunt pur aleatoare, ci prezintă o dinamică corelată în timp (zgomot colorat), structura se extinde la modelul **ARMAX** (*Autoregressive Moving Average with Exogenous Inputs*):

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k)$$

unde polinomul $C(q^{-1})$ modelează media mobilă a perturbațiilor [117], [118]. Estimarea coeficienților se realizează numeric prin metoda celor mai mici pătrate (*Least Squares*) sau algoritmi recursivi (RLS) pentru adaptare online.

Validarea unui model nu se realizează niciodată pe setul de date folosit la antrenare, ci pe date experimentale independente. Reziduul de predicție reprezintă diferența dintre ieșirea reală măsurată și cea estimată de model:

$$e(k) = y(k) - \hat{y}(k)$$

Dacă polinoamele $A(q^{-1})$ și $B(q^{-1})$ au extras cu succes toată dinamica utilă a procesului, eroarea de predicție rămasă $e(k)$ trebuie să aibă proprietățile unui zgomot alb (să fie pur aleatoare, cu medie nulă și fără corelație temporală).

În practica inginerescă, acest lucru este confirmat prin două teste statistice standard:

1. Autocorelația reziduurilor: funcția de autocorelație a lui $e(k)$ trebuie să tindă spre zero pentru orice decalaj de timp $\tau > 0$ (demonstrând că eroarea nu conține un tipar repetitiv ignorat de model);
2. Corelația încrucișată între intrări și reziduu: lipsa corelației dintre comenzile anterioare $u(k - \tau)$ și $e(k)$ confirmă că modelul a surprins complet influența intrărilor asupra ieșirii [118], [150].

Pentru ilustrare, se consideră un cuptor electric de laborator, cu o putere maximă de 2 kW și o constantă de timp termică de ordinul minutelor. Se aplică la intrare o secvență binară pseudoaleatoare (PRBS – *Pseudo-Random Binary Sequence*) care comută puterea între 0% și 40% din maxim, cu o perioadă de eșantionare de 10 secunde. Temperatura internă este măsurată cu o termocuplă. După eliminarea primelor 20 de eșantioane (tranzitoriu de pornire), setul de date este împărțit: 400 de eșantioane pentru estimare și 200 pentru validare.

Se estimează un model ARX de ordinul 2:

$$A(q^{-1}) y(k) = B(q^{-1}) u(k) + e(k)$$

$$A(q^{-1}) = 1 - 1,52 q^{-1} + 0,54 q^{-2}$$

$$B(q^{-1}) = 0,018 q^{-1} + 0,014 q^{-2}$$

Testarea pe setul independent de validare confirmă o dinamică excelentă: reziduul are medie aproximativ nulă, iar funcția de autocorelație se încadrează în intervalul de încredere statistic de 95%. Modelul captează cu fidelitate constanta de timp dominantă și poate fi utilizat direct pentru acordarea unui regulator PID (*Proportional-Integral-Derivative*) avansat sau ca bază de predicție într-un controler MPC, cu mențiunea că domeniul său de validitate garantat este regimul 0–40% din puterea termică.

7.3 Simularea și validarea graduală

Simularea reprezintă evoluția unui sistem prin rezolvarea numerică a unui model [151]. Ea poate susține proiectarea controlului, analiza scenariilor, instruirea operatorilor și punerea în funcțiune virtuală. Modelul trebuie să fie suficient de fidel pentru întrebarea analizată, nu maximal de detaliat. Dacă obiectivul este verificarea unei secvențe PLC, pot fi suficiente stările discrete, semnalele I/O și întârzierile relevante. Pentru reglarea temperaturii, trebuie reprezentate dinamica termică și perturbațiile principale; pentru analiza coliziunilor, sunt necesare geometria și cinematica. O animație realistă nu dovedește existența unui model fizic precis [151].

În teoria modernă a controlului și în mediile de simulare, dinamica sistemului este formalizată prin reprezentarea în spațiul stărilor. În timp continuu, modelul este descris de ecuațiile $\dot{x} = f(x, u, t)$ și $y = g(x, u, t)$, iar în timp discret:

$$x_{k+1} = f_d(x_k, u_k), y_k = g_d(x_k, u_k)$$

unde x_k reprezintă vectorul stărilor interne, u_k vectorul comenzilor de intrare, iar y_k ieșirile măsurabile. Sistemele industriale reale sunt predominant hibride, combinând dinamica fizică continuă cu tranziții discrete declanșate de logica secvențială din PLC. Metoda de integrare influențează precizia și stabilitatea numerică. Pentru simularea în timp real HIL (*Hardware-in-the-Loop*), este necesar un pas fix și un timp de execuție mai mic decât intervalul simulat.

Condițiile inițiale trebuie să corespundă stării reale. În punerea în funcțiune virtuală, starea simulatorului se sincronizează cu logica PLC-ului. Repornirea trebuie să trateze stările persistente și produsele aflate în proces.

Validarea graduală reduce riscul, trecând de la verificarea conceptuală la testarea implementării. Tabelul 7.2 descrie nivelurile de validare recomandate.

Tabelul 7.2. Niveluri de validare bazate pe model

Nivel	Configurație	Obiectiv principal
Testare cu modelul în buclă — MIL	Modelul procesului și algoritmul sunt simulați software	Verificarea conceptului de control, a structurii și a scenariilor
Testare cu software-ul în buclă — SIL	Codul C/C++ compilat rulează conectat la model	Identificarea erorilor de logică software și a preciziei numerice
Testare cu procesorul în buclă — PIL	Algoritmul rulează pe procesorul țintă, procesul simulat	Verificarea timpului de execuție, a memoriei și a tipului de date
Testare cu echipamentul fizic în buclă — HIL	Controlerul fizic (PLC) conectat la simulator I/O timp real	Verificarea semnalelor I/O electrice, latențelor și a reacției la defect
Punere în funcțiune virtuală	Modelul 3D conectat cu PLC-ul, HMI-ul și comunicațiile	Testarea secvențelor complete înaintea disponibilității instalației fizice

Nivelurile sunt complementare și nu se substituie reciproc. Un test HIL poate scoate la iveală probleme de comportament temporal și de intrare–ieșire, dar nu înlocuiește verificarea inițială a algoritmului sau validarea ulterioară pe instalația reală. Punerea în funcțiune virtuală poate identifica erori ale secvențelor, interblocărilor, alarmelor și interfețelor înainte de finalizarea echipamentului fizic, însă rezultatele depind de fenomenele incluse în simulator. Fiecare rezultat trebuie asociat cu scopul, ipotezele, versiunea modelului, parametrii, algoritmul numeric de rezolvare, domeniul de utilizare și incertitudinea.

7.4 Controlul predictiv bazat pe model

Controlul predictiv bazat pe model (*MPC-Model Predictive Control*) utilizează un model pentru a anticipa evoluția sistemului pe un orizont finit și rezolvă repetat o problemă de optimizare [152]. La fiecare moment, se măsoară sau estimează starea, se calculează o secvență de comenzi, se aplică doar prima comandă, iar problema se reia după actualizarea măsurărilor. Acest principiu al orizontului glisant introduce reacția și permite compensarea perturbațiilor. [148], [152].

La fiecare pas de calcul k , algoritmul MPC determină secvența optimă de comenzi viitoare prin minimizarea unei funcții de cost pătratic pe un orizont temporal definit:

$$J = \sum_{i=1}^{N_p} (y_{k+i|k} - r_{k+i})^T Q (y_{k+i|k} - r_{k+i}) + \sum_{i=0}^{N_c-1} \Delta u_{k+i|k}^T R \Delta u_{k+i|k}$$

Semnificația celor doi termeni reflectă direct cerințele practice de control:

- Termenul de performanță (urmărirea referinței): penalizează abaterea ieșirilor estimate $y_{k+i|k}$ față de traiectoria dorită r_{k+i} pe durata orizontului de predicție (N_p);
- Termenul de efort (variația comenzii): penalizează manevrele bruște ale comenzilor $\Delta u_{k+i|k} = u_{k+i} - u_{k+i-1}$ pe durata orizontului de control ($N_c \leq N_p$), protejând actuatorii împotriva uzurii rapide;
- Matricile de ponderare Q și R : permit ajustarea compromisului dorit între viteza de răspuns a buclei (valori mari în Q) și consumul de energie sau agresivitatea comenzii (valori mari în R).

Costul poate include și consumul energetic, uzura, emisiile, variațiile rapide ori penalizarea depășirii unor limite flexibile.

Avantajul principal al MPC este tratarea explicită a constrângerilor [153]:

$$\begin{aligned} u_{\min} &\leq u_k \leq u_{\max} \\ \Delta u_{\min} &\leq \Delta u_k \leq \Delta u_{\max} \\ x_{\min} &\leq x_k \leq x_{\max} \\ y_{\min} &\leq y_k \leq y_{\max} \end{aligned}$$

Aceste limite pot reprezenta temperatura și presiunea maximă, puterea disponibilă sau toleranțele de calitate. Constrângerile dure nu trebuie încălcate; cele flexibile pot fi relaxate controlat prin variabile penalizate. O limită de siguranță nu trebuie transformată într-o constrângere flexibilă pentru a menține fezabilitatea optimizării [148], [153].

MPC este util pentru procese multivariabile, cuplate, cu întârzieri și constrângeri active [154]. Pentru bucle simple, un PID poate fi mai ușor de implementat și menținut. În multe instalații, MPC calculează referințele pentru regulatoarele PID locale, combinând optimizarea multivariabilă cu robustețea buclelor de bază.

Controlerul poate utiliza observatori sau filtre Kalman pentru estimarea stărilor nemăsurate. Perturbațiile măsurate pot fi introduse în predicție, iar eroarea sistematică poate fi estimată pentru eliminarea erorii staționare.

Orizontul trebuie corelat cu întârzierile, constantele de timp, obiectivul economic și perioada de eșantionare. Un orizont prea scurt ignoră efectele întârziate; unul prea lung mărește costul de calcul fără a aduce informație suplimentară semnificativă. Modificarea perioadei de eșantionare fără recalcularea orizontului alterează semnificația controlerului.

MPC neliniar utilizează modele de forma:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= g(x_k, u_k) \end{aligned}$$

Problema de optimizare poate prezenta soluții locale și timp de calcul variabil. Inițializarea cu soluția anterioară, reducerea ordinului și generarea automată de cod pot scurta timpul de rezolvare; limita maximă trebuie măsurată sub încărcare reprezentativă. Dacă rezultatul nu este disponibil înaintea termenului-limită, controlerul trebuie să adopte un comportament predefinit [2], [148]. În această situație, strategia de rezervă poate recurge la un regulator PID, la un controler local sau la o comandă fezabilă anterioară. Păstrarea ultimei comenzi nu este automat sigură, deoarece procesul poate continua să evolueze, iar sistemul trebuie să detecteze pierderea fezabilității și să comunice această stare operatorului.

MPC adaptiv actualizează parametrii pentru a compensa uzura și schimbarea punctului de operare. Adaptarea trebuie limitată și validată: datele de calitate slabă pot conduce la estimări instabile. În unele aplicații, utilizarea mai multor modele locale este mai sigură decât adaptarea continuă.

MPC bazat pe învățare automată poate folosi modele învățate pentru dinamica procesului, reziduuri, perturbații, funcția de cost sau aproximarea algoritmului de optimizare. O formulare hibridă poate fi scrisă astfel:

$$f_{\text{total}}(x, u) = f_{\text{physics}}(x, u) + f_{\text{learned}}(x, u)$$

Componenta fizică păstrează structura și constrângerile de bază; modelul învățat compensează efectele greu de reprezentat. Riscul este ca eroarea maximă să apară tocmai în regiunile rare selectate de optimizare. Modelele probabilistice și estimarea incertitudinii pot fi utilizate pentru a impune constrângeri conservatoare și pentru a activa strategia de rezervă. Protecția procesului trebuie să rămână independentă de componenta învățată [32], [155].

Modelul utilizat în control se monitorizează prin reziduuri, eroarea de predicție, deriva parametrilor și frecvența de activare a strategiei de rezervă. O creștere a acestor activări poate semnaliza degradarea modelului sau funcționarea în afara domeniului validat. Recalibrarea și reentrenarea sunt modificări ingineresti și trebuie validate înainte de activare, nu aplicate automat în bucla de control.

7.5 Gemeni digitali și firul digital

Un geamăn digital este o reprezentare digitală a unei entități fizice, sincronizată cu informații selectate și utilizată pentru un scop ingineresc definit [156], [157]. Aplicațiile variază de la planificarea optimă a traiectoriilor [158] până la optimizarea proceselor de fabricație. El poate include geometrie, configurație, modele fizice, logică de control, parametri și istoric. Un simplu tablou de bord nu este un geamăn digital; este nevoie de o relație persistentă cu activul și de un serviciu ingineresc [156], [157]. Cadrul ISO 23247 oferă o arhitectură de referință pentru gemenii digitali în producție, definind entitățile, interfețele și protocoalele necesare pentru compunerea și interogarea lor [157].

Menținerea unui geamăn digital sincronizat cu activul fizic ridică provocări practice importante. Sincronizarea nu înseamnă doar actualizarea periodică a valorilor, ci și coerența temporală între surse multiple. Dacă un senzor de temperatură eșantionează la 1 Hz și unul de presiune la 100 Hz, iar geamănul le combină într-un model termodinamic, decalajul temporal dintre observații poate produce estimări interne inconsistente. În plus, întârzierile de comunicație variabile între edge și cloud pot face ca starea afișată să fie deja depășită. Arhitectura geamănului trebuie să includă politici explicite de imputare, marcaje temporale și indicatori de prospețime, iar atunci când sincronizarea se pierde dincolo de o limită acceptabilă, geamănul trebuie să semnalizeze această stare și să refuze decizii bazate pe date nesigure.

Relația dintre activ și reprezentare poate îmbrăca mai multe forme (Tabelul 7.3). Autoritatea se stabilește separat de complexitatea modelului [159].

Tabelul 7.3. Model digital, umbră digitală și geamăn digital

Reprezentare	Fluxul dominant al informației	Caracteristică
Model digital	Actualizare predominant manuală	Reprezentarea este modificată exclusiv prin intervenții umane
Umbră digitală	Automat dinspre activ către model	Starea fizică actualizează automat reprezentarea digitală
Geamăn digital	Automat și controlat în ambele direcții	Reprezentarea digitală poate optimiza și comanda activul în limite sigure

Fidelitatea se alege în funcție de utilizare [160], [161]: pentru punerea în funcțiune virtuală contează secvențele și comportamentul temporal, iar pentru analiza energetică, fluxurile și stările de producție. Creșterea fidelității sporește costul de calcul și trebuie justificată prin decizia susținută. Sincronizarea poate fi periodică sau declanșată de evenimente; după pierderea sincronizării, reprezentarea nu trebuie prezentată ca actuală, iar conflictele între sursele de date se rezolvă prin atribuirea explicită a responsabilităților. Totodată, validarea se realizează pentru fiecare utilizare a geamănului. Un model acceptabil pentru instruire poate fi insuficient pentru optimizare, iar geamănul trebuie actualizat după modificarea configurației fizice; o reprezentare depășită poate fi mai periculoasă decât absența ei.

Firul digital (*digital thread*) păstrează relațiile dintre cerințe, proiectare, simulare, operare și mentenanță. Un rezultat de producție trebuie asociat cu versiunile programului, modelului și rețelei utilizate. Autoritatea geamănului poate fi informativă, diagnostică sau prescriptivă; un rezultat al simulării nu trebuie transferat automat către activ fără verificarea protecțiilor locale. Geamănul poate furniza o valoare de referință sau o recomandare către MPC, însă controlerul local trebuie să valideze comanda în raport cu starea curentă și cu interblocările.

7.6 Integrarea identificării, simulării, controlului și geamănului digital

Relația dintre aceste domenii este sintetizată în Figura 7.1 [25]. Fluxul nu presupune un singur model monolitic. Datele și cunoașterea procesului alimentează identificarea; validarea stabilește domeniul de utilizare; simularea verifică scenariul; MPC utilizează modelul în limitele aprobate; geamănul digital menține legătura cu activul.

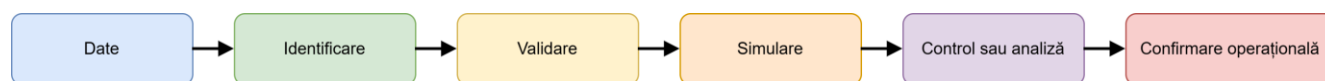


Figura 7.1. Fluxul integrat de la date la confirmarea operațională a modelului

Pot coexista modele distincte pentru control, energie, mentenanță și logistică, fiecare cu propriul scop și criterii de acceptare. Modificarea instalației sau a datelor impune reverificarea reprezentărilor afectate; confirmarea rezultatului în exploatare închide bucla și oferă temeiul pentru recalibrare sau retragere.

Partea a IV-a. Siguranță, securitate și operare centrată pe om

8. Siguranța în funcționare, securitate și operare industrială centrată pe om

Fiabilitatea, disponibilitatea, siguranța, securitatea și interacțiunea om–sistem trebuie proiectate împreună, ca proprietăți ale aceluiași sistem cyber-fizic. O instalație poate fi disponibilă dar nesigură, protejată fizic dar vulnerabilă cibernetic, sau poate deține mecanisme puternice de securitate dar să nu poată fi restaurată după o actualizare eșuată. Similar, o interfață poate afișa valori corecte și totuși să împiedice operatorul să înțeleagă situația sau să intervină la timp.

În sistemele industriale conectate, evenimentele fizice și digitale se pot amplifica reciproc: defectarea unui senzor conduce la o comandă nepotrivită, întreruperea rețelei cauzează pierderea supravegherii, iar compromiterea unei stații de configurare poate modifica logica PLC-ului. O alarmă neclară sau o interfață supraîncărcată poate întârzia identificarea evenimentului declanșator și transforma o abatere locală într-o pierdere a serviciului la nivelul întregului proces [28], [162].

Reziliența se obține prin prevenire, detecție, izolare, tolerare a defectelor, funcționare degradată, recuperare și revenire la o stare de încredere. Redundanța nu compensează automat defectele comune; copiile de siguranță nu ajută dacă restaurarea nu a fost testată. Operatorul trebuie privit ca parte a arhitecturii, nu ca resursă de rezervă chemată doar după eșecul automatizării.

8.1 Siguranța în funcționare și comportamentul la defect

Siguranța în funcționare exprimă capacitatea unui sistem de a furniza un serviciu justificat de încredere [28]. Vocabularul de bază distinge între *defect* (cauza potențială a unei abateri, de natură fizică, software, umană sau de mediu), *eroare* (starea internă incorectă produsă de defect) și *defectare* (momentul când serviciul nu mai respectă cerința). Defectul poate rămâne latent, poate fi detectat înainte de propagare sau poate traversa mai multe componente până când consecința devine observabilă [28].

Fiabilitatea reprezintă probabilitatea de a furniza serviciul corect într-un interval și în condiții specificate. Pentru sistemele reparabile, disponibilitatea este adesea mai relevantă operațional și poate fi aproximată prin:

$$A = \frac{MTBF}{MTBF + MTTR}$$

unde *MTBF* (*Mean Time Between Failures*) este timpul mediu între defectări, iar *MTTR* (*Mean Time To Repair*) este timpul mediu de restaurare. Relația arată că disponibilitatea poate fi crescută atât prin reducerea frecvenței defectărilor, cât și prin scurtarea recuperării. O arhitectură redundantă poate avea o disponibilitate reală scăzută dacă diagnosticarea este dificilă sau piesele de schimb lipsesc. De aceea, mentenabilitatea — care depinde de modularitate, documentație, instrumente de diagnostic și competențele personalului — este la fel de importantă ca fiabilitatea intrinsecă. Un sistem care nu își expune starea internă și cauza blocării poate necesita ore pentru localizarea unui simplu defect. Diagnosticarea, alarmele și interfața operatorului sunt componente ale siguranței în funcționare, nu facilități auxiliare.

Siguranța față de pericole (safety) nu este echivalentă cu fiabilitatea. Un sistem poate ceda frecvent dar întotdeauna într-o stare sigură; un sistem foarte fiabil poate produce o consecință severă printr-o

singură defectare. Pentru sistemele electrice, electronice și electronice programabile, IEC 61508 tratează siguranța funcțională pe baza unui ciclu de viață complet, clasificând funcțiile de securitate prin niveluri de integritate a siguranței [53]. În domeniul mașinilor industriale, standardul armonizat ISO 13849-1 evaluează părțile de comandă referitoare la securitate prin nivelul de performanță, determinat pe baza a patru factori fundamentali: structura arhitecturală, timpul mediu până la o defectare periculoasă ($MTTF_D$), acoperirea de diagnosticare (DC_{avg}) și rezistența la defecte de cauză comună (CCF) [163].

Acest cadru normativ este consolidat la nivel european prin Regulamentul (UE) 2023/1230 privind mașinile (care înlocuiește directiva 2006/42/CE) [164]. Noul regulament impune cerințe esențiale stricte privind riscurile emergente: protecția împotriva coruperii software-ului de control (interconectarea obligatorie a cerințelor de safety cu cele de securitate cibernetică), comportamentul mașinilor dotate cu logică adaptivă sau de autoînvățare (AI) și integritatea canalelor de comandă fără fir utilizate pentru funcțiile de oprire de urgență.

Securitatea cibernetică protejează sistemul împotriva acțiunilor neautorizate și urmărește confidențialitatea, integritatea și disponibilitatea. În mediile OT, se adaugă obligația de a proteja logica de control, secvențele, configurațiile, marcasele temporale și autoritatea de comandă. O comandă poate fi autenticată și corect formatată și, cu toate acestea, să fie incompatibilă cu starea fizică a procesului. Autentificarea trebuie completată de validarea locală a domeniului, a modului de operare, a interblocărilor și a plauzibilității fizice.

Siguranța față de pericole și securitatea cibernetică au obiective diferite, dar trebuie coordonate. Un atac poate declanșa același pericol ca un defect accidental; o măsură de securitate configurată greșit poate bloca o alarmă. Criptarea poate mări latența, redundanța poate extinde suprafața de atac, iar trecerea automată într-o stare sigură poate reduce disponibilitatea. Un patch de securitate poate elimina o vulnerabilitate și poate introduce un nou defect. Asemenea compromisuri trebuie analizate prin scenarii și cerințe explicite, nu rezolvate prin prioritizarea automată a unui singur atribut.

Starea sigură este acea condiție în care riscul rezidual devine acceptabil după detectarea unei pierderi sau a unei situații periculoase. Ea poate însemna oprirea mișcării, întreruperea alimentării, izolarea unei surse sau menținerea temporară a unei funcții. Trecerea în stare sigură nu este întotdeauna sinonimă cu „oprit”: închiderea unei valve de combustibil poate fi sigură, dar oprirea imediată a răcirii unui reactor poate spori pericolul. Starea corectă depinde de proces, de energia acumulată, de modul de operare și de etapa ciclului. În proiectare, circuitele de oprire folosesc frecvent contacte normal închise și logică negativă, astfel încât pierderea alimentării sau ruperea unui fir să genereze automat o cerere de oprire. Principiul ajută la detectarea unor defecte care altminteri ar rămâne ascunse, însă nu acoperă automat contactele sudate sau defectele comune. Proiectarea trebuie să ia în calcul modurile reale de defectare ale întregului lanț, nu doar starea logică ideală.

În anumite aplicații, funcționarea trebuie menținută temporar după apariția unui defect, ceea ce presupune detectarea și izolarea defectului, resurse alternative, transferul coerent al stării și verificarea capacității rămase. Numeroase instalații combină menținerea pe termen scurt a funcției cu trecerea controlată într-o stare sigură [29], [165].

Starea sigură se definește separat pentru modurile automat, manual, mentenanță, pornire, oprire, local, la distanță și degradat. Modul activ și autoritatea asociată trebuie să rămână vizibile operatorului. Întreruperea alimentării circuitului de comandă nu suprimă automat energia reziduală; interfața trebuie să deosebească între comanda de oprire emisă și confirmarea că echipamentul a atins efectiv o stare fără pericole.

8.2 Redundanță, funcționare degradată și recuperare

Redundanța introduce resurse suplimentare pentru detectarea, mascarea sau recuperarea după defecte și poate fi fizică, software, temporală, informațională sau analitică. Redundanța modulară dublă (DMR) detectează neconcordanța; redundanța modulară triplă (TMR) permite votul majoritar, dar nu apără împotriva defectelor comune sau a specificațiilor greșite implementate identic [29].

Independența trebuie analizată la nivel fizic, energetic, software și organizațional. Diversitatea – tehnologii, algoritmi sau echipe diferite – reduce riscul defectelor comune dar introduce diferențe de comportament care trebuie testate. Redundanța temporală repetă o operație pentru a detecta defecte tranzitorii dar mărește latența. Redundanța informațională (paritate, CRC, coduri corectoare) detectează alterarea datelor dar nu protejează împotriva retransmisiei unui mesaj valid.

Redundanța temporală repetă o operație pentru a detecta un defect tranzitoriu, dar mărește latența și devine inutilă după depășirea termenului-limită. Redundanța informațională – biți de paritate, CRC, coduri corectoare – detectează sau corectează alterarea datelor. CRC-ul (*Cyclic Redundancy Check*) nu protejează însă împotriva retransmisiei unui mesaj valid, a unei comenzi malițioase sau a unei valori greșite generate corect de aplicație.

Redundanța analitică elimină necesitatea dublării senzorilor fizici, comparând semnalele măsurate cu valorile estimate pe baza unui model fizic al procesului (de exemplu, verificarea conservării masei prin compararea variației nivelului dintr-un rezervor cu diferența dintre debitele de intrare și ieșire). O discrepantă persistentă între măsurare și ecuația de bilanț semnalează prompt defectarea unui traductor, o scurgere pe conductă sau un fenomen parazit neobservat [29], [58]. Acoperirea diagnosticării exprimă proporția defectelor periculoase detectate; testarea periodică a funcției de protecție trebuie să verifice întregul lanț: senzor, logică, comunicație, actuator și confirmarea efectului.

În sistemele încorporate, mecanismul de tip *watchdog* poate detecta absența progresului software, dar o sarcină blocată poate continua să execute instrucțiunea de rearmare. Variantele robuste verifică fereastra temporală, ordinea etapelor și progresul funcțional. Un semnal periodic de prezență (*heartbeat*) confirmă că o componentă transmite mesaje, nu că algoritmul produce un rezultat corect. Monitorizarea trebuie să includă starea funcțională, validitatea intrărilor, timpul de execuție și confirmarea efectului.

Verificările de domeniu controlează limitele numerice; cele de plauzibilitate controlează relațiile fizice și temporale. În sistemele cu termene-limită, relația dintre detecție și recuperare trebuie să satisfacă:

$$T_{\text{detect}} + T_{\text{isolate}} + T_{\text{recover}} \leq T_{\text{allowed}}$$

Dacă recuperarea durează mai mult decât intervalul în care procesul poate funcționa fără acea funcție, arhitectura trebuie să mențină serviciul prin redundanță sau printr-o cale independentă [2]

Funcționarea degradată controlat permite continuarea activității cu performanță redusă: limitarea vitezei, dezactivarea funcțiilor neesențiale sau operarea locală fără servicii centrale. Modul degradat se proiectează înaintea incidentului, precizând funcțiile disponibile, limitele și criteriile de revenire.

Recuperarea poate însemna repornirea unei sarcini, resetarea controlerului sau restaurarea configurației, păstrând coerența dintre software și procesul fizic. După repornire, trebuie cunoscute poziția actualelor, etapa secvenței, materialul aflat în proces, ultima comandă confirmată și validitatea datelor. Revenirea software-ului la o versiune anterioară nu readuce o piesă pe transportor și nu reduce temperatura unui cuptor. De aceea, sistemele industriale preferă recuperarea prin reconstrucția unei stări valide, pornind de la situația fizică observată.

O componentă repornită sau reparată nu se reintegrează automat. Pot fi necesare autotestarea, verificarea versiunii, sincronizarea stării, compararea cu sistemul activ și confirmarea operatorului. Un server de rezervă cu o bază de date învechită poate fi tehnic disponibil, dar nepotrivit pentru preluarea serviciului.

Revenirea la o versiune anterioară acoperă doar latura software a recuperării. Componenta este reintegrată numai după aplicarea verificărilor de coerență fizică și logică descrise anterior. Cerințele privind autenticitatea pachetelor software, vulnerabilitățile și distribuirea corecțiilor sunt tratate în secțiunea 8.3. Tabelul 8.1 sintetizează principalele mecanisme pentru siguranța în funcționare, împreună cu limitările care trebuie verificate în proiectare.

Tabelul 8.1. Mecanisme pentru siguranța în funcționare și limitele lor

Mecanism	Rol principal	Limitare care trebuie verificată
Redundanță fizică	Mentținerea sau compararea serviciului	Defecte comune, alimentare comună și software comun
Diversitate	Reducerea cauzelor comune prin tehnologii diferite	Complexitate sporită de testare și interpretări diferite ale cerințelor
Mecanism de supraveghere și semnal periodic de prezență	Detectarea lipsei de progres sau a blocării comunicației	Semnalizează activitatea, nu și corectitudinea datelor calculate
Redundanță analitică	Detectarea abaterilor prin modele și relații fizice	Dependență directă de acuratețea modelului și calitatea senzorilor
Partiționare	Limitarea propagării defectelor între module	Vulnerabilitate la resurse hardware și magistrale partajate
Funcționare degradată	Mentținerea temporară a unui serviciu minim critic	Durată limitată; impune măsuri compensatorii de securitate
Copii de siguranță și revenire la versiunea anterioară	Restaurarea rapidă a configurației și software-ului	Nu restaurează automat starea mecanică a instalației din teren
Testarea periodică a funcției de protecție	Verificarea integrității lanțului de oprire	Trebuie să includă întregul lanț fizic (senzor–logică–actuator)

8.3 Securitatea cibernetică industrială și reziliența ciclului de viață

Analiza riscului cibernetic în mediile OT identifică activele critice, vectorii de atac, vulnerabilitățile tehnologice și impactul fizic potențial asupra instalației [166]. Conceptual, Riscul (R) este definit ca produsul dintre Probabilitatea de materializare a amenințării (P) și Severitatea consecințelor operaționale (C). În domeniul securității industriale, probabilitatea este dificil de cuantificat strict statistic, întrucât atacatorul își adaptează dinamic tehnicile; prin urmare, această relație ghidează o ierarhizare calitativă a măsurilor de protecție (conform IEC 62443), punând accent pe limitarea consecințelor prin segmentarea în zone și conducte de securitate [3], [166].

Inventarul OT trebuie să includă funcția activului, versiunea de firmware, protocoalele, impactul fizic și metoda de restaurare. Un dispozitiv fără adresă IP poate fi esențial prin rolul său fizic. Inventarul se corelează cu managementul modificărilor.

Conectarea sistemelor industriale creează căi de acces prin VPN-uri (*Virtual Private Network*), API-uri și relații cu furnizorii. Protocoalele vechi pot transmite comenzi fără autentificare; echipamentele care nu pot fi modernizate se protejează prin zone izolate și limitarea strictă a funcțiilor de scriere. Apărarea în profunzime combină segmentarea, zona demilitarizată industrială (IDMZ), privilegiul minim, accesul controlat la distanță, monitorizarea și protecțiile locale. O zonă grupează active cu cerințe similare și include identități, politici și responsabilități. Fluxurile se documentează prin direcție, protocol și autoritate [3], [4], [167].

Fiecare zonă din modelul Purdue (Figura 8.1) este delimitată de canale de comunicație controlate. Fluxurile care traversează zonele trebuie autorizate și jurnalizate. Nicio comandă nu trebuie să ajungă direct din zona de întreprindere în zona de control fără validare intermediară.

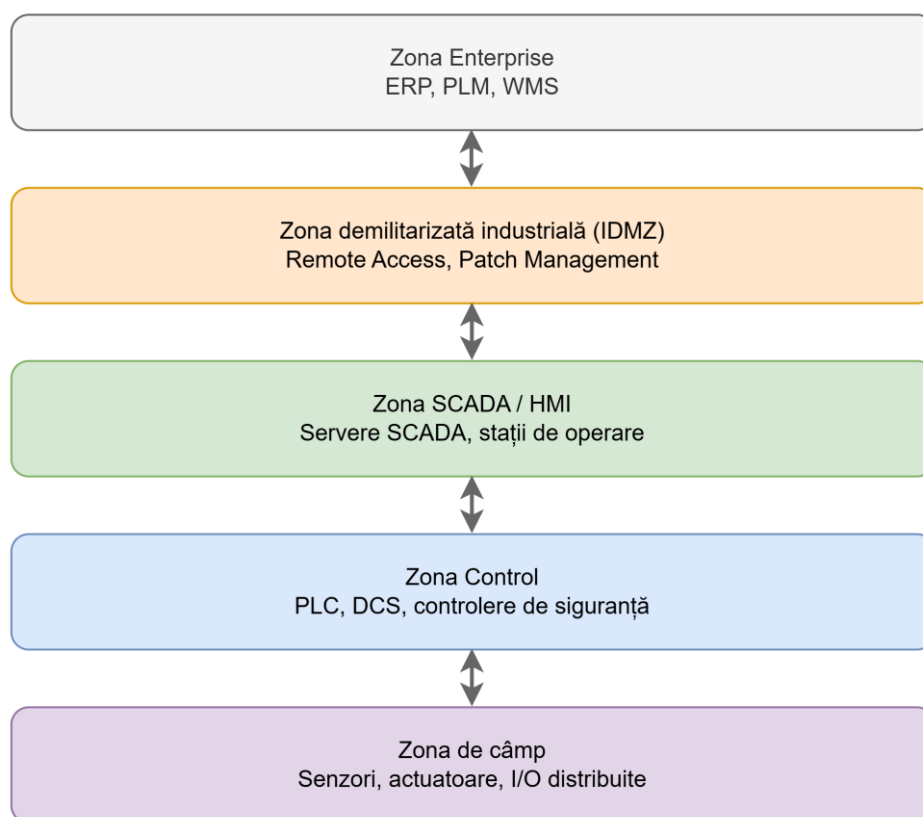


Figura 8.1 Arhitectură de securitate pe niveluri (model Purdue adaptat)

IEC 62443-3-3 structurează nivelurile de securitate în raport cu clase de atacatori. Nivelul nu poate fi acordat doar pentru că o componentă este declarată „securizată”; el trebuie susținut de întreaga arhitectură, de procedurile operaționale și de independența barierelor [167]. Principiul privilegiului minim impune separarea vizualizării, confirmării alarmelor, modificării referințelor și schimbării logicii. Modelul Zero Trust elimină încrederea bazată exclusiv pe poziția în rețea, fără a face controlul local dependent de un serviciu extern de identitate [168]. În domeniul tehnologiei operaționale (OT), principiul de securitate nu impune ca fiecare mesaj de câmp să fie verificat printr-un serviciu cloud. Identitățile bazate pe certificate, autentificarea serviciilor, segmentarea rețelei, accesul acordat doar la momentul utilizării și monitorizarea se aplică la frontierele adecvate, fără a face controlul local dependent, la fiecare ciclu, de un serviciu extern de identitate [168].

Accesul la distanță trebuie să utilizeze autentificare puternică, aprobare, interval limitat și jurnalizare. Drepturile se limitează la echipamentul, funcția și perioada aprobate. Securitatea cibernetică trebuie să protejeze comportamentul instalației, nu doar informația [162]. Amenințările includ injectarea de date false, manipularea comenzilor, refuzul serviciului, retransmisia malițioasă a mesajelor, atacurile asupra sincronizării și modificarea modelelor sau a configurațiilor.

Atacurile asupra sincronizării temporale merită o atenție specială în mediile OT. Un adversar care perturbă sau falsifică semnalele de sincronizare (de exemplu, prin atacuri asupra protocolului PTP/IEEE 1588) poate introduce erori de marcaj temporal fără a modifica conținutul mesajelor. Efectele includ ordonarea incorectă a evenimentelor, pierderea coerenței între măsurări distribuite și, în cazuri extreme, comenzi aplicate în ferestre temporale nepotrivite. Similar, modelele de inteligență artificială utilizate în buclele de control sau de predicție pot fi compromise prin atacuri de tip „*data poisoning*” (date de antrenare alterate) sau „*adversarial examples*” (intrări construite pentru a induce erori sistematice). Aceste amenințări nu sunt acoperite de mecanismele clasice de autentificare și criptare; ele necesită validări suplimentare de coerență fizică, redundanță analitică și monitorizarea derivatei modelului, așa cum s-a discutat în capitolele 5 și 7. O valoare falsificată se poate încadra în domeniul numeric permis și poate influența controlul; verificarea se completează, așadar, cu relații fizice, observatori, senzori independenți și validare locală, înainte ca informația să capete autoritate asupra actuatorului [162], [169].

Un mesaj autentic poate fi capturat și retransmis. Protecția împotriva retransmisiei impune numere de secvență, marcaje temporale, nonce-uri (număr de unică folosință) și verificarea prospețimii; criptarea singură nu este suficientă. Aplicația trebuie să respingă comenzile expirate sau incompatibile cu starea curentă. Prin atacarea sursei de timp, se poate perturba ordonarea evenimentelor, controlul sincronizat, alarmele și analiza incidentelor.

Administrarea corecțiilor software pornește de la vulnerabilitatea remediată; pachetul se obține dintr-o sursă autorizată, se verifică și se instalează controlat. Restaurarea după o actualizare eșuată este tratată în secțiunea 8.2 [170].

Copiile de siguranță trebuie să acopere logica PLC, proiectele HMI, bazele de date SCADA, rețelele, certificatele și configurațiile de rețea. Exercițiile periodice trebuie să demonstreze că restaurarea este posibilă. Răspunsul la incidente apără simultan procesul și dovezile; izolarea se coordonează cu echipele de operare și siguranță [3], [171].

8.4 Cadrul european și efectele asupra proiectării industriale

Reglementările europene nu înlocuiesc analiza tehnică, dar modifică modul în care riscul, modificările și responsabilitățile trebuie documentate. Pentru proiectele industriale, sunt relevante îndeosebi Regulamentul privind inteligența artificială (AI Act) și Regulamentul privind reziliența cibernetică (CRA). AI Act a intrat în vigoare la 1 august 2024, iar majoritatea prevederilor generale se aplică de la 2 august 2026. Sistemele AI cu grad ridicat de risc urmează un calendar distinct: 2 decembrie 2027 pentru sistemele autonome cu grad ridicat de risc și 2 august 2028 pentru cele încorporate în produse reglementate [172], [173].

Pentru aplicațiile care intră sub incidența acestor reglementări, consecințele ingineresti includ managementul riscului, governanța datelor, documentația tehnică, jurnalizarea, instrucțiunile de utilizare, supravegherea umană, robustețea, acuratețea, securitatea cibernetică și monitorizarea post-punere în funcțiune. CRA (*Cyber Resilience Act*) impune cerințe de securitate pe durata întregului ciclu de viață, cu aplicare generală din decembrie 2027 [174].

Din perspectivă tehnică, CRA consolidează nevoia de proiectare securizată, de configurații implicite sigure, de identificare a componentelor și versiunilor afectate și de o cale controlată pentru distribuirea remedierilor, aspecte ale căror detalii operaționale sunt tratate în secțiunea 8.3. Efectul comun al acestor reglementări este trecerea de la validarea punctuală la responsabilitatea continuă, pe întregul ciclu de viață. Riscul, datele, software-ul, modelele, vulnerabilitățile, actualizările și incidentele trebuie urmărite după introducerea pe piață și după integrarea în instalație, iar responsabilitățile producătorului, integratorului, proprietarului și operatorului, deși distincte, trebuie clarificate contractual și tehnic.

8.5 HMI, managementul alarmelor și autoritatea umană

Interfața om-mașină (HMI) permite operatorului să perceapă starea procesului, să interpreteze abaterile și să acționeze asupra instalației. HMI-ul face parte din bucla de control alături de proces, automatizare, operator și proceduri. Calitatea sa se evaluează prin capacitatea de a susține decizii corecte și oportune, nu prin numărul de ecrane sau de animații [175], [176].

Un nivel mai ridicat de automatizare nu este automat superior [177]. Deși reduce sarcina în regim normal, poate diminua înțelegerea necesară pentru a prelua controlul în situații rare. HMI-ul trebuie să facă vizibile modul activ, limitele automatizării, acțiunile executate, condițiile de blocare și motivele deciziilor. Nivelul de autoritate se alege în funcție de consecințe, nu doar de fezabilitatea tehnică [177].

Conștientizarea situației presupune perceperea elementelor relevante, înțelegerea semnificației lor și anticiparea evoluției. Un ecran care afișează exclusiv valori numerice sprijină mai ales percepția; curbele de evoluție, limitele și rata de schimbare ajută operatorul să înțeleagă și să anticipeze [175].

ISA-101 tratează HMI-ul ca pe un sistem cu ciclu de viață: filozofie de proiectare, standarde grafice, implementare, verificare, operare și îmbunătățire. Orice modificare a procesului sau a logicii impune reverificarea ecranelor, alarmelor și comenzilor, deoarece o variabilă reasociată poate ajunge să comande un alt echipament fără ca eroarea să fie evidentă vizual [178].

Informația se structurează după sarcina operatorului și ierarhia procesului. Ecranul de ansamblu redă starea generală, producția și abaterile majore; ecranele de zonă și de echipament permit investigarea progresivă. Nivelul superior nu reproduce toate detaliile, iar grafica reflectă relațiile fizice, direcția fluxului și starea funcțională.

Principiile HMI de înaltă performanță recomandă fundaluri neutre și rezervarea culorilor saturate pentru condițiile care reclamă atenție. Dacă totul este intens colorat, alarma nu se mai distinge. Culoarea este completată de text, simbol sau formă și se utilizează consecvent; roșul nu semnifică automat „oprit” dacă oprirea este starea normală [178], [179].

Valorile se afișează cu unitatea de măsură și cu o precizie justificată de senzor. Multe zecimale pot sugera o exactitate iluzorie. O valoare izolată nu arată direcția de evoluție: o temperatură stabilă la 90 °C este fundamental diferită de una care urcă rapid spre aceeași valoare. Curbele de evoluție se raportează la limite și la intervale de timp relevante; scalarea automată nu trebuie să mascheze apropierea de o limită.

HMI-ul diferențiază valorile valide, incerte, învechite, substituite, estimate, introduse manual și simulate. Păstrarea ultimei valori după pierderea comunicației, fără marcarea acestui fapt, poate induce operatorul în eroare. Se indică momentul ultimei actualizări și, când contează, originea marcajului temporal [65], [107].

Atenția și memoria de lucru ale operatorului sunt limitate. Prea multe valori, prea multe niveluri de navigare, simbolurile inconsistente și avalanșa de alarme măresc încărcarea cognitivă. Reducerea

acesteia nu înseamnă eliminarea informației, ci organizarea ei după prioritate, context și acțiunea necesară [180].

O alarmă notifică o condiție anormală care reclamă un răspuns al operatorului într-un interval relevant. Schimbările de stare, evenimentele sau mesajele informative nu trebuie transformate automat în alarme. Organizația definește o filozofie de alarmare: fiecare alarmă are o cauză, o consecință, o acțiune realistă, un timp de răspuns și o prioritate justificată. Dacă operatorul nu poate acționa, mesajul nu este o alarmă [181], [182].

Prioritatea unei alarme se stabilește în funcție de severitatea consecinței și de timpul disponibil pentru reacție. Conform standardelor ANSI/ISA-18.2 [181] și IEC 62682 [182], o alarmă este justificată doar dacă respectă simultan trei criterii:

1. Indică o abatere anormală neprevăzută a procesului;
2. Impune o acțiune operatorie corectivă explicită;
3. Este declanșată suficient de devreme pentru a permite evitarea pericolului.

Clasificarea nediferențiată a surselor de alarme drept „critice” anulează semnificația acestei categorii și generează fenomenul de avalanșă de alarme (*alarm flood*), în care operatorul este copleșit de notificări secundare provocate de un singur eveniment inițiator. Sistemul HMI modern gestionează aceste situații prin tehnici avansate:

- Filtrarea comutării repetate (*chattering*): prin introducerea de benzi moarte (*deadband*) și temporizări;
- Suspendarea temporară (*shelving*): permite operatorului să mascheze motivat o alarmă repetitivă pe o durată fixă;
- Suprimarea automată bazată pe stare: dezactivează alarmele irelevante în regimurile de oprire sau pornire ale instalației.

Confirmarea alarmei de către operator atestă doar luarea la cunoștință, nu rezolvarea problemei. Interfața HMI trebuie să diferențieze vizual clar stările: activă, confirmată, revenită la normal, suspendată și suprimată automat.

Mesajul de alarmă trebuie să indice ce s-a întâmplat, unde și ce intervenție se recomandă. „Alarmă 3412” este mai puțin util decât „Presiune ridicată la pompa P-204; verificați valva de refulare și debitul”. Textul este suficient de specific pentru acțiune, fără a pretinde confirmarea unei cauze doar probabile. Calea de comandă, mai critică decât cea de afișare, este ilustrată în Figura 8.2.

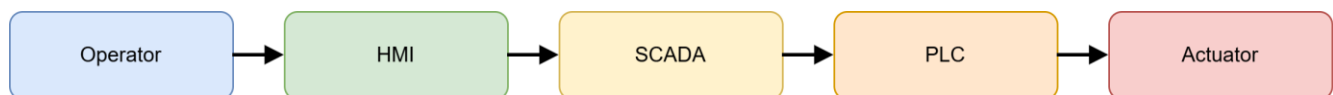


Figura 8.2. Calea de comandă de la operator la actuator

Fiecare etapă verifică identitatea, permisiunea, obiectul comandat, valoarea, starea procesului și acceptarea de către controler. HMI-ul nu trebuie să considere comanda reușită doar pentru că butonul a fost apăsât; acolo unde este posibil, se confirmă efectul fizic.

Confirmarea suplimentară se justifică pentru acțiunile greu reversibile, neobișnuite sau cu impact ridicat. Dialogurile de confirmare folosite pentru orice acțiune devin un reflex și își pierd valoarea. Confirmarea descrie consecința concretă. Valorile de referință se validează după domeniu, unitate, rată

maximă de schimbare și modul de operare; o valoare invalidă este respinsă explicit, nu înlocuită tacit cu o valoare implicită. Conturile partajate reduc trasabilitatea și se evită pentru acțiunile importante. Autorizarea se bazează pe rol și pe context: operatorul poate modifica valorile de referință în limitele aprobate, dar nu logica PLC; furnizorul accesează doar echipamentul și intervalul autorizate. Accesul în citire se separă de control, astfel încât tablourile de bord la distanță să poată primi telemetrie fără drept de comandă.

Perspectiva Industry 5.0 aduce transparență, ergonomie, menținerea competențelor, delimitarea autorității și posibilitatea reală de a interveni. Interfața se adaptează mediului de lucru (mănuși, zgomot, vibrații, lumină, distanță). Informația critică nu se codează exclusiv prin culoare; semnalele sonore au întotdeauna și o reprezentare vizuală. Adaptarea interfeței la rol rămâne predictibilă, pentru că mutarea dinamică a comenzilor critice crește riscul de eroare [17], [20].

Un sistem de inteligență artificială poate recomanda verificarea unui echipament, modificarea unei referințe sau planificarea unei intervenții. Recomandarea include motivul, observațiile relevante, nivelul de încredere, domeniul de validitate și limitele cunoscute. O explicație bazată pe creșterea unei componente spectrale și pe temperatura unui lagăr este mai utilă decât o importanță abstractă a caracteristicilor, dar nu se prezintă drept dovadă cauzală cât timp modelul a identificat doar o corelație [32], [183].

Nivelul de încredere se redă într-o formă utilă deciziei, preferabil prin categorii precum încredere ridicată, moderată, date insuficiente sau rezultat în afara domeniului validat, nu printr-un procent cu zecimale iluzorii. Scopul este calibrarea încrederii, nu maximizarea ei: un sistem care greșește rar, dar grav, poate genera încredere excesivă, iar unul care produce alarme frecvente riscă să fie ignorat. Preluarea manuală a controlului nu trebuie să fie simbolică. Procedura, timpul de reacție și consecințele se testează. Operatorul știe ce a decis automatizarea, ce poate modifica, cine aprobă schimbarea și ce protecții independente rămân active. Autoritatea umană nu trebuie să devină o cale necontrolată de eludare a funcțiilor de siguranță.

Indiferent de traseul informației (PLC, SCADA, arhivă, OPC UA, servicii edge), interfața indică sursa, calitatea, prospețimea și modul activ. Rolurile tehnice ale nivelurilor sunt prezentate în capitolele 2 și 10; succesiunea solicitare–validare–execuție–confirmare este sintetizată în Figura 8.2. Pierderea comunicației se afișează clar și persistent, iar interfața blochează comenzile bazate pe o stare necunoscută. Regulile de marcarea a valorilor neactualizate au fost definite mai sus, autonomia controlului local în capitolul 2, iar funcționarea degradată în secțiunea 8.2.

8.6 Validarea operațională integrată

Siguranța în funcționare, securitatea cibernetică și interacțiunea om–sistem se validează împreună, prin scenarii care antrenează software-ul, rețeaua, echipamentele fizice, interfața și procedurile. Testarea separată a unui algoritm, a unui firewall sau a unui ecran nu dovedește comportamentul corect al serviciului cyber-fizic. Se verifică modul în care defectul este detectat, informația este prezentată, automatizarea reacționează, operatorul intervine și procesul revine într-o stare cunoscută. Tabelul 8.2 propune un set de scenarii integrate de validare și comportamentul așteptat.

Pentru fiecare scenariu se definesc condiția declanșatoare, serviciul afectat, starea așteptată, timpul maxim de detecție și reacție, informația prezentată operatorului, confirmarea efectului fizic și criteriul de revenire. Testele acoperă condițiile normale, tranzițiile, pierderile parțiale și combinațiile de defecte. Ele se repetă după modificarea logicii, actualizarea software-ului, schimbarea arhitecturii de rețea, înlocuirea unui echipament cu impact semnificativ sau modificarea procedurilor. Restaurarea

software-ului nu echivalează cu restaurarea instalației; confirmarea unei comenzi nu echivalează cu producerea efectului fizic.

Tabelul 8.2. Scenarii integrate de validare

Scenariu	Comportament așteptat
Defectarea unui senzor	Valoarea este marcată invalidă, se emite alarmă, controlul trece în mod degradat
Pierderea sistemului SCADA	Controlul local PLC continuă autonom; datele sunt stocate în buffer local
Compromiterea unui cont	Politica de drepturi (RBAC) blochează modificarea logicii, rețetelor și configurației
Actualizare eșuată	Sistemul revine automat (<i>rollback</i>) la versiunea funcțională anterioară
Avalanșă de alarme	HMI-ul evidențiază evenimentul inițiator și filtrează notificările secundare
Indisponibilitatea modelului de inteligență artificială	Sistemul comută pe logica deterministă de rezervă fără oprirea procesului
Restaurarea unui server	Sunt verificate versiunea, certificatele de securitate și sincronizarea de stare
Pierderea sursei de timp	Datele dependente de timestamp sunt marcate drept nesincronizate
Pierderea unui canal redundant	Defectul este izolat, comutarea se face fără șoc, modul degradat este semnalizat
Comandă necorespunzătoare transmisă la distanță	PLC-ul local respinge comanda incompatibilă cu interblocările fizice
Mecanism de protecție blocat	Sistemul detectează lipsa confirmării fizice și nu consideră acțiunea executată
Recuperare după întreruperea alimentării	Starea software este reconciliată cu poziția mecanică reală înainte de repornire

Reziliența se dovedește prin capacitatea sistemului de a detecta pierderea, de a limita propagarea, de a face vizibilă starea, de a păstra protecțiile locale și de a reveni controlat la o stare cunoscută. Siguranța în funcționare, securitatea cibernetică și autoritatea umană sunt tratate ca proprietăți ale arhitecturii și ale întregului ciclu de viață.

Partea a V-a. Integrarea fabricii

9. Robotică industrială, roboți colaborativi și sisteme mobile autonome

Robotica industrială reunește mecanica, acționarea, percepția, controlul, comunicația și siguranța într-un sistem capabil să execute operații fizice repetabile. Un robot poate mișca brațul repede și precis, dar celula poate produce lent dacă piesele nu ajung la timp, sunt greu de prins sau provoacă opriri dese. De aceea, evaluăm aplicația completă, nu doar robotul: dispozitivul de prindere, senzorii, alimentarea cu piese, controlerul, spațiul de lucru și interacțiunea cu operatorul [184]. Pornim de la operația dorită — de exemplu, mutarea unei piese de pe o bandă într-o cutie — și stabilim cum trebuie să coopereze aceste componente. Abia apoi alegem robotul și verificăm timpul de ciclu și condițiile de siguranță [185], [186].

În perioada 2015–2026, integrarea roboților cu viziunea artificială, cu PLC-urile, cu sistemele MES (*Manufacturing Execution System*) și WMS (*Warehouse Management System*), cu simularea și cu platformele de analiză a devenit la fel de importantă ca programarea mișcărilor. Roboții colaborativi extind posibilitățile de lucru în spații comune cu oamenii, însă caracterul colaborativ trebuie demonstrat pentru aplicația completă. Roboții mobili autonomi (AMR) introduc localizarea, planificarea și coordonarea flotei în fluxurile logistice. Modelele bazate pe învățare automată pot sprijini percepția, prinderea și recuperarea, dar nu înlocuiesc controlul determinist, protecțiile locale și procedurile de recuperare — trebuie încadrate prin limite, monitorizare și posibilitatea revenirii la o strategie verificată.

9.1 Sistemul robotic și performanța aplicației

Un robot industrial este un manipulator reprogramabil, cu mai multe axe, utilizat pentru manipulare, asamblare, sudare, vopsire, paletizare, prelucrare și inspecție. Structura mecanică — elemente legate prin articulații de rotație sau de translație — determină forma spațiului de lucru, rigiditatea și orientările posibile ale uneltei. Acționarea electrică domină datorită preciziei, eficienței și integrării facile cu servomotoarele. Fiecare axă include un motor, o transmisie, un traductor de poziție și, unde este necesar, o frână; jocul mecanic, elasticitatea, frecarea și variațiile termice influențează precizia reală.

Cele șase grade de libertate necesare poziționării și orientării complete corespund celor trei translații și trei rotații ale efectorului. Roboții cu mai mult de șase axe sunt cinematic redundanți pentru sarcini carteziane obișnuite; redundanța poate fi exploatată pentru evitarea obstacolelor, îndepărtarea de limitele articulare și alegerea unor configurații mai eficiente. Trebuie analizate însă spațiul de lucru accesibil, singularitățile, sarcina, viteza, rigiditatea și interferențele cu instalația.

Sarcina utilă nominală nu este doar masa maximă transportată: poziția centrului de masă, momentul de inerție și accelerațiile influențează solicitarea fiecărei articulații. Robotul se selectează pe baza diagramelor de sarcină și a traiectoriilor reale; profilul de mișcare trebuie corelat cu stabilitatea prinderii și cu rigiditatea efectorului. Accelerația permisă de manipulator poate fi prea mare pentru produsul transportat, chiar dacă masa respectă specificația nominală.

Acuratețea descrie apropierea dintre poziția comandată și cea realizată; repetabilitatea reprezintă capacitatea de a reveni în aceeași poziție. Repetabilitatea este suficientă pentru punctele memorate cu terminalul de programare, dar acuratețea și calibrarea devin esențiale pentru programarea offline, coordonarea mai multor roboți și ghidarea prin viziune artificială. Calibrarea se reverifică după coliziuni, modificări mecanice sau schimbarea sistemelor de coordonate.

Configurațiile robotice se aleg în funcție de sarcină: roboții articulați oferă flexibilitate, cei SCARA (*Selective Compliance Assembly Robot Arm*) sunt eficienți pentru inserție, structurile delta pentru sortare rapidă, iar sistemele carteziane asigură o cinematică simplă. Roboții colaborativi integrează funcții de limitare și monitorizare care facilitează colaborarea, dar acestea nu constituie prin ele însele o demonstrație de siguranță.

Efectorul final — dispozitiv de prindere, cap de sudare, unealtă de înșurubare sau instrument de măsurare — influențează disponibilitatea aplicației mai mult decât manipulatorul [187], [188]. Procesul poate eșua deși robotul execută traiectoria corect, din cauza alunecării, prinderii excentrice sau deformării piesei. Închiderea geometrică restricționează mișcarea prin geometria contactelor; închiderea prin forță se bazează pe forțele normale și pe frecare, stabilitatea depinzând de poziția contactelor, coeficientul de frecare și perturbațiile induse de mișcare [185], [189].

Dispozitivele de prindere mecanice folosesc degete acționate pneumatic sau electric; cele cu vid se potrivesc suprafețelor relativ etanșe; cele magnetice sunt destinate materialelor feromagnetice. Dispozitivele moi și subacționate se adaptează obiectelor fragile, dar pot avea o precizie mai redusă. Selecția ține cont de temperatură, porozitate, contaminare, toleranțe și efectul forței asupra produsului.

Comanda unui dispozitiv de prindere nu certifică executarea prinderii. Un sistem cu vid trebuie să monitorizeze presiunea; un dispozitiv electric poate utiliza poziția degetelor și curentul motorului pentru verificare. Efectorii inteligenți integrează senzori de poziție, forță și tactili, pot detecta alunecarea și pot adapta forța de prindere — o forță mare și fixă poate deteriora produsul și accelera uzura. Tabelul 9.1 sintetizează criteriile de selecție și erorile frecvente de proiectare pentru componentele principale ale unei celule robotizate.

Tabelul 9.1. Selectarea componentelor principale ale aplicației robotice

Element	Criterii dominante	Erori frecvente de proiectare
Manipulator	Spațiu de lucru, sarcină utilă, inerție, rigiditate, timp de ciclu	Dimensionare exclusiv după masa statică, ignorând inerția dinamică
Efector final	Geometrie piesă, forță de strângere, deformare, confirmare prindere	Considerarea comenzii electrice drept dovadă a prinderii fizice
Alimentarea pieselor	Ritm (<i>takt time</i>), variația orientării, separare piese	Compensarea prin software a unei așezări mecanice instabile
Senzori	Tip mărime, precizie, latență, robustețe la mediu	Folosirea unui singur semnal fără validare prin plauzibilitate
Dispozitive de fixare	Repetabilitate mecanică, accesibilitate, rigiditate	Toleranțe constructive incompatibile cu precizia necesară
Software și interfețe	Stări explicite, sincronizare magistrală, recuperare la eroare	Dependențe ascunse între programe și lipsa procedurii de resetare

9.2 Cinematica, dinamica și controlul interacțiunii

Cinematica descrie relația geometrică dintre coordonatele articulare ale robotului și postura (poziția și orientarea) efectorului final în spațiul de lucru, fără a lua în calcul forțele care generează mișcarea [186]. Fundamentele cinematice și dinamice sunt tratate pe larg în literatura clasică de robotică [190].

Cinematica directă calculează poziția și orientarea efectorului pornind de la unghiurile sau deplasările fiecărei articulații (q). Pentru un manipulator serial cu n grade de libertate, transformarea omogenă globală se obține prin multiplicarea matricilor de transformare ale fiecărui segment:

$${}^0T_n(q) = {}^0T_1(q_1) {}^1T_2(q_2) \dots {}^{n-1}T_n(q_n)$$

Fiecare transformare omogenă înglobează matricea de rotație și vectorul de poziție, parametrii Denavit–Hartenberg oferind convenția standardizată pentru atașarea sistemelor de coordonate pe fiecare axă [186], [189].

Cinematica inversă rezolvă problema inversă: determină configurația articulară (q) necesară pentru a atinge o postură țintă dorită a efectorului (T_d):

$$q = f^{-1}(T_d)$$

Spre deosebire de cinemática directă, ecuațiile cinematicii inverse sunt neliniare și pot genera soluții multiple, soluții inexistente (postură în afara spațiului de lucru) sau configurații incompatibile cu limitele mecanice ale brațului.

La nivel diferențial, matricea Jacobian $J(q)$ descrie sensibilitatea mișcării, legând vitezele articulare ($\dot{q}$) de viteza liniară și unghiulară a efectorului în spațiul cartezian (V), precum și forțele externe aplicate (F) de cuplurile din motoare (τ):

$$V = J(q)\dot{q}$$

$$\tau = J^T(q)F$$

În rezolvarea cinematicii diferențiale pentru controlul traiectoriei, apar în practică două aspecte esențiale:

- Singularitățile cinemate: configurații particulare în care determinantul Jacobianului se anulează (de exemplu, la întinderea completă a brațului sau când două axe devin coliniare). Într-o singularitate, anumite mișcări carteziene ar impune teoretic viteze articulare infinite, fiind fizic imposibile. Planificatoarele de traiectorie utilizează metode de inversare amortizată (*damped least-squares*), care mențin vitezele motoarelor în limite sigure în vecinătatea singularității, acceptând o deviere minimă temporară de la traiectoria carteziană ideală;
- Redundanța cinematică: roboții cu 7 sau mai multe grade de libertate posedă grade interne de mobilitate ce permit modificarea posturii brațului fără a deplasa efectorul final. Această mișcare în „spațiul nul” este exploatată algoritmic pentru optimizări secundare, precum evitarea coliziunilor cu echipamentele din celulă sau minimizarea efortului energetic.

Figura 9.1 ilustrează sistemele de coordonate atașate fiecărei articulații și legătura dintre configurația articulară q și postura efectorului final.

Pentru fixarea intuitivă a acestor noțiuni, se consideră un braț planar simplu cu două segmente (L_1, L_2) și două articulații de rotație (θ_1, θ_2). Poziția carteziană a vârfului este:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

Jacobianul acestui sistem are dimensiunea 2×2 . Când brațul este complet extins ($\theta_2 = 0$), determinantul Jacobianului devine zero, apărând o singularitate la limita anvelopei de lucru: robotul nu se mai poate deplasa instantaneu spre exterior, indiferent de vitezele imprimate motoarelor.

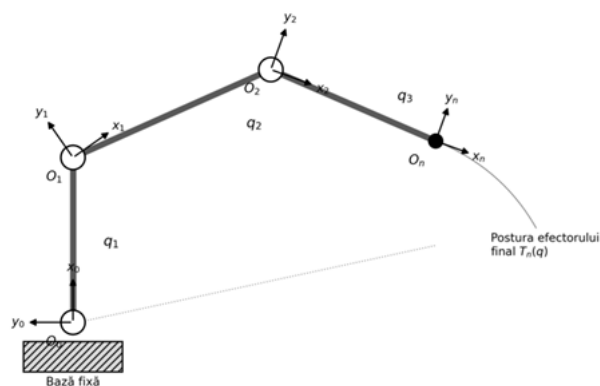


Figura 9.1. Sistemele de coordonate ale unui manipulator serial și legătura dintre configurația articulară și postura efectorului final

Dinamica descrie relația fizică dintre mișcare, accelerații, forțe și cuplurile dezvoltate de acționări [185], [186]:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau + \tau_{ext}$$

unde $M(q)$ este matricea de inerție, $C(q, \dot{q})\dot{q}$ reprezintă forțele centrifuge și Coriolis, $g(q)$ este vectorul gravitațional, $f(\dot{q})$ forțele de frecare, iar τ_{ext} reprezintă cuplurile generate de contactul cu mediul. Dinamica inversă calculează cuplurile τ necesare pentru a urmări o traiectorie impusă și este utilizată pentru comanda anticipativă (*feedforward*) și dimensionarea servomotoarelor; dinamica directă calculează accelerațiile rezultate și este baza simulării realiste a celei robotizate [185], [186].

În timp ce pentru manipularea liberă în aer este suficient un control rigid de poziție (cum este comanda prin cuplu calculat, care compensează neliniaritățile dinamice), procesele tehnologice care implică contact fizic direct (asamblare, debavurare, lustruire sau ghidare manuală de către operator) necesită controlul forței și al interacțiunii.

Controlul rigid aplicat la operații de contact generează forțe excesive chiar și la erori de poziționare de sub un milimetru, riscând distrugerea piesei sau avarierea mecanică a robotului. De aceea, se utilizează două strategii principale:

- Controlul impedenței: modelează interacțiunea brațului ca un sistem mecanic virtual de tip masă–arc–amortizor, ajustând forța exercitată proporțional cu deformarea sau rezistența întâmpinată;
- Controlul admitanței: măsoară forțele de reacțiune cu un senzor montat pe flanșa robotului și generează dinamic corecții de traiectorie pentru a menține forța de contact la valoarea prescrisă.

Parametrii de rigiditate și amortizare virtuală se calibrează în funcție de operație: o rigiditate prea mare generează șocuri mecanice, în timp ce o amortizare insuficientă produce oscilații instabile. În toate cazurile, bucla de control al forței este dublată de monitorizări hardware independente ale limitelor de viteză și cuplu.

9.3 Percepția, prinderea bazată pe învățare și învățarea prin întărire

Percepția permite adaptarea la variațiile de poziție, formă și mediu. Camerele RGB, camerele de adâncime și norii de puncte localizează obiecte, estimează postura și aleg prinderea. Principiile generale ale sistemelor de viziune au fost prezentate în capitolul 6. Pentru robotică sunt esențiale

transformarea corectă între sistemul camerei și cel al robotului și sincronizarea între observație și mișcare.

Modelele bazate pe învățare automată pot estima punctele și orientările de prindere, fiind utile pentru piese dispuse dezordonat și obiecte variabile. Predicția geometrică nu garantează însă stabilitatea fizică: o prindere aparent corectă poate eșua din cauza frecării, deformării sau distribuției masei. Contactul și stabilitatea trebuie confirmate prin senzori, iar sistemul trebuie să poată reîncerca sau solicita intervenția operatorului. Modelele TinyML pot executa local funcții de detectare a contactului sau alunecării, reducând latența, dar nu înlocuiesc protecțiile de siguranță [103].

Învățarea prin întărire (RL-*Reinforcement Learning*) modelează interacțiunea dintre un agent și mediul său [191], [192]. Starea poate cuprinde pozițiile articulare, imaginile, forțele și poziția obiectului; acțiunea poate fi un cuplu, o viteză sau o abilitate de nivel superior. Metodele bazate pe funcția de valoare estimează valoarea acțiunilor; cele bazate pe gradientul politicii optimizează direct politica; arhitecturile actor–critic combină ambele abordări [193].

Recompensa trebuie să reflecte obiectivul industrial complet. Dacă se recompensează numai timpul de ciclu, politica poate selecta mișcări agresive; dacă penalizarea contactului este prea blândă, robotul poate accepta interacțiuni riscante. Funcțiile de siguranță nu se implementează exclusiv prin penalizări — acestea rămân obiective ale optimizării, nu bariere independente.

Învățarea direct pe robotul fizic este costisitoare. Simularea, demonstrațiile și învățarea prin imitare reduc explorarea pe echipamentul real. Variația aleatoare controlată a domeniului de simulare (modificarea masei, frecării, iluminării, texturii) reduce dependența de un singur mediu virtual, dar nu înlocuiește validarea pe sistemul real [194]. Strategia de rezervă poate recurge la un planificator clasic sau la oprirea controlată. RL se justifică atunci când variabilitatea face programarea explicită neeconomică; nu se justifică atunci când o secvență clasică, transparentă și verificabilă satisface cerințele. Modificarea unei politici învățate se tratează cu versionare, testare și posibilitatea revenirii.

9.4 Arhitectura distribuită, integrarea și simularea

Într-o celulă robotizată, unitățile de acționare închid buclele rapide; controlerul robotului calculează cinematica și traiectoriile; PLC-ul coordonează secvența și echipamentele auxiliare. Sistemele bazate pe recunoaștere vocală oferă o interfață om–robot intuitivă pentru aplicații mobile [195]. Procesarea percepției poate rula pe un calculator edge, dar rezultatul trebuie asociat observației și poziției robotului pentru care este valabil. În buclele de ghidare vizuală, întârzierea totală între achiziția imaginii și aplicarea corecției trebuie verificată. Criteriile generale de repartizare a funcțiilor sunt prezentate în capitolele 2 și 3.

Protocoalele de comunicație deterministă — inclusiv EtherCAT și FSoE — sunt descrise în secțiunea 3.5. În robotică, valorile de referință și semnalele de reacție ale axelor trebuie să aparțină aceluiași ciclu și să rămână sincronizate cu transportoarele și senzorii. Sincronizarea rețelei nu este suficientă dacă aplicația procesează observațiile la momente diferite. Funcțiile de siguranță utilizează componente și căi de comunicație evaluate pentru acest rol, separate de percepția și optimizarea obișnuite [88], [89].

Integrarea cu PLC-ul se bazează pe stări și confirmări explicite: „pregătit”, „ocupat”, „finalizat”, „defect”, „stare sigură”. Comanda de pornire nu se acceptă dacă unealta, produsul și rețeta nu sunt confirmate. Schimbarea automată a uneltei verifică blocarea mecanică, utilitățile, identitatea uneltei, punctul central al uneltei (TCP), sarcina și configurația de siguranță.

În integrarea cu MES și WMS, sistemele superioare transmit operația sau misiunea logistică, iar controlerul celulei verifică starea, interblocările și configurația activă. La pierderea comunicației, se precizează dacă se finalizează ciclul curent sau dacă robotul poate utiliza o coadă locală de operații aprobate. Pentru fiecare operație se păstrează identitatea robotului, programul, rețeta, unealta, produsul, rezultatul și intervențiile. Datele alimentează analiza calității și mentenanța, fără a interfera cu buclele de control.

Nivelurile de testare și punerea în funcțiune virtuală sunt definite în secțiunea 7.3. Pentru o aplicație robotică, modelul trebuie să reprezinte geometria și cinematica pentru coliziuni, contactul și frecarea pentru prindere, caracteristicile senzorilor pentru percepție și secvențele pentru integrarea cu PLC-ul. Scenariile includ variația piesei, prinderea nereușită, contactul neprevăzut, pierderea senzorilor și procedurile de recuperare. Testarea verifică și timpul de ciclu pe platforma finală. Punerea în funcțiune virtuală nu înlocuiește validarea contactului și a comportamentului fizic pe instalația reală.

Configurația operațională asociază programul robotului cu versiunea controlerului, configurația de siguranță, TCP, modelul sarcinii, identitatea efectorului, calibrarea cameră-robot, rețeta și interfața cu PLC-ul. Aceste elemente se verifică împreună după orice modificare; o combinație incompatibilă poate genera erori de poziționare sau traiectorie chiar dacă fiecare componentă funcționează corect separat.

9.5 Roboți colaborativi și colaborarea om-robot

Un robot colaborativ nu înseamnă automat o aplicație colaborativă sigură. Evaluarea privește sistemul complet — robot, unealtă, piesă, proces, configurația spațiului și persoanele expuse. ISO 10218-1:2025 – succesor al ISO 10218-1:2011 [196] - stabilește cerințele de siguranță pentru roboții industriali [197], iar ISO 10218-2:2025 tratează integrarea, operarea și mentenanța aplicațiilor robotice [198]. Analiza acoperă contactul cu brațul, muchiile piesei, uneltele și riscul de prindere între robot și structurile fixe. Un robot lent poate rămâne periculos dacă transportă o piesă ascuțită [197], [198]. Tabelul 9.2 rezumă principalele moduri de operare colaborativă.

În oprirea monitorizată cu nivel de siguranță, mișcarea periculoasă este oprită și supravegheată. Ghidarea manuală trebuie activată intenționat, după verificarea condițiilor. Monitorizarea vitezei și a separării presupune calcularea distanței pe baza vitezelor robotului și persoanei, a timpilor de reacție și a incertitudinii. Limitarea puterii și a forței reduce consecințele contactului prin masă redusă, forme adecvate și monitorizarea cuplului. ISO/TS 15066:2016 oferă îndrumări suplimentare [22], [23], distingând contactele tranzitorii de cele cvasistatice.

În aplicarea regimului de limitare a puterii și forței conform ISO/TS 15066:2016, detecția coliziunii fără senzori tactili externi se bazează pe calculul cuplului rezidual ($r_{\tau} = \tau_{\text{masurat}} - \tau_{\text{model}}$). Controlerul compară permanent cuplurile dezvoltate de motoare cu cele estimate din modelul dinamic al robotului; apariția unui reziduu ce depășește un prag calibrat indică aplicarea unei forțe externe (atingerea operatorului), declanșând oprirea controlată în câteva milisecunde. Pentru a preveni alarmele false, algoritmul filtrează dinamic perturbațiile cauzate de frecările mecanice și de accelerațiile rapide ale traiectoriei.

Un reziduu mare semnalează o forță externă, dar trebuie diferențiat de frecare, accelerație și eroarea de model. Un prag prea sensibil generează opriri frecvente; unul prea îngăduitor întârzie reacția. Funcția de protecție nu se bazează exclusiv pe un model a cărui incertitudine nu este cunoscută.

Tabelul 9.2. Moduri uzuale de operare colaborativă

Mod	Principiu	Cerință operațională importantă
Oprire monitorizată cu nivel de siguranță	Mișcarea periculoasă este oprită cât timp persoana este în spațiul colaborativ	Reluarea mișcării se face controlat, fără a surprinde operatorul
Ghidare manuală	Operatorul conduce robotul prin forță manuală directă	Activare intenționată exclusiv prin buton de validare (<i>enabling switch</i>)
Monitorizarea vitezei și a separării	Viteza robotului scade proporțional cu distanța până la persoană	Include distanța de oprire, timpii de reacție și incertitudinea senzorilor
Limitarea puterii și a forței	Contactul fizic este menținut sub pragurile biomecanice	Limitare activă de cuplu și amortizare mecanică a muchiilor

Colaborarea eficientă depășește simpla evitare a coliziunii. Robotul comunică intenția, modul activ și condițiile de reluare. Oamenii preiau activitățile care solicită judecată și dexteritate; roboții execută mișcările repetitive și solicitante ergonomic [20], [175]. Pe lângă aplicațiile industriale, roboții au fost mobilizați în contexte de urgență sanitară [199]. Interfața face vizibile programul activ, viteza, modul și motivul opririi. Analiza riscului se repetă după schimbarea uneltei, a piesei, a sarcinii sau a configurației spațiului.

9.6 AMR, managementul flotei și ciclul de viață

Vehiculele ghidate automat (AGV – *Automated Guide Vehicle*) urmează trasee prestabilite; roboții mobili autonomi (AMR) percep mediul, se localizează și își replanifică traseul [200]. Un AGV este preferat în fluxuri stabile, unde predictibilitatea predomină; un AMR este mai potrivit când configurația spațiului și traficul se modifică frecvent.

Un AMR poate utiliza traductoare de poziție la roți, unități de măsurare inerțială (IMU), LiDAR, camere, senzori de adâncime și repere artificiale [201]. Niciun senzor nu este infailibil: LiDAR-ul poate fi păcălit de suprafețe reflectante, camera de iluminare, iar odometria de alunecare. Fuziunea senzorială estimează atât poziția, cât și incertitudinea. Robotica probabilistă reprezintă starea printr-o distribuție de probabilitate [202], [203]. În cazul vehiculelor autonome, percepția cooperativă permite partajarea informațiilor 3D între vehicule conectate, sporind robustețea detecției obstacolelor [204].

Localizarea Monte Carlo (MCL) operează cu particule; alinierea scanărilor corelează măsurările cu harta. Sistemul trebuie să detecteze momentul în care localizarea nu mai este credibilă și să reducă viteza sau să execute o procedură de relocalizare. SLAM (*Simultaneous Localization and Mapping*) estimează împreună poziția și harta, dar în mediul industrial elementele temporare — paleți, vehicule, uși — sunt frecvente. Sistemul deosebește structurile stabile de obstacolele dinamice; harta se versionează după schimbarea configurației.

Planificatorul global alege ruta; planificatorul local generează mișcarea imediată și evită obstacolele [205], [206], luând în calcul conturul complet al vehiculului și al încărcăturii. Distanța de oprire depinde de viteză, sarcină, aderență și timpii de procesare. Câmpurile de protecție se adaptează condițiilor de deplasare; validarea se face cu sarcina maximă și în condițiile reale ale pardoselii [207], [208].

Managerul de flotă atribuie misiuni, coordonează uși și ascensoare, planifică încărcarea bateriilor și previne blocajele [209]. Blocajul reciproc apare când roboții își așteaptă resurse ce nu pot fi eliberate; prevenirea se face prin rezervare ordonată, priorități și zone cu sens unic [210]. Alocarea misiunilor ia

în calculul distanța, nivelul bateriei, prioritatea și congestia [211]. Învățarea prin întărire poate sprijini rutarea, dar deciziile ei trebuie filtrate prin reguli verificate, fără a interveni asupra opririi de urgență sau a câmpurilor de protecție.

ISO 3691-4:2023 stabilește cerințe de siguranță pentru vehiculele industriale fără conducător, acoperind atât funcțiile de navigație și percepție, cât și integrarea cu mediul de lucru și cu personalul din incintă [212]. Siguranța decurge din îmbinarea percepției, localizării, distanței de oprire, sarcinii și configurației spațiului [212]. Analiza include oamenii, stivuitoarele, colțurile fără vizibilitate, încărcăturile instabile și pierderea localizării. La pierderea managerului de flotă, AMR-ul păstrează local funcțiile de protecție și se oprește într-o stare sigură. Pentru stațiile de încărcare se confirmă alinierea, contactul electric și temperatura.

Integrarea cu MES și WMS separă misiunea logistică de controlul local. Pierderea WMS poate suspenda generarea de noi misiuni, fără a dezactiva siguranța. Performanța se măsoară la nivelul întregului lanț: timpul de preluare, transport, predare și recuperare. Disponibilitatea include pierderea localizării, blocajele și intervențiile manuale. O optimizare care reduce timpul mediu dar sporește intervențiile poate scădea productivitatea reală.

Recuperarea pornește de la starea fizică observată. Pentru celulă se verifică postura robotului, unealta, etapa secvenței și sincronizarea cu PLC-ul. Pentru AMR se verifică poziția estimată, încărcătura, versiunea hărții și misiunea activă. Reluarea nu inițiază automat mișcarea când poziția persoanelor sau a vehiculului nu este cunoscută cu suficientă certitudine. Principiile generale ale recuperării sunt prezentate în secțiunea 8.2.

Activele AMR-ului administrate împreună sunt harta, calibrarea senzorilor, parametrii de localizare, configurația vehiculului și politicile de alocare. Pentru celule se folosește configurația operațională din secțiunea 9.4. Orice modificare se asociază cu echipamentul fizic și cu domeniul validat. Administrarea modelelor este tratată în secțiunea 5.4; securitatea și revenirea la versiunea anterioară, în capitolul 8.

Acceptarea industrială a unei aplicații robotice înseamnă executarea verificabilă a operației, confirmarea efectului fizic și capacitatea de a recupera controlat atunci când starea sistemului devine incertă.

10. MES, ERP și integrarea operațională

Integrarea verticală urmărește coerența dintre intențiile de producție, execuția fizică și tranzacțiile întreprinderii. Conectarea tehnică a aplicațiilor este utilă doar când stările, identificatorii, versiunile și responsabilitățile pot fi corelate și verificate.

Un ordin poate apărea ca „lansat” în sistemul firmei și ca „în lucru” în sistemul de execuție, în timp ce mașina este oprită de o alarmă. Aceste informații nu sunt neapărat greșite: fiecare descrie alt pas și poate fi actualizată la alt moment. Capitolul arată cine autorizează pornirea, cine cunoaște starea fizică a utilajului și cine confirmă piesele produse și calitatea lor. Când sistemele au temporar informații diferite, discrepanța trebuie detectată și rezolvată după reguli explicite, nu corectată fără urmă. Rolurile sunt detaliate în secțiunea 10.1 [7], [213].

10.1 Rolurile PLC–SCADA–MES–ERP și autoritatea fiecărui nivel

Planificarea resurselor întreprinderii (ERP) integrează procesele organizaționale prin date de bază și tranzacții comune — comenzi comerciale, planificare, achiziții, inventar, costuri și finanțe. Valoarea sa derivă din standardizarea proceselor, a rolurilor și a responsabilităților: recepția unui material modifică inventarul și poate genera obligații către furnizor; confirmarea producției generează consumuri, stocuri și costuri [214], [215].

Separarea responsabilităților previne conflictele de interese. Aceeași persoană nu trebuie să creeze, să aprobe și să finalizeze o tranzacție cu risc ridicat. În producție, aceasta poate viza crearea unui material, modificarea listei de materiale, lansarea ordinului și aprobarea derogărilor. Rolurile se revizuiesc periodic, deoarece utilizatorii acumulează permisiuni pe măsură ce își schimbă responsabilitățile.

ISA-95 descrie activitățile, modelele și interfețele dintre controlul procesului, managementul operațiunilor de fabricație și nivelul întreprinderii. Standardul nu impune ca fiecare funcție să fie implementată într-un produs separat: o platformă SCADA poate include raportare și trasabilitate, iar un MES poate integra funcții de calitate sau mentenanță. Important este ca funcțiile, responsabilii și interfețele să fie definite, nu presupuse din eticheta furnizorului [7], [213]. Tabelul 10.1 sintetizează responsabilitățile orientative și autoritatea asociată fiecărui nivel.

Autoritatea urmează scara temporală și consecințele unei erori. ERP poate crea și elibera un ordin, dar nu scrie valori de referință direct într-un PLC. MES poate selecta operația și rețeta, însă controlerul verifică modul de operare, starea echipamentului și interblocările. SCADA poate solicita o comandă, dar acceptarea și executarea rămân la nivelul controlului local.

Fiecare categorie de informație are un responsabil și un sistem de referință. Definiția comercială a produsului aparține ERP-ului; definiția tehnică și reviziile, sistemului PLM („*Product Lifecycle Management*”). Rețeta principală aprobată aparține MES-ului, iar valorile efectiv utilizate sunt confirmate de echipament. Rezultatul ciclului este generat la nivelul mașinii și contextualizat de MES; tranzacția financiară este administrată de ERP.

În practică, nu există o singură „sursă unică de adevăr”, ci o rețea de surse de referință, fiecare pentru domeniul propriu. Două aplicații nu modifică independent aceeași informație fără atribuirea responsabilității, reguli de reconciliere și controlul versiunilor. Replicarea unei valori în scop de performanță nu transferă automat și autoritatea asupra ei.

Relația ERP–MES separă coerența tranzacțională de cea operațională. ERP creează sau eliberează ordinul; MES îl verifică și îl transformă în operații, transmitând progresul, consumurile și

neconformitățile. ERP nu declară ordinul finalizat doar pentru că termenul planificat a expirat; închiderea se bazează pe confirmarea execuției și pe regulile de calitate [214], [215].

Tabelul 10.1. Responsabilități orientative în integrarea verticală

Nivel sau sistem	Responsabilitate principală	Autoritate obișnuită
PLC, DCS și unități de acționare	Controlul fizic determinist, secvențele și interblocările	Comenzi directe și acționare asupra procesului
SCADA și HMI	Supraveghere, alarmare, vizualizare și rețete de lucru	Comenzi de supraveghere validate de PLC
MES	Coordonarea, urmărirea și înregistrarea execuției	Lansarea ordinelor și a rețetelor aprobate
ERP	Planificarea resurselor, bilanț financiar, comenzi clienți	Eliberarea ordinelor la nivel de întreprindere
PLM	Definiția tehnică, desenele și reviziile produsului	Structuri de produs (BOM) și proceduri aprobate
WMS	Gestiunea depozitului, stocurilor și mișcării materialelor	Misiuni logistice și confirmări privind stocul

Datele de bază formează vocabularul comun: produs, material, echipament, traseu tehnologic, listă de materiale, rețetă, unitate de măsură, calendar. Un identificator nealiniat face ca mesajele să fie corecte sintactic și inutilizabile semantic. Pentru fiecare categorie se stabilește cine creează, aprobă, administrează versiunile și distribuie informația. Un ordin lansat cu versiunea 3 a rețetei nu se execută automat cu versiunea 4 doar pentru că aceasta a devenit activă.

Conversiile de unități sunt explicite și versionate. ERP poate lucra în bucăți și kilograme; procesul, în grame, litri sau impulsuri. Marcajele temporale includ fusul orar și tratează schimbarea sezonieră a orei. Pentru evenimentele importante se păstrează atât momentul producerii la sursă, cât și momentul recepției.

10.2 Execuția producției: ordine, producție în curs, rețete și resurse

În timp ce nivelul ERP gestionează comenzile comerciale la o scară de timp agregată (zile sau săptămâni), transformarea acestor planuri în produse fizice devine responsabilitatea directă a sistemului MES (*Manufacturing Execution System*). MES coordonează și înregistrează în timp real activitățile de fabricație din atelier, preluând comanda lansată din ERP și transformând-o într-o succesiune fină de operații executabile pe minute și secunde.

Pentru fiecare ordin de producție, MES-ul corelează rețeta de fabricație, materiile prime, sculele, utilajele disponibile și operatorii calificați conform standardului ISA-88. Pe durata fabricației, sistemul urmărește progresul, consumurile reale, stările utilajelor și parametrii de calitate, transmițând la final confirmările tranzacționale către ERP [216], [217].

Cu toate acestea, MES-ul nu înlocuiește bucla de control deterministă de la nivelul mașinii: el solicită inițierea unei etape tehnologice, însă PLC-ul local verifică interblocările fizice, semnalele senzorilor și condițiile de securitate funcțională. Confirmarea de execuție înregistrată în MES descrie ceea ce s-a produs efectiv în mediul fizic, nu doar simplul fapt că o comandă a fost expedită către utilaj.

Sistemul diferențiază cantitatea planificată, lansată, începută, finalizată, conformă, respinsă și confirmată. O operație începută nu este automat completă; o piesă produsă fizic poate rămâne blocată de o verificare de calitate. Aceste diferențe se regăsesc în mașinile de stare și în mesajele de integrare. Tabelul 10.2 definește stările distincte ale execuției în MES și ERP, iar Figura 10.1 ilustrează ciclul de viață al ordinului și autoritatea pe niveluri.

Tabelul 10.2. Stări distincte ale execuției

Stare	Semnificație
Planificat	Ordinul există în planificare, dar nu este autorizat pentru lansare
Eliberat	Ordinul a fost verificat și poate fi programat pe utilaje
Repartizat	Operația a fost atribuită unei resurse și unui echipaj
Început	Prelucrarea fizică pe mașină a fost inițiată
Finalizat parțial	O parte din cantitatea comandată a traversat operația
Finalizat fizic	Producția fizică declarată s-a încheiat complet
Blocat pentru verificarea calității	Lotul este carantinat până la emiterea deciziei de calitate
Confirmat	Cantitățile conforme și consumurile au fost validate și transmise
Închis	Ordinul a fost reconciliat operațional și decontat financiar
Anulat	Execuția a fost oprită definitiv printr-o acțiune autorizată

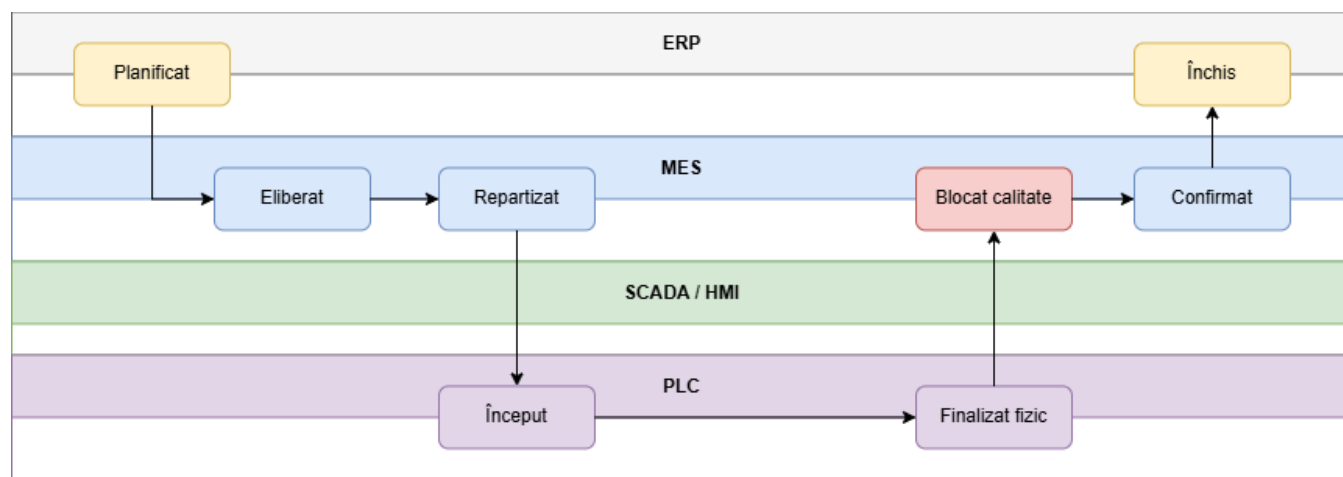


Figura 10.1 Ciclul de viață al unui ordin de producție și autoritatea pe niveluri

Trecerea unui ordin prin stările Planificat → Eliberat → Repartizat → Început → Finalizat fizic → Confirmat → Închis implică niveluri diferite de autoritate (Figura 10.1). Responsabilitatea fiecărei tranziții aparține sistemului care deține autoritatea asupra acelei etape: ERP pentru planificare, MES pentru coordonare și calitate, PLC pentru execuția fizică.

Planificarea sarcinilor transformă planul general într-o succesiune realizabilă, ținând seama de starea echipamentelor, materialele eliberate, personalul calificat, scule, rețete, timpii de pregătire și restricțiile de calitate. O resursă nu este potrivită doar pentru că este liberă: poate fi funcțională, dar neautorizată pentru un anumit produs din lipsa calibrării sau a programului aprobat. Resursele — echipamente, operatori, materiale, scule, utilități și componente software — au stări complexe (în producție, în pregătire, în curățare, în mentenanță, în calibrare, rezervate sau blocate pentru calitate), iar MES diferențiază disponibilitatea tehnică de permisiunea operațională.

Planificarea detaliată optimizează termenele, capacitatea, schimbările de produs, consumul energetic sau mentenanța. Rezultatul matematic se verifică în raport cu restricțiile reale. Replanificarea prea frecventă poate reduce predictibilitatea fabricii, chiar dacă optimizează un indicator local. Produsele în lucru și promisiunile transmise altor sisteme se respectă sau se renegociază explicit.

Producția în curs (WIP - *Work in Progress*) cuprinde materialele și produsele dintre începutul și sfârșitul fluxului. Sistemul cunoaște locația, identitatea, operațiile executate, rezultatele și pasul următor permis. Erorile legate de WIP pot duce la omiterea sau repetarea unei operații ori la procesarea cu o versiune greșită. De aceea, instrucțiunile electronice trebuie să corespundă exact produsului, operației și reviziei active. Simpla afișare nu certifică executarea; sistemul diferențiază accesarea documentului, confirmarea pasului și verificarea rezultatului.

Rețeta conține parametrii, secvențele și limitele necesare. Se disting rețeta principală, versiunea aprobată, rețeta descărcată, valorile efectiv utilizate și modificările temporare. Calea de scriere către control este autorizată, iar PLC-ul verifică versiunea, domeniul valorilor și interblocările. Modificările temporare se înregistrează cu utilizatorul, motivul și intervalul de valabilitate. Înaintea execuției se verifică compatibilitatea dintre ordin, material, rețetă, programul mașinii și configurația echipamentului.

10.3 Date operaționale, calitate, genealogia produsului și indicatori

Datele operaționale sunt colectate automat — din PLC-uri, SCADA, instrumente, sisteme de viziune — sau introduse manual. Fiecare înregistrare păstrează sursa, marcajul temporal, starea de calitate, unitatea, identificatorul produsului și versiunea configurației. Principiile generale privind calitatea și proveniența au fost prezentate în capitolul 4; în MES, fiecare valoare este asociată cu operația, ordinul și decizia pentru care a fost utilizată [107], [218].

Valorile introduse manual se marchează distinct. Corectarea unei înregistrări păstrează valoarea inițială, autorul, momentul și motivul modificării. Fără această trasabilitate, sistemul afișează o situație coerentă fără a permite reconstituirea modului de obținere. În plus, MES aplică verificări de calitate, blochează produse și inițiază fluxuri pentru neconformități. Se diferențiază defectul detectat, produsul suspect, blocarea pentru verificare, reinspecția, reprelucrarea și acceptarea prin derogare, iar decizia se asociază cu persoana sau rolul care a aprobat-o. Finalizarea fizică nu eliberează automat produsul pentru pasul următor.

Genealogia produsului leagă materialele de intrare de operații, echipamente, operatori, parametri, rezultate și produsul final. Trasabilitatea bidirecțională permite limitarea retragerilor și analiza cauzelor. Modelele 1-la-1 devin incorecte în procesele de amestecare, dozare, tăiere, reprelucrare și asamblare. Identitatea lotului se menține și când materialele sunt divizate, reunite sau substituite. Tabelul 10.3 enumeră transformările pe care sistemul de genealogie trebuie să le susțină.

Timpul de oprire se clasifică după cauză — defect, lipsă material, absență operator, pregătire, curățare, mentenanță, blocare pentru calitate, lipsă cerere — deoarece fiecare situație reclamă o acțiune diferită. Agregarea opririlor tehnice cu cele comerciale produce indicatori greu de interpretat.

Tabelul 10.3. Relații care trebuie susținute de genealogia produsului

Transformare	Exemplu
Un lot către mai multe produse	Debitarea sau dozarea unui lot de materie primă în mai multe repere
Mai multe loturi către un produs	Asamblarea sau amestecarea componentelor din loturi diferite
Consum parțial	Preluarea unei cantități de material dintr-un container standard
Divizare și reunire	Împărțirea unui lot pe linii paralele și reunirea la ambalare
Reprelucrare	Reintroducerea unei piese neconforme într-o operație anterioară
Substituție controlată	Utilizarea unui material alternativ aprobat în lipsa stocului
Reambalare	Modificarea unității logistice fără alterarea produsului fizic
Rebut parțial	Eliminarea pieselor defecte cu păstrarea trasabilității lotului rămas

Eficiența globală a echipamentului (OEE) combină disponibilitatea (A), performanța (P) și calitatea (Q):

$$OEE = A \times P \times Q$$

$$A = \frac{\text{Timp de funcționare}}{\text{Timp de producție planificat}}$$

$$P = \frac{\text{Timp ideal de ciclu} \times \text{Total Piese}}{\text{Timp de funcționare}}$$

$$Q = \frac{\text{Piese bune}}{\text{Total piese}}$$

Rezultatul depinde de definiția timpului planificat, de timpul ideal de ciclu și de clasificarea opririlor. Două fabrici pot raporta OEE (*Overall Equipment Effectiveness*) diferit pentru aceeași performanță fizică dacă una exclude timpii de pregătire. OEE nu se folosește ca obiectiv unic: creșterea vitezei poate îmbunătăți performanța și poate deteriora calitatea sau disponibilitatea [14], [219].

Fiecare indicator-cheie de performanță (KPI) trebuie să aibă o definiție, o formulă, o sursă, o frecvență, un responsabil și un domeniu de utilizare. Definiția se versionează; schimbarea formulei nu se aplică retroactiv fără a semnala modificarea. Scăderea OEE poate fi cauzată de o mentenanță planificată, de un produs cu timp de ciclu mai mare, de o lipsă de material sau de o reducere a cererii. Utilizatorul trebuie să primească factorii contributivi și intervalul în care au fost calculați [15]. Trasabilitatea unui KPI permite urmărirea rezultatului până la măsurările și evenimentele care l-au generat — formulă, versiune, filtre, interval, excluderi și corecții — pentru a putea fi reprodus, comparat sau auditat [218].

10.4 Integrarea orientată pe evenimente, consistența și funcționarea fără conexiune

Arhitectura MES cuprinde conectori către echipamente, servicii de integrare, baze de date operaționale, motoare de fluxuri și reguli operaționale. Conectorii păstrează marcajul temporal, calitatea, unitatea,

identitatea sursei și diferențiază o valoare zero de lipsa informației [216], [217]. Stratul de integrare poate utiliza API-uri, brokeri de mesaje sau procesarea continuă a fluxurilor; comportamentele privind întârzierile, duplicatele și ordonarea au fost prezentate în secțiunea 4.2. În integrarea MES–ERP, mesajele sunt versionate și conțin identificatori stabili, care permit corelarea ordinului, a operației, a produsului și a tranzacției [111].

Un model informațional canonic reduce corespondențele punctuale, definind concepte stabile: produs, ordin, lot, material, echipament, persoană, rețetă, stare. Conceptele comune sunt standardizate; extensiile locale sunt controlate și documentate.

Într-o arhitectură orientată pe evenimente, comenzile și evenimentele au semnificații distincte: „pornirea operației” este o solicitare, „operația a început” este o confirmare. A trata intenția drept fapt poate determina ERP-ul să consume materiale înainte ca operația fizică să fi avut loc. Retransmiterea unei confirmări nu dublează consumul sau cantitatea produsă; aplicarea unică se verifică la nivelul ERP și al întregului lanț.

Sistemele pot ajunge temporar în stări diferite: MES poate considera ordinul finalizat, iar ERP să nu fi primit confirmarea; un produs poate fi blocat în MES și disponibil în inventarul ERP. Reconcilierea identifică și corectează aceste diferențe prin reguli auditabile, fără corecții silențioase. Limitele tranzacțiilor trebuie să fie explicite: o tranzacție ERP poate agrega mai multe evenimente fizice, materialul poate fi consumat treptat și raportat la finalul operației, iar corespondența dintre stările MES și ERP se administrează cu grijă, precizând când devine tranzacția finală și cum se operează corecțiile.

Producția nu se oprește automat când ERP-ul sau o conexiune externă devin indisponibile. MES-ul păstrează local ordinele lansate, rețetele aprobate și datele de bază, continuă funcțiile autorizate și stochează evenimentele pentru sincronizarea ulterioară. Se definește durata maximă de funcționare fără conexiune și funcțiile suspendate; producția ordinelor aprobate poate continua, dar lansarea de produse noi este blocată. Funcționarea fără conexiune prevede, de asemenea, limite pentru memoria tampon, marcarea modului degradat și o procedură de reconciliere. Evenimentele nu se aplică mecanic în ordinea sosirii dacă între timp s-au modificat ordinele sau permisiunile, iar conflictele se rezolvă pe baza identității tranzacției, a momentului producerii și a sistemului de referință.

Analiza de disponibilitate pentru MES identifică dependențele critice — baza de date, serviciul de identitate, brokerul de mesaje, certificatele — iar mecanismele generale de redundanță și recuperare sunt prezentate în secțiunea 8.2. La nivel arhitectural, microserviciile pot separa gestiunea ordinelor, calitatea, genealogia și raportarea, permițând scalare independentă, dar introduc consistență distribuită și complexitate operațională sporită. Un sistem monolitic bine modularizat poate fi mai ușor de operat; alegerea se face în funcție de nevoia reală de autonomie, nu urmărind tendințele din industria software.

Calea de scriere MES–PLC se protejează mai strict decât colectarea de date. Interfața permite numai obiectele și operațiile aprobate — identificatorul ordinului, produsul, referința rețetei, cantitatea țintă — și păstrează identitatea sursei și rezultatul validării locale. Regulile privind autoritatea nivelurilor și verificarea rețetelor sunt prezentate în secțiunile 10.1–10.2.

10.5 ERP, PLM, WMS și opțiuni de implementare

ERP, PLM, MES și WMS privesc aceeași întreprindere din unghiuri diferite [215]: ERP de planificare și tranzacții, PLM de definiția produsului și schimbările tehnice, MES de coordonarea execuției, WMS de operațiile depozitului. Integrarea păstrează coerența identificatorilor, listelor de materiale, traseelor, rețetelor, echipamentelor, loturilor și reviziilor, fără a anula specializarea fiecărei aplicații.

În practică, fiecare sistem deține o parte a adevărului operațional. PLM-ul este sursa de referință pentru definiția produsului (specificații, desene, liste de materiale), ERP-ul pentru datele financiare și de achiziții, MES-ul pentru execuția și trasabilitatea producției, iar WMS-ul pentru stocurile fizice și mișcările din depozit. Integrarea nu presupune replicarea tuturor datelor în fiecare sistem, ci stabilirea unor fluxuri controlate prin care fiecare aplicație preia din sursa autorizată doar informația de care are nevoie, în formatul și la momentul potrivit. Când aceste responsabilități se suprapun — de exemplu, atât ERP-ul cât și MES-ul gestionează stocuri — regulile de reconciliere trebuie să fie explicite, iar sistemul care deține imaginea fizică (de regulă WMS-ul sau MES-ul) are prioritate în caz de neconcordanță.

Sistemele on-premise oferă control asupra infrastructurii; SaaS (*Software as aService*) aduce scalare și actualizări frecvente, dar introduce dependențe de furnizor, conectivitate și contracte de date. Arhitecturile hibride păstrează local funcțiile operaționale și migrează treptat planificarea sau analiza. Alegerea ia în calcul indisponibilitatea, recuperarea, rezidența datelor și durata suportului, nu doar costul inițial.

În mediile industriale, trecerea la SaaS implică și o reevaluare a arhitecturii de securitate: datele de producție părăsesc perimetrul fabricii, actualizările automate pot modifica funcționalități critice fără un ciclu de validare controlat, iar dependența de conexiunea la internet poate intra în conflict cu cerințele de autonomie locală discutate în secțiunea 2.1. Din aceste motive, sistemele care deservește direct execuția (MES, SCADA) sunt adesea păstrate on-premise sau într-o arhitectură hibridă, în timp ce planificarea și analiza avansată migrează mai ușor către cloud.

Un exemplu frecvent de risc operațional apare atunci când o modificare de produs introdusă în PLM (de exemplu, o nouă revizie a unei componente) nu se propagă corect în MES și ERP. Operatorul din producție vede instrucțiunea veche, achizițiile comandă materialul după specificația depășită, iar depozitul recepționează piesa nouă sub un cod necunoscut. Impactul nu este doar informațional, ci direct asupra calității, termenelor și costurilor.

Personalizarea excesivă duce la actualizări dificile și interfețe fragile. Sistemele de întreprindere impun procese comune; o particularitate se păstrează doar când răspunde unei cerințe operaționale, tehnice sau legale, nu pentru că așa s-a lucrat dintotdeauna [215]. O integrare bine proiectată se recunoaște prin faptul că fiecare particularitate păstrată are o justificare explicită, iar fluxurile între sisteme pot fi auditate și reproduse.

10.6 Sprijinul decizional și urmărirea rezultatului

Un sistem de sprijin decizional (DSS-*Decision Support System*) combină date, modele și interacțiunea umană pentru a compara alternative. Un tablou de bord descrie starea; un DSS analizează opțiunile, consecințele, ipotezele și incertitudinile, folosind date istorice, simulare, prognoză, optimizare și inteligență artificială [220].

Recomandarea își declară ipotezele. O propunere de reducere a costurilor nu este credibilă dacă presupune timpi de pregătire nerealist de mici sau materiale indisponibile. Decizia industrială implică frecvent compromisuri între cost, termen, energie, calitate și risc. Analiza multi-obiectiv oferă alternative Pareto, fără să impună o singură soluție [221], [222], [223] ; algoritmi precum SMS-EMOA utilizează măsura hipervolumului pentru un front bine distribuit [224]. Lanțul informațional al sprijinului decizional industrial este ilustrat în Figura 10.2.



Figura 10.2. Lanțul informațional al sprijinului decizional industrial

La fiecare etapă se adaugă interpretare și se poate introduce eroare: senzorul observă starea fizică, PLC-ul o integrează, MES-ul o asociază cu produsul, ERP-ul o convertește în tranzacție, iar DSS-ul o combină pentru a evalua o acțiune. Responsabilitatea și sursa de referință se păstrează pe tot parcursul.

Recomandarea este urmărită și după execuție: sistemul reține modelul, utilizatorul, decizia, justificarea, acțiunea și rezultatul. Compararea rezultatului cu predicția permite evaluarea DSS-ului și îmbunătățirea modelelor. Fără această urmărire, sistemul rămâne un raport avansat, nu o buclă decizională închisă.

Gemenii digitali utilizează informațiile MES și ERP pentru simularea planurilor și evaluarea capacității; teoria generală a fost prezentată în capitolul 7. În integrarea operațională, modelul se bazează pe starea reală a resurselor — o simulare cu echipamente disponibile doar nominal poate recomanda un plan imposibil [160]. Totodată, inteligența artificială sprijină prognoza, detectarea anomaliilor tranzacționale și planificarea [225], [226], fără a eluda aprobările și integritatea tranzacțiilor. O recomandare de modificare a programului se verifică în raport cu materialele, mentenanța, calitatea și permisiunile, iar autoritatea decizională rămâne distinctă de acuratețea statistică.

10.7 Implementarea și schimbarea organizațională

Implementarea unui ERP sau MES este un program de transformare, nu doar un proiect software. Problemele frecvente includ date de bază incorecte, procese neclare, responsabilități nedefinite și instruire insuficientă. Procesul se definește înainte ca aplicația să fie configurată [215], [227].

Migrarea nu este o copie fidelă a vechiului sistem. Datele se selectează, se curăță și se validează; duplicatele, identificatorii depășiți și unitățile incompatibile se rezolvă înainte de migrare, altfel erorile se propagă mai rapid și mai consecvent.

Testarea de acceptare acoperă întregul proces — de la crearea ordinului până la reconciliere. Scenariile tehnice privind duplicatele, întârzierile și funcționarea fără conexiune sunt definite în secțiunea 10.4; în implementare se verifică responsabilitățile, procedurile și capacitatea utilizatorilor de a le aplica. Trecerea la noul sistem definește oprirea sistemului vechi, migrarea finală, verificarea soldurilor, activarea interfețelor și posibilitatea revenirii. Produsele nu se transferă între sisteme fără o corespondență verificată a stărilor, operațiilor și deciziilor de calitate.

Instruirea explică procesul și responsabilitatea, nu doar interfața. Un operator care cunoaște succesiunea butoanelor fără să înțeleagă semnificația confirmării poate introduce date formal corecte și operațional greșite. Beneficiile precum reducerea stocurilor, creșterea preciziei planificării, scurtarea timpilor, îmbunătățirea datelor de bază, se măsoară după stabilizare, nu la lansare. Tehnologia produce valoare prin îmbinarea proceselor, competențelor, guvernantei și schimbării organizaționale [228], [229].

Factorii critici de succes în astfel de proiecte includ: sprijinul vizibil și constant al conducerii, desemnarea unor responsabili de proces din rândul beneficiarilor (nu doar din IT), alocarea de timp pentru instruire și validare în programul de lucru curent, precum și definirea unor indicatori de succes măsurabili. Riscurile frecvente includ: subestimarea efortului de curățare a datelor, personalizarea excesivă a pachetului standard, rezistența la schimbare și lipsa de comunicare între echipele de

implementare și utilizatorii finali. Lecțiile învățate din proiecte de anvergură arată că o abordare iterativă, cu livrări incrementale și bucle scurte de feedback, produce rezultate mai bune decât o lansare de tip „big-bang”, iar implicarea utilizatorilor-cheie încă din faza de proiectare crește atât calitatea soluției, cât și acceptanța organizațională.

ERP și MES administrează active informaționale cu impact direct: date de bază, ordine, rețete, genealogie, confirmări, conturi de integrare, reguli de reconciliere. Compromiterea ERP poate genera tranzacții false; compromiterea MES poate modifica rețete și stări de producție. Cerințele generale de securitate sunt tratate în capitolul 8 [3], iar conturile de serviciu primesc doar drepturile necesare; se jurnalizează modificările datelor de bază, ale rețetelor, ale rolurilor și ale confirmărilor, păstrând identitatea, valoarea anterioară, valoarea nouă și justificarea. Planul de continuitate atribuie responsabilitatea pentru activarea modului fără conexiune, aprobarea operațiilor care pot continua și validarea reconcilierii, comportamentul tehnic fiind definit în secțiunile 10.4, 8.2 și 8.3.

Datele MES și ERP pot asocia energia, apa, materialele și rebuturile cu ordinul sau lotul, susținând analiza costurilor și a sustenabilității, cu condiția ca măsurările și regulile de alocare să fie versionate și trasabile (măsurarea resurselor la nivelul fabricii este dezvoltată în secțiunea 11.3). Rezultatul distinct al integrării operaționale este o legătură verificabilă între intenția autorizată, execuția fizică confirmată și efectul tranzacțional înregistrat — însoțită de reguli de reconciliere pentru situațiile în care aceste stări nu coincid.

11. Fabrica inteligentă, sustenabilitate și arhitecturi industriale emergente

Fabrica inteligentă nu se definește prin numărul de senzori sau prin utilizarea izolată a inteligenței artificiale, ci prin capacitatea organizației de a observa coerent procesele, de a explica abaterile, de a anticipa evoluția probabilă și de a aplica schimbări controlate ale căror rezultate sunt măsurate și utilizate pentru îmbunătățire continuă. Maturitatea apare când procesul fizic, datele, controlul, aplicațiile operaționale și responsabilitatea umană formează un sistem verificabil [230], [231].

O fabrică poate avea multe mașini automatizate și totuși să nu poată răspunde la întrebări simple: câte piese bune au rezultat, din ce lot provin și câtă energie a cerut fiecare produs? Datele pot exista, dar pot fi înregistrate cu denumiri, unități sau stări incompatibile. De aceea, conectarea utilajelor nu este suficientă: trebuie să știm la ce echipament și produs se referă fiecare valoare, cât de sigură este și cine poate acționa pe baza ei. Capitolul reunește arhitectura fabricii, descrierea activelor, schimbul de date și măsurarea resurselor. Apoi tratează administrarea aplicațiilor locale și folosirea inteligenței artificiale în proiectare. Gemenii digitali sunt prezentați în capitolul 7, iar securitatea și autoritatea umană în capitolul 8.

11.1 Arhitectura integrată și perspectivele de proiectare

Arhitectura integrată poate fi descrisă prin planuri complementare, fiecare cu propriul rol, autoritate și cerințe [231] : planul fizic descrie procesele și limitele reale; planul de măsurare și acționare realizează legătura cu instalația; planul controlului determinist gestionează reacția imediată; planurile de conectivitate, procesare, date, operațiuni și guvernare susțin coordonarea și utilizarea informației. Niciun plan nu este suficient izolat: rezultatul depinde de interfețele dintre ele (Tabelul 11.1).

Valoarea acestei descompunerii constă în definirea relațiilor dintre planuri. O măsurare își păstrează unitatea, marcajul temporal și starea de calitate traversând nivelurile sistemului; o recomandare calculată central este validată local; modificarea unui program sau a unei rețete se propagă prin versiuni compatibile. Pentru fiecare interfață se definesc informația transferată, sursa de referință, autoritatea de scriere, comportamentul la indisponibilitate și metoda de verificare a efectului.

ISO/IEC/IEEE 42010:2022 organizează descrierea arhitecturii prin părți interesate, preocupări și puncte de vedere. Cadrul NIST pentru sisteme cyber-fizice evidențiază comportamentul temporal, datele, oamenii, încrederea, compoziția și ciclul de viață [10], [232]. Perspectiva funcțională prezintă funcțiile; cea informațională, entitățile și semnificația lor; cea a comunicațiilor, fluxurile și proprietățile temporale; cea a distribuției, platformele de execuție; cea a ciclului de viață acoperă versionarea, punerea în funcțiune, actualizarea și retragerea. Limita sistemului include toate componentele care pot modifica rezultatul.

Modelele RAMI 4.0, IIRA [233] și ISO/IEC 30141 oferă vocabulare utile, dar nu sunt proiecte complete. RAMI 4.0 rămâne relevant conceptual pentru separarea straturilor și a ciclului de viață [234]; IIRA (*Industrial Internet Reference Architecture*) corelează perspectivele de afaceri, funcțională și de implementare; ISO/IEC 30141:2024 oferă o arhitectură de referință pentru IoT, care în industrie trebuie completată cu cerințe de timp real și siguranță [235]. Concepte similare de clădiri inteligente integrează senzori și automatizare pentru optimizarea consumurilor [236].

Evaluarea arhitecturii se face prin capabilitățile și relațiile dintre planuri, nu printr-un scor unic. Conectivitatea, vizibilitatea, OEE și consumul specific descriu aspecte parțiale; în paralel se verifică calitatea datelor, domeniul de validitate al modelelor, autoritatea de scriere, posibilitatea operatorului de a interveni și capacitatea de restaurare [175], [178], [181], [230], [231].

Tabelul 11.1. Planurile unei arhitecturi industriale integrate

Plan	Rol dominant	Exemple de funcții	Cerință principală
Fizic	Procesul tehnologic și limitele reale	Mașini, materiale, energie, operatori	Modelarea constrângerilor fizice
Măsurare și acționare	Observarea și influențarea procesului	Senzori, actuatoare, semnale de reacție	Calibrare, marcare temporală, fidelitate
Control determinist	Reacție în timp real predictibilă	PLC, vPLC, bucle rapide, safety	Respectare deadline, comportament la defect
Conectivitate	Transportul securizat al datelor	OPC UA, MQTT, TSN, magistrale de câmp	Latență controlată, integritate, segmentare
Edge industrial și fabrică	Coordonare locală și analitică rapidă	Gateway-uri edge, viziune, baze time-series	Procesare locală, autonomie la deconectare
Date și platforme centrale	Agregare istorică și optimizare globală	Depozite de date, antrenare AI, IDSA	Scalabilitate, guvernanță, securitate
Operațional și la nivelul întreprinderii	Planificare și execuție economică	MES, ERP, WMS, genealogie, OEE	Coerență tranzacțională, trasabilitate
Uman și guvernanță	Supraveghere și decizie centrată pe om	HMI, managementul alarmelor, etică AI	Autoritate umană, ergonomie cognitivă

11.2 Asset Administration Shell, firul digital și spațiile de date industriale

Interoperabilitatea presupune o reprezentare coerentă a activelor, nu doar protocoale comune. Asset Administration Shell (AAS), standardizată prin IEC 63278-1:2023, oferă o structură pentru reprezentarea activelor fizice, digitale sau intangibile, incluzând identitatea, proprietățile, capabilitățile, documentația și submodelele [237]. AAS nu este echivalentă cu un geamăn digital complet; rolul ei este de a furniza o interfață informațională standardizată, permițând aplicațiilor să descopere și să interpreteze uniform informațiile despre un activ.

Într-o implementare industrială, AAS poate reuni identificatorul activului, datele tehnice, starea de mentenanță, parametrii energetici și relațiile cu alte componente. Submodelele se administrează pe durata ciclului de viață și se leagă de sursele de referință. Copierea aceleiași proprietăți în mai multe submodele, fără reguli clare de responsabilitate, reproduce problema replicării necontrolate între MES, ERP și PLM. O proprietate trebuie însoțită de unitate, semantică, versiune și stare de actualitate; serviciile de scriere se separă de accesul informativ.

Firul digital, definit în secțiunea 7.5, asigură continuitatea informațională pentru a corela AAS cu versiunile programelor, rețetelor și modelelor. Managementul modificărilor menține reprezentarea sincronizată cu activul; un model neactualizat poate descrie convingător o configurație care nu mai există [157], [160].

Spațiile de date industriale extind interoperabilitatea dincolo de granițele unei fabrici, urmărind schimbul controlat de date între producători, furnizori și clienți, păstrând identitatea participanților, politicile de utilizare, semantica și proveniența [238], [239], [240]. Ele pot susține schimbul de date privind trasabilitatea, amprenta de carbon și capabilitățile de producție, fără o bază de date centrală.

Concepte similare au fost explorate în agricultură pentru optimizarea lanțurilor de aprovizionare [241], [242].

Schimbul controlat de informații despre capabilitățile de producție stă la baza fabricației ca serviciu, prin care organizațiile publică și solicită servicii de producție, fără acces direct la controlul echipamentelor [160], [243], [244]. Paradigma cloud manufacturing completează această viziune [245]. Un spațiu de date presupune modele informaționale compatibile, identități organizaționale și tehnice, reguli de acces și mecanisme de audit. Datele pot fi partajate pentru un scop precis, pe o perioadă și cu un nivel de agregare definite; furnizorul poate retrage accesul, iar consumatorul poate demonstra proveniența informației.

AAS, OPC UA, MES–ERP și spațiile de date tratează niveluri diferite ale interoperabilității. OPC UA oferă servicii și modele informaționale [30]; AAS organizează reprezentarea activului [237]; MES și ERP păstrează contextul operațional; spațiul de date guvernează schimbul dintre organizații [238]. O arhitectură poate folosi toate aceste mecanisme fără suprapunere. Esențial este să se stabilească sursa de referință: o proprietate tehnică poate proveni din AAS sau PLM, starea curentă din PLC, istoricul din arhiva de proces, ordinul din MES, iar costul din ERP.

Spațiile de date pot susține analize distribuite, inclusiv învățarea federată (secțiunea 3.8), caz în care acordurile precizează scopul modelului, categoriile de date și responsabilitatea pentru validarea locală [104], [105].

11.3 Măsurarea resurselor și sustenabilitatea operațională

Sustenabilitatea industrială modernă nu poate fi tratată ca o simplă raportare financiară globală, ci trebuie corelată direct cu dinamica procesului fizic și cu fiecare lot de fabricație realizat [246], [247]. Consumul total de energie la nivel de fabrică oferă o imagine contabilă, dar nu explică randamentul scăzut al unui compresor, pierderile din circuitele de răcire sau resursele alocate unui reper specific. Arhitectura de măsurare trebuie proiectată la o granularitate adecvată deciziei operative, monitorizând consumul de energie electrică și termică, apa, aerul comprimat, materiile prime, precum și volumul de rebuturi și re prelucrări. Standardul ISO 50002-1:2025 definește metodologia pentru auditurile energetice industriale, iar ISO 50001:2018 structurează sistemul de management și îmbunătățire continuă a performanței energetice [16], [248].

Măsurarea resurselor se organizează ierarhic, coborând de la punctul de intrare în fabrică către utilități, linii de fabricație și consumatori majori (SEU – *Significant Energy Uses*). Un sistem complet realizează bilanțul continuu al fluxurilor: energia totală absorbită la intrare este comparată cu suma consumurilor contorizate pe fiecare utilaj plus pierderile tehnologice estimate:

$$E_{\text{intrare}} \approx \sum_{i=1}^n E_i + E_{\text{pierderi}}$$

Diferențele persistente nespecificate scot la iveală ramuri necontorizate, erori de scalare a traductoarelor sau pierderi parazite ascunse (cum sunt scurgerile din rețelele de aer comprimat). În proiectare, trebuie făcută o distincție clară între energie (mărime integrală, cumulativă) și putere (mărime instantanee): optimizarea sarcinii de vârf (*peak shaving*) reduce factura la putere absorbită, dar nu diminuează automat consumul total de energie dacă procesul este doar decalat în timp.

Contoarele industriale inteligente se asociază în sistem cu metadate esențiale: locația fizică, clasa de precizie, factorii de scalare și buletinele de calibrare periodică [107]. Pentru ca datele să aibă valoare

operațională, consumul energetic este contextualizat prin corelarea cu starea utilajului (mers în gol, accelerare, regim nominal), codul de produs, rețeta și cantitatea de piese conforme obținută.

Pentru evaluarea și compararea obiectivă a performanței energetice, standardul ISO 50006:2023 [249] recomandă utilizarea a doi indicatori fundamentali:

1. Consumul specific de energie (*SEC – Specific Energy Consumption*): definit ca raportul dintre energia totală consumată (E) și volumul de producție conformă realizat (Q_{bun}):

$$SEC = \frac{E}{Q_{\text{bun}}}$$

Deși *SEC* este un indicator mult mai relevant decât consumul brut, el trebuie interpretat cu atenție atunci când pe aceeași linie se fabrică produse cu complexități sau mase diferite.

2. Linia de referință energetică (*Energy Baseline – EnB*): un model statistic multi-variabil (bazat pe regresie liniară sau neliniară) care estimează consumul de bază așteptat în funcție de factorii de influență relevanți: volumul de fabricație (Q), temperatura exterioară (T_a), umiditatea (H) și mixul de producție (M):

$$EnB = \beta_0 + \beta_1 Q + \beta_2 T_a + \beta_3 H + \beta_4 M$$

Economia reală de energie (S) obținută în urma unei intervenții de modernizare se evaluează riguros conform ISO 50015:2014 [250], ca diferență dintre consumul prognozat de linia de referință și consumul efectiv înregistrat după implementare:

$$S = EnB - E_{\text{actual}}$$

Managementul energetic devine eficient atunci când informația este integrată direct în bucla decizională: la nivelul mașinii se optimizează automat trecerea în modul *standby* la pauze scurte; la nivelul liniei se sincronizează buffer-ele pentru a evita pornirile și opririle bruște ale motoarelor de mare putere; la nivelul fabricii, planificatorul MES adaptează orarul de fabricație în funcție de tarifele dinamice ale energiei electrice sau de disponibilitatea surselor regenerabile locale (panouri fotovoltaice, cogenerare) [246]. Planificarea producției în funcție de criterii energetice este o problemă tipică de optimizare multi-obiectiv, unde costul energiei, emisiile de carbon, termenele de livrare și calitatea trebuie echilibrate transparent pe un front de compromis de tip Pareto [27], [221], [224].

Sustenabilitatea completă a fabricii inteligente depășește însă consumul strict de energie, integrând managementul materiilor prime, consumul de apă, recuperarea căldurii reziduale și potențialul de remanufacturare. O decizie de proces care reduce consumul electric dar crește rata de rebuturi transferă impactul negativ în amonte, risipind resurse materiale. Sistemele de genealogie din MES susțin direct principiile economiei circulare: standardele ISO 59004:2024 (terminologie și principii) și ISO 59020:2024 (măsurarea circularității) oferă cadrul de evaluare a gradului de reutilizare a resurselor, completate de metodologia de Analiză a Ciclului de Viață (LCA) reglementată de ISO 14040 și ISO 14044 [251], [252], [253], [254].

În final, ecuația de sustenabilitate trebuie să includă și amprenta ecologică a infrastructurii digitale (*Green IT/OT*). Rețelele de senzori, serverele industriale, unitățile de calcul edge și acceleratoarele GPU generează un consum energetic propriu și necesită materiale critice. Stocarea nediferențiată a terabiților de date brute de înaltă frecvență sporește costurile și amprenta de carbon fără a genera valoare decizională. Fiecare nod de achiziție, model de inteligență artificială și flux de transfer în cloud trebuie justificat printr-o cerință funcțională explicită, realizând un compromis

conștient între densitatea datelor, viteza de reacție și impactul energetic al infrastructurii de calcul [3], [11].

11.4 Automatizarea definită prin software și administrarea infrastructurii industriale edge

Automatizarea definită prin software (*Software-Defined Automation* – SDA) urmărește separarea controlată a funcției logice de o platformă fizică fixă, utilizând mecanisme standardizate pentru distribuire, configurare și administrare: PLC-uri virtuale, aplicații containerizate, infrastructură declarativă și orchestrare. În timp ce modelul tradițional IEC 61131-3 este centrat pe execuția ciclică a sarcinilor pe un singur controler dedicat, standardul IEC 61499 redefinește arhitectura de control prin blocuri funcționale declanșate de evenimente (*event-driven*) [255]. Această abordare permite distribuirea nativă a logicii pe noduri eterogene de calcul, decuplând complet codul aplicației de topologia hardware subiacentă și facilitând interoperabilitatea promovată de inițiative industriale deschise (precum *UniversalAutomation.org*) [256]. Controlul industrial nu poate fi însă tratat ca o aplicație IT obișnuită: accesul la I/O, termenele-limită, izolarea resurselor și durata de viață a echipamentelor trebuie demonstrate riguros pe platforma finală de execuție.

Un PLC virtual (vPLC) sau un mediu de execuție bazat pe containere deterministe poate rula pe un server industrial edge numai dacă proprietățile temporale, alocarea nucleelor și accesul determinist la magistralele I/O sunt garantate. Configurarea declarativă descrie starea software dorită și permite versionarea prin mecanisme de tip GitOps; după aplicare se verifică versiunile, conexiunile I/O, latențele de buclă și funcțiile de protecție. Revenirea la o versiune anterioară (*rollback*) restaurează un ansamblu compatibil și verificat — mediu de execuție, aplicație distribuită, parametri și interfețe.

Inițiativa Margo oferă un mecanism comun pentru administrarea sarcinilor de lucru pe dispozitive industriale eterogene. Prima versiune publică (2026) definește elemente pentru distribuirea aplicațiilor și gestionarea ciclului de viață, fără a înlocui orchestratorul sau funcțiile locale de protecție [64]. Unitatea administrată este ansamblul format din platforma de execuție, aplicație, parametri, interfețe de I/O, dispozitive fizice și politica de amplasare. Descrierea declarativă include resursele minime și versiunile compatibile; orchestratorul refuză distribuirea pe un nod incompatibil.

Funcțiile analitice fără stare persistentă se transferă mai ușor decât serviciile care dețin autoritate asupra procesului; pentru acestea, transferul include starea aplicației și compatibilitatea ansamblului. Capacitatea de observare confirmă starea serviciului, accesul la date valide și respectarea timpului de execuție [28], [107]. Cerințele privind inventarul, certificatele, actualizarea securizată și restaurarea sunt prezentate în capitolele 3 și 8 [3], [4], [168].

11.5 Proiectare și configurare tehnică asistate de inteligența artificială și modele fundamentale industriale

Inteligența artificială poate asista proiectarea și configurarea prin generarea sau verificarea codului, transformarea cerințelor în teste, explicarea alarmelor și corelarea evenimentelor. Rolurile se diferențiază după autoritate: un asistent care propune o descriere are un profil de risc diferit de cel al unui sistem care modifică logica PLC. Codul plauzibil nu este automat corect sau sigur; rezultatul generat este supus aceluiași reguli de revizuire, simulare și acceptare ca o modificare manuală. În această categorie, asistenții pentru proiectare pot transforma diagrame și configurații în propuneri de logică sau asocieri OPC UA. Utilitatea crește când lucrează cu biblioteci interne și standarde de denumire aprobate. Rezultatul generat poate conține identificatori inexistenți sau secvențe incompatibile cu starea fizică. Nicio sugestie nu ajunge în producție fără aprobare și verificare pe platforma țintă.

Asistenții pentru operatori pot rezuma alarme și corela evenimente din arhiva de proces, CMMS și MES. Interfața diferențiază măsurările, regulile, inferențele și ipotezele; o corelație nu trebuie prezentată drept cauză confirmată. Recomandarea păstrează starea datelor, incertitudinea și acțiunile permise.

Modelele fundamentale industriale urmăresc reutilizarea unor reprezentări antrenate pe volume mari de date tehnice — texte, imagini, semnale — și adaptarea lor la sarcini multiple: căutare semantică, interpretarea alarmelor, generarea de teste, clasificarea defectelor [257], [258]. Aceste modele sunt emergente și nu elimină nevoia de date locale și validare pe configurația concretă. Guvernanța leagă versiunea modelului de sursele interne și datele de adaptare; principiile de validare și reantrenare sunt definite în secțiunea 5.4, iar autoritatea umană în secțiunea 8.5. Reantrenarea automată necontrolată este inacceptabilă când datele noi pot conține defecte sau atacuri [32], [129].

Autoritatea funcțiilor asistate de inteligență artificială crește gradual, conform Figura 11.1, de la consultare la generarea de propuneri și, în domenii delimitate, la aplicarea automată. Fiecare nivel superior presupune teste pe platforma țintă, protecții independente și o cale de revenire. Rezultatul nu ocolește interblocările, funcțiile de siguranță sau aprobările organizaționale.



Figura 11.1. Creșterea graduală a autorității funcțiilor asistate de inteligență artificială

11.6 Implementarea graduală a fabricii inteligente

Transformarea pornește de la o problemă bine delimitată și o decizie care poate fi îmbunătățită. Colectarea de date înainte de definirea utilizării produce platforme populate cu semnale fără responsabilitate. Înaintea funcțiilor analitice, se rezolvă inventarul activelor, identificatorii, marcasele temporale, unitățile și calitatea datelor. Arhitectura țintă definește limitele de încredere și criteriile de acceptare.

Evoluția sistemului urmează etapele din Figura 11.2. Fiecare etapă generează valoare și reduce riscul următoarei; proiectul-pilot este măsurabil și extensibil prin modele reutilizabile. Interfețele, distribuirea aplicațiilor edge și măsurarea resurselor utilizează șabloane, dar parametrii și competențele se validează local. Tabelul 11.2 propune o foaie de parcurs cu rezultatele urmărite și criteriile de trecere.

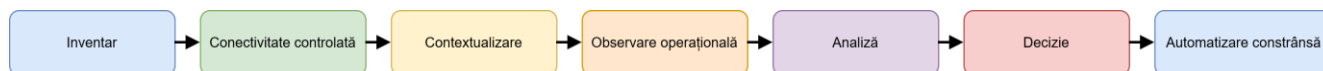


Figura 11.2. Etapele maturizării către automatizarea constrânsă

Extinderea se aprobă când rezultatul are un responsabil, criterii de acceptare și o cale verificată de revenire. Guvernanța reunește producția, automatizarea, mentenanța, IT-ul, finanțele și sustenabilitatea; pentru impact semnificativ, verificarea necesită independență față de echipa de implementare.

Cel mai frecvent mod de eșec este ruperea lanțului care leagă cerința, datele, decizia, acțiunea și verificarea. Un tablou de bord fără flux operațional, un model fără domeniu validat sau o optimizare care ignoră efectele asupra calității adaugă complexitate fără valoare. Implementarea se justifică doar

când efectul poate fi măsurat, investigat și restaurat, iar dependențele introduse rămân proporționale cu rezultatul.

Tabelul 11.2. Foaia de parcurs și criteriile de trecere

Etapă	Rezultat urmărit	Criteriu înaintea extinderii
Inventar și delimitare	Active, procese și responsabilități cartografiate	Responsabili stabiliți și limite de sistem aprobate
Conectivitate controlată	Fluxuri de date securizate și segmentate	Protocoale validate și funcționare offline testată
Contextualizare	Date legate de activ, produs și regim	Unități standardizate, marcaje unificate, trasabilitate
Observare operațională	Starea serviciului și a procesului investigabilă	Alarme raționalizate și proceduri de recuperare verificate
Analiză și model	Ipoteze evaluate pe date independente	Domeniu de validitate, incertitudine și derivă monitorizate
Decizie asistată	Recomandările AI intră în fluxul operațional	Responsabilitate umană definită și efecte urmărite
Automatizare constrânsă	Acțiuni automate limitate prin reguli și protecții	Strategie de rezervă (<i>fallback</i>), <i>rollback</i> și acceptare

Sinteza părții teoretice. Principii transversale de proiectare

Această secțiune reunește principiile dezvoltate pe parcursul capitolelor precedente și evidențiază relațiile dintre ele. Ea nu reia tehnologiile și metodele deja expuse, ci oferă un cadru integrator pentru proiectarea și evaluarea sistemelor industriale.

Proiectarea unui sistem informatic industrial pornește de la procesul fizic, de la obiectivele operaționale și de la consecințele unei funcționări incorecte. Alegerea tehnologiei, a arhitecturii de calcul sau a algoritmului nu este punctul de plecare, ci rezultatul analizei procesului, a stărilor sale, a pericolelor, a constrângerilor și a deciziilor care trebuie susținute. O soluție se justifică prin capacitatea de a îmbunătăți un rezultat măsurabil, fără a compromite timpul de răspuns, siguranța, securitatea, disponibilitatea, mentenabilitatea sau responsabilitatea umană [5], [6], [8].

Funcțiile se distribuie între controlere, noduri edge, infrastructura fabricii și sistemele centrale în funcție de latență, volumul și sensibilitatea datelor, disponibilitatea comunicației, resursele necesare și consecințele pierderii serviciului. Controlul imediat, interblocările și protecțiile rămân cât mai aproape de proces, pentru un comportament temporal predictibil și autonomie față de infrastructura externă. Nodurile edge asigură validarea rapidă, filtrarea, agregarea, conversia de protocoale, inferența locală și păstrarea temporară a datelor. Serverele fabricii coordonează mai multe celule și servicii operaționale, iar sistemele cloud și cele de întreprindere se potrivesc analizei multi-locatie, antrenării modelelor, stocării pe termen lung și planificării. Această distribuie nu este rigidă; pierderea nivelurilor superioare nu trebuie însă să anuleze funcțiile locale esențiale și nici să împiedice trecerea procesului într-o stare sigură [2], [11], [12].

Timpul este o componentă a semnificației informației industriale. O măsurare fără marcaj temporal credibil, fără indicarea prospețimii sau fără informații despre calitatea sincronizării poate duce la interpretări greșite. Similar, o comandă executată după expirarea termenului-limită poate avea efecte echivalente cu cele ale unei comenzi incorecte. Evaluarea acoperă întregul lanț temporal, de la achiziția semnalului până la acțiunea fizică. Debitul mediu sau timpul mediu de execuție nu sunt suficiente pentru funcțiile cu consecințe grave; trebuie cunoscute și limitele întârzierii, variația temporală, timpul maxim de execuție și comportamentul la încărcare maximă [2].

Fluxurile industriale de date se proiectează ca sisteme complete — de la sursa informației până la utilizarea rezultatului — nu ca simple înlănțuiri de componente independente. Achiziția, memoria tampon, echipamentul de interconectare, brokerul de mesaje, procesarea continuă, stocarea, extragerea caracteristicilor, modelul analitic și interfața operațională contribuie împreună la calitatea finală. Performanța unui model devine irelevantă dacă datele sosesc după termenul-limită, sunt asociate cu o unitate greșită, provin de la un senzor necalibrat sau nu pot fi reconstituite după o întrerupere. Datele brute, validate, suspecte, estimate, agregate și rezultatele derivate trebuie diferențiate; utilizatorii și aplicațiile trebuie să poată identifica sursa, transformările și condițiile în care informația a fost obținută. Calitatea datelor nu se verifică o singură dată: senzorii derivă, structurile de date se modifică, ceasurile se desincronizează, iar procesul poate intra în regimuri absente din datele inițiale.

Interoperabilitatea se analizează la nivel tehnic, sintactic, semantic și organizațional. Folosirea aceluiași protocol nu garantează că două sisteme atribuie aceeași semnificație valorilor, stărilor și comenzilor. Alături de valoarea numerică trebuie păstrate identitatea activului, unitatea de măsură, marcajul temporal, indicatorul de calitate, versiunea configurației și contextul operațional. Echipamentele de interconectare conservă semnificația informației, nu doar transferă sau convertesc valori. Fiecare categorie de informație are un proprietar și un sistem care constituie sursa de referință; dreptul de a emite o comandă este separat de dreptul de a vizualiza datele. Datele de referință ale

organizației formează vocabularul comun al integrării și se administrează controlat, pentru ca modificarea unui identificator, a unei rețete sau a structurii unui produs să nu genereze interpretări incompatibile între automatizare, MES și ERP [7], [30].

Modelele matematice și algoritmi de inteligență artificială se selectează în funcție de scop și de consecințele utilizării. În mentenanța predictivă, proiectarea începe prin definirea comportamentului normal, a regimurilor de funcționare și a defectelor care trebuie observate — nu prin alegerea unei arhitecturi de învățare automată. Indicatorii de stare sunt legați, când este posibil, de mecanisme fizice și validați prin evenimente confirmate. Un scor de anomalie nu echivalează automat cu un defect, iar o predicție a duratei de viață rămase trebuie însoțită de incertitudine și de ipotezele privind solicitarea viitoare. Performanța urmărită nu este doar precizia algoritmică, ci și timpul de avertizare, costul alarmelor false, consecințele evenimentelor ratate și capacitatea organizației de a efectua o intervenție utilă [119], [120].

Validarea modelelor reproduce situația reală de utilizare. Separarea datelor previne transferul neintenționat de informație între antrenare și testare, inclusiv între momente apropiate, produse din același lot, imagini ale aceleiași piese sau observații de la același activ. Evaluarea acoperă regimurile, produsele, activele, locațiile, anotimpurile și condițiile de mediu reprezentative. O valoare medie bună poate ascunde o degradare semnificativă într-o situație rară, dar critică. Domeniul de validitate se documentează și se monitorizează după introducerea în exploatare, deoarece modificările procesului, ale senzorilor, ale produselor și ale configurației pot schimba distribuția datelor sau relația cu rezultatul urmărit. Modelul, preprocesarea, pragurile, caracteristicile, rețeta și datele de referință se versionează împreună; revenirea la o versiune anterioară trebuie să fie posibilă și verificată [24], [25], [129].

În viziunea artificială, calitatea deciziei depinde de întregul lanț de observare. Defectul sau caracteristica urmărită trebuie să fie vizibilă și bine reprezentată înainte ca alegerea algoritmului să devină relevantă. Iluminarea, optica, poziționarea piesei, calibrarea, sincronizarea și mecanismul de respingere pot limita performanța mai mult decât arhitectura neuronală. Metodele clasice rămân utile pentru măsurare, localizare și aplicarea de constrângeri geometrice; clasificarea, detecția sau segmentarea se aleg în funcție de rezultatul industrial urmărit. Acceptarea unei stații în producție nu se bazează doar pe seturi de referință, ci presupune evaluarea acceptărilor false, a respingerilor false, a latenței maxime și a aplicării deciziei corecte asupra piesei corecte, cu confirmarea efectului fizic.

Identificarea sistemelor, simularea și controlul predictiv se raportează la întrebarea inginerescă analizată. Experimentul excită dinamica relevantă fără a încălca limitele procesului, iar domeniul de validitate al modelului este verificat și în exploatare. Modelele fizice, hibride și cele bazate exclusiv pe date sunt complementare; alegerea lor depinde de cunoașterea disponibilă, de cantitatea și calitatea datelor, de nevoia de interpretabilitate și de riscul aplicației. Precizia la un pas nu garantează calitatea simulării libere: analiza reziduurilor, stabilitatea și comportamentul în regimuri diferite se tratează împreună cu eroarea medie. Controlul predictiv (MPC) este valoros pentru sisteme multivariabile și constrânse, dar nu înlocuiește automat un PID; fezabilitatea optimizării nu se obține sacrificând constrângerile dure sau protecțiile de siguranță. Modelele învățate utilizate în control sunt încadrate prin estimarea incertitudinii, regiuni de încredere, limite și strategii de rezervă.

Un geamăn digital se definește prin scop și prin domeniul de valabilitate, nu prin cantitatea de informație stocată. El poate susține simularea, diagnosticarea, predicția sau optimizarea; valoarea sa depinde de menținerea legăturii cu activul și de actualizarea controlată a modelelor și a datelor. Fidelitatea necesară variază cu utilizarea: un model pentru verificarea logicii de control nu are aceleași cerințe ca unul pentru estimarea degradării. Geamănul digital este întreținut prin firul digital, prin

managementul modificărilor și prin revalidare pe tot parcursul ciclului de viață. Autoritatea sa asupra procesului este stabilită independent de precizia estimată a modelului [157], [160].

Siguranța în funcționare se analizează prin relația defect–eroare–pierdere a serviciului, nu doar prin disponibilitatea nominală a componentelor. Fiabilitatea, disponibilitatea, siguranța și securitatea sunt proprietăți distincte, care pot intra în conflict. Starea sigură este definită pentru procesul și modul concret de operare: oprirea imediată nu este întotdeauna reacția optimă. Redundanța se evaluează în raport cu independența reală a canalelor și cu defectele comune — software, configurație, alimentare, comunicație, proceduri. Funcționarea degradată, izolarea defectelor și recuperarea se proiectează înaintea incidentului. Restaurarea unui server sau a unei aplicații nu echivalează cu restaurarea instalației; se verifică totdeauna coerența dintre starea software, poziția echipamentelor și condițiile de reluare a producției [36].

Securitatea cibernetică protejează comportamentul procesului fizic, nu doar confidențialitatea datelor. Segmentarea în zone, privilegiul minim, controlul accesului la distanță și monitorizarea sunt proiectate fără a crea dependențe care ar bloca oprirea sigură sau funcționarea locală. Comenzile sunt validate în funcție de identitatea utilizatorului, de autoritatea rolului, de modul de operare și de starea curentă a procesului. O aplicație centrală poate propune o referință, însă controlerul local verifică domeniul admis, interblocările și condițiile fizice. Copiile de siguranță sunt utile numai dacă restaurarea completă este verificată prin exerciții, iar procedurile de recuperare trebuie să permită revenirea la o configurație cunoscută și de încredere. Securitatea cibernetică, siguranța în funcționare și recuperarea sunt validate împreună [3], [4].

Operatorul rămâne informat și capabil să intervină, fără a deveni o cale necontrolată de ocolire a protecțiilor. HMI-ul sprijină conștientizarea situației afișând modul de operare, calitatea și prospekția datelor, limitele automatizării și acțiunile disponibile. O alarmă semnalează o condiție anormală care reclamă o acțiune; ea este prioritarizată după consecință și timpul disponibil și este asociată unui flux de lucru și unei persoane responsabile. Recomandările generate de inteligența artificială sunt însoțite de explicațiile relevante pentru decizie, de incertitudine, de ipoteze și de limitele cunoscute. Supravegherea umană este efectivă doar atunci când persoana primește informațiile necesare, dispune de suficient timp și deține autoritatea reală de a confirma, suspenda sau refuza acțiunea propusă [21], [24], [25].

Sistemele robotice sunt evaluate prin procesul complet, nu doar prin caracteristicile manipulatorului. Sarcina utilă se analizează împreună cu poziția centrului de masă și accelerațiile; efectorul final și alimentarea pieselor pot limita productivitatea mai mult decât robotul însuși. Cinematica, dinamica și controlul forței sunt tratate la nivelul cerut de aplicație, iar bucele rapide rămân în controlerele locale. Funcțiile bazate pe învățare se justifică atunci când variabilitatea sarcinii depășește posibilitățile rezonabile ale programării clasice; ele sunt încadrate de limite, straturi de siguranță și strategii de rezervă. Un robot colaborativ nu face automat o aplicație sigură — evaluarea include unealta, piesa, configurația spațiului și persoanele expuse. Un AMR detectează incertitudinea de localizare și se oprește controlat, iar optimizarea flotei nu intervine direct asupra funcțiilor de siguranță. Recuperarea restabilește o stare fizică și logică coerentă, nu se reduce la ștergerea unei alarme [21], [22].

Integrarea verticală respectă scara temporală, responsabilitatea și autoritatea fiecărui nivel. PLC-ul controlează procesul local, MES-ul coordonează și înregistrează execuția producției, iar ERP-ul gestionează planificarea și tranzacțiile întreprinderii. MES-ul nu înlocuiește interblocările; ERP-ul nu scrie direct valori de referință în controlere. Planul, intenția, execuția și confirmarea sunt reprezentate distinct, astfel încât o comandă transmisă să nu fie confundată cu o acțiune executată sau cu un rezultat

confirmat. Funcționarea fără conexiune și reconcilierea sunt proiectate dinainte, iar calea de scriere către control este limitată, protejată și validată local [7].

Trasabilitatea producției reflectă transformările reale ale materialelor — inclusiv separarea și combinarea loturilor, consumul parțial, rebutul și re prelucrarea. Indicatorii de performanță sunt definiți, versionați și interpretați în raport cu regulile care i-au generat. O valoare OEE nu este automat comparabilă între linii sau perioade, dacă definițiile opririlor și ale produselor conforme diferă. Sistemele de sprijin decizional își declară ipotezele și incertitudinea; recomandările sunt urmărite până la rezultatul operațional. Implementarea MES și ERP înseamnă schimbare organizațională, nu doar un proiect software.

În sistemele industriale mobile, informația spațială devine parte a deciziei. Poziția este însoțită de sistemul de referință, precizie și versiunea hărții; planificarea ia în calcul conturul vehiculului, încărcătura, distanța de oprire și resursele comune. Pierderea localizării sau modificarea mediului sunt detectate și tratate înainte de continuarea misiunii.

Confidențialitatea se proiectează prin minimizarea datelor, prin limitarea scopului și prin procesare locală acolo unde transferul datelor brute nu este necesar. Mecanisme precum agregarea sau învățarea federată nu elimină nevoia de control al accesului și de validare locală. Datele și modelele sunt utilizate numai în condițiile pentru care au fost colectate și acceptate; schimbul între organizații definește proprietatea, durata de păstrare și responsabilitatea pentru efectele unei actualizări.

Sustenabilitatea se evaluează la nivelul întregului sistem și al întregului ciclu de viață. Reducerea consumului energetic al unei mașini nu este suficientă dacă infrastructura digitală asociată produce un consum comparabil sau dacă durata de viață a echipamentelor scade din cauza complexității. Beneficiile sunt puse în balanță cu energia, materialele, apa, deșeurile, fiabilitatea, reparabilitatea și condițiile de muncă. Digitalizarea se justifică atunci când contribuie la un rezultat verificabil, iar costurile complete sunt incluse în evaluare [16].

Validarea finală urmărește serviciul complet, nu doar componentele izolate. Testarea unui algoritm sau a unei interfețe nu dovedește comportamentul corect al sistemului cyber-fizic. Se verifică achiziția și calitatea datelor, sincronizarea, decizia, aplicarea comenzii, răspunsul procesului și capacitatea de recuperare. Închiderea buclei prin reacție este esențială: efectul unei intervenții se măsoară și se folosește la îmbunătățirea modelelor și a procedurilor. O infrastructură de date sau un sistem de inteligență artificială produce valoare numai atunci când informația rezultată este asociată unei responsabilități, unei acțiuni și unei verificări.

Principiul care le unește pe toate este disciplina inginerescă. Tehnologiile, algoritmii și platformele se schimbă, dar criteriile fundamentale rămân ancorate în proces, consecințe, timp, date, resurse și responsabilitate. Un sistem informatic industrial bine proiectat nu este doar conectat și performant în regim nominal, ci rămâne inteligibil, controlabil și recuperabil atunci când datele sunt incomplete, comunicația se întrerupe sau apar situații neprevăzute. Valoarea sa este dovedită prin rezultate operaționale măsurabile, printr-un comportament sigur și prin capacitatea de a fi exploatat și menținut pe întreaga durată de viață.

Partea a VI-a — Studii de caz și aplicații integrate

12. Studii de caz de informatică industrială

Domeniu și metodă de analiză

Studiile de caz din acest capitol urmăresc aplicarea integrată a conceptelor dezvoltate în partea teoretică. Ele nu reiau definițiile generale privind calculul la margine (*edge computing*), inteligența artificială, comunicațiile industriale, mentenanța predictivă, gemenii digitali, securitatea cibernetică sau integrarea MES–ERP, ci evidențiază modul în care aceste tehnologii și principii sunt combinate într-un sistem operațional complet. Scopul nu este descrierea exhaustivă a unor produse comerciale și nici furnizarea unor soluții șablon, ci formularea unor probleme ingineresti suficient de concrete pentru analiză, proiectare, seminarii aplicate sau dezvoltarea unor laboratoare practice.

Fiecare studiu pornește de la un proces fizic real și urmărește lanțul complet prin care o observație din teren este transformată într-o acțiune fizică verificabilă:

Proces fizic → Măsurare → Achiziție și comunicație → Contextualizare → Analiză → Decizie
→ Acțiune → Confirmarea rezultatului

La fiecare etapă a acestui lanț pot interveni erori, întârzieri, pierderi de pachete sau interpretări incorecte. Un senzor poate prezenta derivă de calibrare, o măsurătoare poate fi asociată greșit cu alt produs, un mesaj pe magistrală poate sosi după termenul-limită (*deadline*), iar un model analitic poate primi date din afara domeniului său de validitate. O decizie poate fi impecabilă din punct de vedere algoritmic și, cu toate acestea, să fie complet incompatibilă cu starea fizică a mașinii. Din acest motiv, studiile de caz nu evaluează doar acuratețea izolată a unei metode de calcul, ci robustețea întregului traseu dintre fenomenul fizic observat și efectul tehnologic produs.

O soluție industrială viabilă trebuie analizată prin calitatea și prospețimea măsurărilor, sincronizarea temporală a canalelor, distribuirea corectă a funcțiilor între controlul local (PLC), nodul edge și platformele centrale, precum și prin comportamentul la pierderea comunicației. Trebuie evaluate protecțiile autonome locale, funcționarea în regim degradat, autoritatea operatorului uman, securitatea căilor de comandă, trasabilitatea versiunilor de software și posibilitatea revenirii rapide la o configurație anterioară sigură (*rollback*).

În ingineria industrială se aplică un set strict de principii practice:

- **O comandă transmisă** nu demonstrează că actuatorul a produs efectiv mișcarea cerută;
- **O alarmă confirmată pe ecran** nu atestă că defecțiunea din teren a fost remediată fizic;
- **O recomandare corectă statistic** devine inutilă dacă ajunge după expirarea ferestrei optime de intervenție sau dacă nu este integrată într-un flux operațional de lucru (*workflow*).

Structura standardizată a studiilor de caz

Toate cele zece cazuri utilizează o structură unitară, organizată în cinci secțiuni ingineresti:

1. **Contextul cazului și problema:** definește procesul tehnologic, anomaliile observate și obiectivul operațional urmărit;
2. **Arhitectura propusă:** identifică traductoarele, actuatorele, controlerele, magistralele de comunicație și platformele software, delimitând strict responsabilitatea fiecărui nivel;
3. **Scenariul operațional:** introduce o perturbație concretă (un defect mecanic, o anomalie de calitate, o cădere de rețea sau o abatere energetică) și descrie modul în care sistemul reacționează;

4. **Sarcini și decizii de proiectare:** implică alegeri argumentate privind ratele de eșantionare, protocoalele, strategiile de control, algoritmi de analiză, securitatea cibernetică și mecanismele de siguranță (*fallback*);
5. **Livrabile, evaluare și întrebări:** stabilesc documentele tehnice cerute, indicatorii măsurabili de performanță (ROI, disponibilitate, precizie) și întrebări critice privind limitele de aplicabilitate ale arhitecturii.

Cele zece cazuri au fost selectate pentru a acoperi principalele provocări ale informaticii industriale moderne, realizând o trecere graduală de la echipamente individuale din fabrică la sisteme distribuite, mobile și autonome.

Tabelul 12.1. Acoperirea tehnică a studiilor de caz

Studiu de caz	Problemă dominantă	Elemente integrate
1. Mentenanța unui ansamblu motor–pompă	Detectarea degradării	Senzori, prelucrarea semnalelor, inteligență artificială <i>edge</i> și CMMS
2. Inspecția optică automată	Decizia de calitate	Viziune artificială, PLC, MES și trasabilitate
3. Atelier eficient energetic	Resurse și sustenabilitate	Contorizare, linie de referință energetică, analiză și programarea producției
4. Linie de ambalare integrată	Execuția producției	PLC, SCADA, MES și ERP
5. Modernizarea unei celule existente	Integrarea instalațiilor existente (<i>brownfield</i>)	Modbus, echipament de interconectare (<i>gateway</i>), OPC UA și MQTT
6. Flotă AMR	Logistică internă autonomă	Localizare, managementul flotei, WMS/MES și siguranță funcțională (<i>safety</i>)
7. Celulă colaborativă	Interacțiunea om–robot	Cobot, controlul interacțiunii, HMI și siguranță funcțională (<i>safety</i>)
8. Clădire industrială inteligentă	Energie, confort și disponibilitate	BMS (<i>Building Management System</i>), BACnet, detecție automată a defectelor (FDD) și control predictiv (MPC)
9. Mobilitate urbană cooperativă	Percepție și coordonare distribuită	ADAS (<i>Advanced Driver Assistance Systems</i>), comunicații V2X (<i>Vehicle-to-Everything</i>), procesare <i>edge</i> și control adaptiv al traficului
10. Fermă cyber-fizică	Automatizare geospațială	Rețele LPWAN, irigație automatizată, magistrală ISOBUS și utilaje autonome

Prin această abordare, rezultatul unui studiu de caz nu este redus la simpla existență a unui tablou de bord (*dashboard*) sau la acuratețea teoretică a unui algoritm, ci demonstrează închiderea completă și sigură a buclei de decizie în mediul industrial.

Studiul de caz 1. Mentenanța predictivă a unui ansamblu motor–pomă

Contextul cazului și problema

O instalație industrială utilizează motoare asincrone pentru acționarea unor pompe centrifuge de proces. Înlocuirea preventivă a rulmenților la intervale fixe generează costuri ridicate și nu previne avariile accidentale survenite între revizii. Înaintea defectării s-au constatat creșteri ale vibrațiilor și ale temperaturii lagărelor, dezechilibre ale curenților absorbiți și scăderea randamentului hidraulic. Obiectivul este proiectarea unui sistem capabil să detecteze degradarea incipientă, să distingă defectele mecanice de anomaliiile de măsurare, să estimeze durata de viață utilă rămasă (*RUL*) și să integreze automat recomandările în fluxul de mentenanță [119], [120], [123].

Arhitectura propusă

Sistemul include accelerometre piezoelectrice, traductoare de temperatură PT100 pe lagăre, senzori de curent și tensiune (pentru analiză MCSA- *Motor Current Signature Analysis*), traductoare de presiune diferențială și un debitmetru electromagnetic. Un nod de calcul edge achiziționează sincron semnalele, calculează indicatori statistici (RMS, factor de creastă, curtoză, energie spectrală) și rulează un model nesupervizat de detecție a anomaliilor, menținând monitorizarea locală chiar și la căderea rețelei. Datele agregate sunt transmise prin OPC UA și MQTT către arhiva istorică de proces (*process historian*), iar sistemul CMMS primește exclusiv notificări validate [11], [125].

Scenariul operațional

O componentă spectrală asociată căii de rulare crește progresiv pe fondul funcționării pompei la sarcină redusă. Nodul edge normalizează indicatorii în funcție de punctul de funcționare, eliminând alarmele false. Odată confirmată persistența tendinței și creșterea temperaturii, severitatea este ridicată. Nivelul central corelează starea cu istoricul, stocul de piese de schimb și planificarea din MES, generând un ordin de intervenție programat la următoarea pauză tehnologică.

Sarcini și decizii

Proiectarea impune dimensionarea lanțului de achiziție (frecvențe de eșantionare, filtre antialiasing), distribuirea funcțiilor de procesare între edge și platforma centrală și configurarea logicii de detecție. Este necesară stabilirea modului de reprezentare a incertitudinii estimării RUL, tratarea derivei de calibrare a senzorilor (*drift*) și definirea procedurii de reinițializare a liniei de bază după finalizarea reparației mecanice [129], [259].

Livrabile și evaluare

- **Livrabile:** Schema bloc a sistemului, dicționarul de semnale, matricea modurilor de defectare (FMEA), fluxul de calcul edge, logica alarmelor și diagrama de integrare cu CMMS.
- **Indicatori:** Rata de detecție timpurie ($> 95\%$), numărul de alarme false pe lună, eroarea de predicție a RUL și reducerea timpilor morți neprogramați.

Întrebări pentru discuție

- Cum se separă vibrațiile cauzate de cavitație de cele produse de o fisură în rulment?
- De ce raportarea punctuală a duratei RUL este riscantă în planificarea mentenanței?
- Ce mecanism de siguranță (*fallback*) se aplică dacă nodul edge pierde conexiunea la rețeaua fabricii?
- Cum se validează eficiența modelului predictiv pe baza pieselor înlocuite fizic?

Studiul de caz 2. Inspecția optică automată a componentelor fabricate

Contextul cazului și problema

O linie automată produce piese metalice ștanțate la un ritm de 1 piesă pe secundă (*takt time* = 1 s). Defectele tipice includ fisuri, bavuri, abateri dimensionale și zgârieturi pe suprafața activă. Controlul vizual manual este lent, subiectiv și permite scăparea unor piese neconforme la clienți. Obiectivul este implementarea unei stații de inspecție optică automată (AOI) capabile să inspecteze 100% din repere, să transmită decizia de conformitate către PLC în timp util pentru devierea pe banda de rebut și să asigure trasabilitatea fiecărui produs [140], [141], [144], [146].

Arhitectura propusă

Stația cuprinde o incintă optică izolată, o sursă de iluminare structurată cu dom LED, o cameră industrială monocromă de înaltă rezoluție cu lentilă telecentrică și o barieră optică de declanșare (*trigger*). Un calculator industrial (IPC) rulează algoritmi clasici de metrologie dimensională și rețele neuronale pentru detecția defectelor de textură. Decizia binară este transmisă determinist către PLC prin I/O digitale sau PROFINET, iar imaginile pieselor respinse sunt arhivate automat în MES împreună cu metadatele de lot [135].

Scenariul operațional

O piesă prezintă o micro-fisură marginală într-o zonă critică. Senzorul de prezență declanșează expunerea camerei sincronizată cu flash-ul luminos, iar imaginea este procesată în memoria IPC-ului. Algoritmul finalizează inferența în 220 ms (încadrându-se în bugetul total alocat de 600 ms). IPC-ul transmite semnalul de rebut către PLC, care comandă cilindrul pneumatic de sortare. Un senzor de barieră optică de pe jgheabul de carantină confirmă PLC-ului că piesa a fost deviată fizic.

Sarcini și decizii

Se impune dimensionarea câmpului vizual (*FOV*), a rezoluției optice și a geometriei de iluminare, alături de calculul riguros al bugetului temporal complet de inspecție (T_{total}). Trebuie stabilită procedura de calibrare periodică cu mostre etalon (*Golden Samples*), implementarea mecanismelor de monitorizare a derivatei optice și logica de tratare a reperelor clasificate în zona gri de incertitudine statistică [32].

Livrabile și evaluare

- **Livrabile:** Schema opto-mecanică, cronograma bugetului de timp, fluxul de procesare a imaginii, matricea de confuzie a clasificatorului și logica de interblocare cu PLC-ul.
- **Indicatori:** Rata defectelor scăpate – *Miss Rate* ($< 0.05\%$), rata de respingere falsă a pieselor bune ($< 0.5\%$), timpul de ciclu și disponibilitatea stației.

Întrebări pentru discuție

- De ce este esențială o lentilă telecentrică în raport cu un obiectiv clasic la măsurători dimensionale?
- Cum se gestionează variația naturală a texturii pieselor brute fără a declanșa alarme false?
- Ce riscuri operaționale apar dacă semnalul de respingere se transmite prin mesagerie asincronă TCP/IP?
- Cum se asigură securitatea imaginilor arhivate pentru prevenirea manipulării datelor de calitate?

Studiul de caz 3. Operarea eficientă a unui atelier din punct de vedere energetic și a consumurilor materiale

Contextul cazului și problema

Un atelier de prelucrări mecanice cuprinde 10 centre de frezare CNC, o stație centrală de aer comprimat și un circuit de răcire tehnologică. Facturile lunare indică vârfuri de putere taxate suplimentar, consumuri reziduale/parazite ridicate în perioadele de repaus și pierderi nemonitorizate de aer comprimat. Obiectivul este implementarea unui sistem de management energetic conform ISO 50001:2018, capabil să determine consumul specific de energie (SEC), să localizeze pierderile tehnologice și să optimizeze funcționarea utilajelor fără a afecta programul de fabricație [16], [248], [249].

Arhitectura propusă

Sistemul integrează analizoare trifazate de rețea pe fiecare utilaj, debitmetre termice de masă pe circuitele de aer comprimat, debitmetre ultrasonice și senzori de temperatură pe circuitele de răcire. Datele sunt colectate prin gateway-uri Modbus TCP și publicate prin OPC UA către platforma locală de monitorizare energetică. Sistemul este corelat cu MES-ul pentru a asocia consumurile cu comenzile de producție curente și cu stările utilajelor (*mers în gol, prelucrare, oprire tehnologică*) [27], [246].

Scenariul operațional

În timpul schimbului de noapte, debitmetrul de pe magistrala de aer comprimat indică un consum rezidual de 35% din capacitatea nominală, deși toate centrele CNC sunt oprite. Sistemul emite o alertă de pierderi masive prin neetanșetăți pe un circuit secundar. Simultan, la reluarea producției, algoritmul de optimizare detectează riscul depășirii puterii maxime contractate și decalează automat pornirea a două cicluri de încălzire (*peak shaving*).

Sarcini și decizii

Analiza presupune definirea consumatorilor semnificativi de energie (SEU), construirea liniei de referință energetice (*Energy Baseline – EnB*) prin regresie multivariabilă și formularea logicii de trecere automată a utilajelor în regim *standby*. Se impune configurarea modului de calcul al consumului specific ($SEC = E/Q_{bun}$) și stabilirea procedurii de verificare a economiilor conform ISO 50015 [250].

Livrabile și evaluare

- **Livrabile:** Diagrama bilanțului energetic (Sankey), specificația contoarelor și a gateway-urilor, ecuațiile liniei de bază *EnB*, tabloul de bord energetic și procedurile de audit intern.
- **Indicatori:** Reducerea consumului specific SEC , diminuarea vârfurilor de putere absorbită, reducerea pierderilor de aer comprimat și economia financiară anuală realizată.

Întrebări pentru discuție

- De ce raportarea consumului brut de energie fără corelare cu volumul de piese bune este eronată?
- Cum se compensează influența variațiilor de temperatură exterioară asupra consumului de climatizare?
- Ce constrângeri de producție trebuie respectate înainte ca sistemul să decaleze pornirea unui utilaj?
- Cum se demonstrează că o măsură de eficiență energetică nu a mărit rata de rebuturi mecanice?

Studiul de caz 4. Integrarea verticală a unei linii de ambalare

Contextul cazului și problema

O fabrică farmaceutică deține o linie automată de ambalare compusă din trei mașini independente: blisterizare, cartonare și paletizare. Fiecare utilaj dispune de un PLC propriu, însă transferul comenzilor de producție și al rețetelor se realizează manual pe formulare tipărite. Această abordare generează timpi mari de schimbare a formatului, erori la setarea temperaturilor de sigilare și dificultăți majore în reconcilierea trasabilității loturilor de medicamente. Obiectivul este integrarea verticală completă a liniei conform standardelor ANSI/ISA-95 și ANSI/ISA-88 [7], [213].

Arhitectura propusă

Arhitectura conectează controlerele de mașină printr-o rețea Ethernet industrială dedicată la un server SCADA de linie și la sistemul MES. Schimbul de informații se bazează pe mașina de stări standardizată PackML (ISA-TR88.00.02) și pe servere OPC UA integrate în PLC-uri. Sistemul ERP emite ordinul de producție, MES-ul îl descompune în operații tehnologice și descarcă rețeta validată către mașini, iar stările de execuție sunt sincronizate automat între niveluri [67], [216], [217].

Scenariul operațional

Un ordin de ambalare pentru 30.000 de unități este eliberat din ERP. MES-ul verifică disponibilitatea liniei și descarcă parametrii rețetei în PLC-ul de blisterizare. În timpul funcționării, un blocaj mecanic pe mașina de cartonare comută starea PackML în *Held*; mașina din amonte comută automat în *Suspending*, prevenind aglomerarea ambalajelor pe conveyer. După remediere, producția se reia coordonat, iar MES-ul transmite către ERP cantitățile realizate și timpii de funcționare.

Sarcini și decizii

Este necesară implementarea modelului PackML pe fiecare controler, structurarea spațiului de adrese OPC UA conform ierarhiei ISA-95 și proiectarea descărcării securizate a rețetelor fără posibilitatea modificării locale neautorizate. Se impune configurarea calculului în timp real al indicatorului OEE defalcat ($A \times P \times Q$) și a mecanismelor de trasabilitate a loturilor conform cerințelor de calitate [14], [218].

Livrabile și evaluare

- **Livrabile:** Modelul funcțional ISA-88/95, diagrama de stări PackML, specificația tag-urilor OPC UA, schema fluxului de date ERP–MES–PLC și ecranul HMI de calcul OEE.
- **Indicatori:** Creșterea disponibilității liniei (A), scurtarea timpului de schimbare a formatului (*setup*), eliminarea erorilor de ambalare și trasabilitatea completă a lotului.

Întrebări pentru discuție

- De ce este interzis ca sistemul ERP să scrie comenzi de execuție direct în registrele PLC-ului?
- Cum se garantează continuarea sigură a producției dacă legătura dintre MES și linie este întreruptă?
- Cum se asigură conformitatea integrității datelor cerută de normele farmaceutice?
- Cum se calculează viteza ideală a liniei când utilajele componente au capacități nominale diferite?

Studiul de caz 5. Modernizarea securizată a unei celule de fabricație existente

Contextul cazului și problema

O secție de tratamente termice operează două cuptoare industriale comandate de PLC-uri vechi (*legacy*), conectate prin magistrale seriale Modbus RTU / RS-485. Echipamentele nu dispun de interfețe Ethernet native și nu oferă funcții de securitate cibernetică, însă procesul fizic este stabil și fiabil. Conducerea fabricii dorește achiziția datelor de temperatură în platforma de analiză fără a înlocui controlerile de bază și fără a expune instalația la atacuri de rețea. Obiectivul este realizarea unui retrofit IIoT securizat conform standardului IEC 62443 [11], [31], [65].

Arhitectura propusă

Se instalează un gateway industrial edge dotat cu două interfețe de rețea izolate fizic (*dual-homed*). Portul serial comunică exclusiv local cu PLC-urile prin interogare Modbus RTU master. Gateway-ul convertește registrele brute în obiecte semantice standardizate și le expune către rețeaua superioară prin OPC UA securizat (TLS) și mesaje MQTT autentificate cu certificate X.509. În amonte se instalează un firewall industrial configurat conform modelului Purdue pentru separarea strictă a celulei de rețeaua IT a fabricii [3], [4].

Scenariul operațional

O tentativă de scanare a porturilor provenită dintr-o rețea comercială este oprită de firewall-ul celulei, iar gateway-ul refuză conexiunile neautorizate. Pe canalul intern de câmp, gateway-ul citește ciclic temperaturile la fiecare 500 ms, salvează datele într-o bază locală time-series în cazul indisponibilității rețelei centrale și retransmite pachetele comprimate imediat ce conexiunea este restaurată, fără a pierde date de proces.

Sarcini și decizii

Proiectarea impune definirea frecvenței de interogare Modbus (pentru a evita supraîncărcarea ciclului PLC-ului), configurarea zonelor și conductelor de securitate conform IEC 62443-3-3 și realizarea mapării semantice a registrelor numerice. Se impune aplicarea politicilor stricte de acces pe roluri (RBAC) și blocarea oricăror comenzi de scriere transmise de la distanță către actuatori [3].

Livrabile și evaluare

- **Livrabile:** Topologia de rețea segmentată, matricea de mapare Modbus-la-OPC UA, regulile de filtrare firewall, planul de gestiune a certificatelor și analiza de risc cibernetic.
- **Indicatori:** Disponibilitatea fluxului de telemetrie ($> 99.9\%$), timpul de scanare serială, latența de publicare și absența incidentelor de securitate.

Întrebări pentru discuție

- Ce vulnerabilități critice apar dacă gateway-ul edge este configurat cu funcție de rutare directă (*IP forwarding*)?
- Cum se previne blocarea execuției programului din PLC la interogări Modbus prea frecvente?
- Cum se compensează erorile de calibrare ale senzorilor vechi dacă firmware-ul PLC-ului nu poate fi modificat?
- Ce mecanism garantează că gateway-ul edge nu poate trimite comenzi fizice neautorizate către arzătoare?

Studiul de caz 6. Flotă de roboți mobili autonomi (AMR) pentru logistică internă

Contextul cazului și problema

Un depozit logistic de componente auto urmărește eliminarea transportului manual cu motostivuitoare prin introducerea unei flote de 6 roboți mobili autonomi (AMR). Mediul de lucru este dinamic, cu pietoni, paletă repoziționați și culoare înguste de trecere. Obiectivul este integrarea unitară a flotei pe baza standardului VDA 5050, asigurând sincronizarea misiunilor de transport cu sistemul WMS al depozitului și garantarea securității personalului în conformitate cu standardul ISO 3691-4:2023 [202], [207], [212].

Arhitectura propusă

Fiecare robot AMR integrează două scanere laser de securitate (LiDAR 2D) pentru navigație SLAM și câmpuri de protecție dinamice, camere 3D pentru detecția obstacolelor suspendate și o unitate de calcul industrială echipată cu ROS 2 (*Robot Operating System*). Comunicația cu serverul central de management al flotei se realizează printr-o rețea wireless industrială Wi-Fi 6 (sau 5G privat), utilizând protocolul MQTT peste care se transmit mesajele standardizate VDA 5050 (*Orders, Instant Actions, State*). Serverul de flotă este interfațat cu WMS-ul și MES-ul fabricii [205].

Scenariul operațional

Sistemul WMS emite un ordin de reprovizionare cu piese la postul de montaj 4. Managerul de flotă atribuie misiunea robotului AMR-02, transmițând traiectoria ca o succesiune de noduri și muchii VDA 5050. Pe traseu, un operator uman traversează culoarul. Scannerul laser detectează pietonul în câmpul de avertizare și reduce viteza robotului; la intrarea în câmpul de oprire de siguranță, AMR-02 frânează controlat până la oprire completă. După eliberarea culoarului, robotul își reia automat deplasarea.

Sarcini și decizii

Este necesară configurarea câmpurilor de protecție laser în funcție de viteza de deplasare și sarcina transportată, implementarea stivei de mesaje VDA 5050 / MQTT și proiectarea algoritmilor de evitare a blocajelor reciproce (*deadlock*) la intersecții. Trebuie definite procedurile de trecere în regim manual și mecanismele de siguranță la pierderea semnalului radio [185].

Livrabile și evaluare

- **Livrabile:** Harta digitală SLAM a depozitului, configurația ariilor de securitate laser, specificația interfeței VDA 5050, analiza de risc conform ISO 3691-4 și tabloul de bord al flotei.
- **Indicatori:** Rata livrărilor la timp (*On-Time Delivery*), numărul de opriri de urgență, gradul de utilizare a flotei și timpul mediu de transport per palet.

Întrebări pentru discuție

- Care este diferența arhitecturală fundamentală între un vehicul ghidat automat (AGV) și un robot autonom (AMR)?
- Ce comportament de siguranță adoptă robotul dacă pierde conexiunea Wi-Fi/5G în timpul deplasării?
- Cum se asigură cooperarea sigură când în aceeași hală circulă simultan roboți AMR și stivuitoare manuale?
- De ce senzorii laser utilizați pentru navigație SLAM nu pot prelua funcții de oprire dacă nu sunt certificați *PL d* (conform ISO 13849-1)?

Studiul de caz 7. Celulă colaborativă pentru asamblare mecanică

Contextul cazului și problema

O stație dintr-o fabrică de echipamente auto necesită montarea manuală a unor garnituri elastice și înșurubarea automată a 6 șuruburi la un cuplu controlat. Spațiul disponibil este redus, ceea ce împiedică instalarea unor garduri fizice de protecție. Soluția proiectată este o celulă robotizată colaborativă în care operatorul uman și un robot colaborativ (*cobot*) împart același spațiu de lucru. Obiectivul este realizarea aplicației în deplină conformitate cu specificația ISO/TS 15066:2016 și standardul ISO 10218-2:2025 [22], [197], [198].

Arhitectura propusă

Celula conține un braț robotizat colaborativ cu 6 grade de libertate prevăzut cu traductoare interne de cuplu pe fiecare articulație, o șurubelniță automată cu monitorizare de cuplu și unghi, un controller de siguranță (*Safety PLC*) și o interfață HMI intuitivă. *Cobotul* implementează regimul de limitare a puterii și a forței (*PFL*), combinat cu monitorizarea vitezei și a separării prin intermediul unui scanner laser de securitate montat sub masa de lucru [21], [23].

Scenariul operațional

Operatorul poziționează corpul piesei și montează garnitura de cauciuc. Cobotul preia șuruburile și se apropie cu viteză redusă de zona de asamblare. Dacă operatorul se află în proximitate, viteza brațului este limitată automat la 250 mm/s. În cazul unui contact fizic accidental cu mâna operatorului, traductoarele articulare detectează forța de contact (< 130 N, sub pragul biomecanic de durere admis de ISO/TS 15066) și opresc instantaneu mișcarea prin relaxare elastică (*back-drive*), eliminând riscul de strivire.

Sarcini și decizii

Proiectarea impune calculul forțelor maxime de contact tranzitoriu și cvasistatic permise pe fiecare zonă a corpului uman, optimizarea geometriei efectorului final (rotunjirea muchiilor sculei) și configurarea zonelor de viteză pe scannerul laser. Este necesară implementarea secvenței de reinițializare a celulei după o oprire de protecție și proiectarea ergonomică a interfeței HMI [19], [175].

Livrabile și evaluare

- **Livrabile:** Raportul de evaluare a riscurilor, planul de dispunere al zonelor colaborative, buletinul de măsurare a forțelor de impact cu dinamometrul de contact, logica Safety PLC și instrucțiunile de lucru.
- **Indicatori:** Timpul de ciclu per ansamblu, forța maximă la impact, frecvența opririlor de protecție, confortul operatorului și conformitatea cu marcajul CE.

Întrebări pentru discuție

- De ce certificarea *cobotului* ca echipament nu garantează automat securitatea întregii aplicații robotice?
- Cum se face distincția tehnică între o coliziune cu operatorul și forța normală opusă la strângerea șurubului?
- Ce caracteristici constructive trebuie să aibă degetele *gripper*-ului pentru a preveni riscurile de perforare?
- De ce butonul de rearmare a celulei după oprire trebuie amplasat în afara zonei colaborative?

Studiul de caz 8. Monitorizarea și controlul inteligent al unei clădiri industriale (BMS)

Contextul cazului și problema

O clădire industrială multifuncțională include birouri, laboratoare tehnologice, camere curate de producție și o centrală termică/HVAC. Subsistemele de climatizare, iluminat, distribuție electrică și detecție incendiu funcționează independent, provocând risipă de energie și dificultăți în menținerea presiunii diferențiale în camerele curate. Obiectivul este integrarea subsistemelor într-o platformă de automatizare a clădirii (BMS – *Building Management System*) conform ISO 16484, aplicând algoritmi de diagnosticare automată a defectelor (FDD) și control predictiv (MPC) [260], [261], [262].

Arhitectura propusă

Sistemul utilizează controlere digitale directe (DDC) amplasate în tablourile HVAC și tablourile electrice, comunicând prin BACnet/IP și magistrale seriale Modbus RTU / DALI. O rețea de senzori măsoară temperatura, umiditatea, concentrația de CO₂, compușii organici volatili (VOC), presiunea diferențială și ocuparea spațiilor. Serverul central BMS integrează o bază de date time-series, un motor de reguli FDD pentru detecția vanelor blocate și un modul MPC care optimizează pornirea centralelor de tratare a aerului (CTA) în funcție de prognoza meteo și tariful orar al energiei [122], [148], [154], [236].

Scenariul operațional

Într-o cameră curată de producție, senzorul de presiune diferențială indică o scădere lentă a suprapresiunii, concomitent cu creșterea consumului electric al ventilatorului de introducere. Modulul FDD corelează presiunea, poziția clapetei și curentul motorului, generând o alertă de „filtru HEPA colmatat peste pragul de 80%”. Sistemul mărește turația convertizorului pentru a menține presiunea pozitivă și transmite un tichet automat de înlocuire către echipa de mentenanță.

Sarcini și decizii

Se impune configurarea spațiului de obiecte BACnet (intrări/ieșiri analogice și binare), formularea funcției de cost a regulatorului MPC pentru optimizarea energiei termice și stabilirea matricei de interblocare de securitate la incendiu (*Life Safety*). Este necesară implementarea logicii de compensare a debitului de aer proaspăt pe baza concentrației de CO₂ [68].

Livrabile și evaluare

- **Livrabile:** Topologia rețelei BACnet/IP, lista de puncte I/O fizice și virtuale, regulile FDD de diagnosticare a vanelor și clapetelor, algoritmul de reglare MPC și tabloul de bord energetic.
- **Indicatori:** Reducerea consumului de energie HVAC (> 15%), menținerea calității aerului sub 800 ppm CO₂, stabilitatea presiunii în camerele curate și timpul de detecție a defectărilor.

Întrebări pentru discuție

- De ce comenzile de oprire la semnalul de incendiu trebuie executate prin relee cablate fizic, ocolind software-ul BMS?
- Cum se evită fenomenul de pendulare (*hunting*) între vana de încălzire și cea de răcire ale aceleiași centrale de aer?
- Ce politici de securitate cibernetică sunt obligatorii pentru accesul tehnicienilor prin conexiuni VPN la rețeaua BACnet?
- Cum se verifică faptul că optimizarea energetică nu a degradat calitatea aerului și confortul ocupanților?

Studiul de caz 9. Mobilitate urbană cooperativă cu sisteme ADAS și V2X

Contextul cazului și problema

Un coridor de transport public urban prezintă blocaje, întârzieri ale autobuzelor și risc ridicat de coliziune la virajele la stânga în intersecțiile semaforizate. Senzorii de pe vehicule (camere, radar) au câmpul vizual obstrucționat de clădiri sau autovehicule mari (*obstacole ocluzive*). Obiectivul este realizarea unui sistem de transport inteligent cooperativ (C-ITS) bazat pe comunicații vehicul-către-infrastructură (V2X), capabil să acorde prioritate transportului public și să transmită avertizări timpurii de coliziune [200], [201], [206], [263], [264].

Arhitectura propusă

Arhitectura cuprinde unități instalate la marginea drumului (RSU – *Roadside Units*) în intersecții, echipate cu camere video, senzori radar și procesoare edge pentru detecția pietonilor și bicicliștilor. Autobuzele și vehiculele de test sunt dotate cu unități de bord (OBU – *On-Board Units*) conectate la magistrala CAN și la o antenă C-V2X (PC5 direct / 5G). Centrul de management al traficului coordonează automatele de semaforizare prin protocoale standardizate [204], [211], [226].

Scenariul operațional

Un autobuz întârziat cu 3 minute se apropie de intersecție. Unitatea OBU transmite o cerere de prioritate (SRM) și primește confirmarea stării semaforului (SSM). Simultan, camera RSU detectează un pieton angajat în traversare pe culoarea roșie după o curbă fără vizibilitate. Procesorul RSU transmite un mesaj colectiv de percepție (CPM) și un avertisment critic către autobuz. Șoferul este alertat pe display, iar automatul de semaforizare prelungește faza de verde doar după eliberarea completă a trecerii.

Sarcini și decizii

Analiza impune configurarea stivei V2X (latență < 20 ms, rază directă > 300 m), definirea mesajelor ETSI (CAM, DENM, CPM) și SAE (SPaT, MAPEM) și elaborarea algoritmului de prioritate semaforică. Este necesară verificarea plauzibilității datelor și protecția împotriva atacurilor de tip *spoofing* prin certificate digitale PKI [203].

Livrabile și evaluare

- **Livrabile:** Topologia de comunicație C-ITS, specificația interfețelor V2X, diagrama fluxului de mesaje de siguranță, algoritmul de adaptare a semaforizării și scenariile de testare în simulator HIL.
- **Indicatori:** Scurtarea timpului de călătorie al autobuzelor, reducerea numărului de frânări de urgență, latența de livrare a mesajelor critice și eliminarea accidentelor pe coridor.

Întrebări pentru discuție

- Ce mecanism de siguranță intervine pe vehicul dacă mesajul de avertizare V2X sosește după expirarea timpului util de frânare?
- De ce informația V2X este utilizată doar pentru avertizare și sprijin decizional, iar frânarea automată depinde de senzorii proprii ai mașinii?
- Cum se garantează confidențialitatea traseului vehiculelor civile prin rotirea periodică a pseudonimelor în rețeaua V2X?
- Cum gestionează automatul de semaforizare recepționarea simultană a două cereri de prioritate de pe direcții perpendiculare?

Studiul de caz 10. Fermă cyber-fizică: irigație de precizie, ISOBUS și utilaj autonom

Contextul cazului și problema

O fermă agricolă cuprinde culturi de câmp și sere automatizate. Irigațiile tradiționale bazate pe programe orare fixe irosesc cantități mari de apă și energie, fertilizarea uniformă ignoră variabilitatea solului, iar utilajele de la producători diferiți au terminale incompatibile. Obiectivul este dezvoltarea unui sistem integrat bazat pe senzori IoT distribuiți geografic, comunicații fără fir de mică putere (LPWAN), hărți de prescripție (hărți de aplicare cu rată variabilă) și interoperabilitate ISOBUS (ISO 11783) [77], [241], [242].

Arhitectura propusă

Sistemul integrează noduri senzoriale wireless autonome (pe baterie și panou solar) care măsoară umiditatea solului pe trei adâncimi, conductivitatea electrică, temperatura și radiația solară, transmițând datele prin LoRaWAN către un gateway 4G/5G. Stația de pompare și electrovalvele sunt comandate de un PLC local. Pentru operațiunile de câmp, tractorul și utilajele comunică prin magistrala ISOBUS, utilizând un modul *Task Controller* (TC-BAS/TC-GEO) care interpretează hărțile geospațiale și reglează rata de aplicare prin ghidare GNSS RTK [77], [78], [265], [266].

Scenariul operațional

Senzorii confirmă o rezervă optimă de apă pe parcela 3 în pofida prognozei caniculare; serverul reduce durata de irigație și comandă PLC-ul de pompare. Simultan, pe parcela 1, un utilaj autonom de erbicidat parcurge rândurile: sistemul ISOBUS modulează debitul pe fiecare duză conform hărții de infestare încărcate și oprește automat secțiunile la capăt de rând (*Section Control*) pentru a preveni suprapunerea tratamentelor [267], [268], [269].

Sarcini și decizii

Proiectarea impune dimensionarea rețelei LoRaWAN (buget radio, frecvență și autonomie a bateriilor), proiectarea logicii de control al irigației în buclă închisă și configurarea obiectelor ISOBUS (ISO 11783-10) pentru controlul secțiunilor și aplicarea cu rată variabilă (*VRA*). Este necesară definirea limitelor de siguranță geografică (*Geofencing*) și a mecanismelor de siguranță la pierderea semnalului satelitar GNSS [129], [160].

Livrabile și evaluare

- **Livrabile:** Topologia rețelei LoRaWAN, schema logică de irigație automată, fișierele de schimb de date în format ISO-XML, specificația arhitecturii ISOBUS și planul de testare agronomică.
- **Indicatori:** Reducerea consumului de apă ($> 20\%$), economia de chimicale prin eliminarea suprapunerilor, precizia de ghidare sub 2 cm și autonomia nodurilor senzoriale (> 3 ani).

Întrebări pentru discuție

- Ce măsuri de protecție se iau dacă o electrovalvă de irigație primește comandă de deschidere, dar debitmetrul indică flux zero?
- De ce standardul ISOBUS este critic pentru evitarea dependenței de un singur producător de tractoare și utilaje?
- Cum se gestionează pierderea semnalului de corecție GNSS RTK în timpul aplicării autonome a substanțelor toxice?
- Cum se reconciliază măsurătorile punctuale ale senzorilor de sol cu hărțile multi-spectrale preluate din satelit?

Lucrări de laborator

Lucrare de laborator 1: Dezvoltare de sisteme de tip Internet of Things (IoT) cu ESP8266

1. Introducere

Tehnologia IoT este destinată pentru monitorizarea și controlul unor echipamente electromecanice simple sau mai complexe prin intermediul Internetului. IoT stă la baza multor aplicații de monitorizare și control din diverse domenii, de la cele industriale și până la cele de monitorizare a mediului, a traficului, a clădirilor și a orașelor inteligente.

Un nod IoT în principiu trebuie să conțină: un element de procesare (element inteligent), modul de memorare/stocare a programului și a datelor, un canal de conectare la Internet, senzori sau interfețe de proces, o sursă de alimentare și eventual o interfață minimă pentru interacțiunea cu un utilizator. De multe ori majoritatea acestor componente sunt asigurate de un circuit microcontroler sau de o placă de dezvoltare destinată pentru utilizarea acelui microcontroler.

2. Obiectivul lucrării

Această lucrare de laborator prezintă primii pași în dezvoltarea de sisteme Internet of Things (IoT) folosind placa de dezvoltare ESP8266. Aceasta placa oferă o serie de facilități care îl fac potrivit pentru dezvoltarea unui nod IoT:

- are la bază un circuit de tip System-on-chip care presupune integrarea unui minisistem de calcul într-un singur circuit integrat
- are o interfață Wi-Fi minimală pe care s-a implementat stiva de protocoale TCP/IP necesară pentru conectarea plăcii la rețeaua Internet.

Această lucrare prezintă într-o formă pragmatică ce presupune realizarea unui nod IoT, de la conectarea unor senzori, programarea plăcii de dezvoltare și până la interacțiunea cu utilizatorul prin intermediul unui browser WEB.

3. Componente necesare

3.1. Circuitul ESP8266

Caracteristici:

- Procesor: Tensilica L106 32-bit RISC
- Frecvență: 80/160 MHz
- Tensiune de operare: 3.3V
- Memorie internă RAM: ~128 KB total (32 KB instrucțiuni, 80 KB date-utilizator, 16 KB sistem)
- Memorie flash: 4 MB (flash extern QSPI)
- Interfața de comunicare fără fir: WiFi 802.11 b/g/n (2.4 GHz)
- Interfețe seriale: UART, SPI, I2C (software), I2S

- Pini I/O digitale: 17 GPIO (multiplexate cu alte funcții)
- Intrare analogică: 1 canal, ADC 10-bit

3.2 Placa de dezvoltare NodeMCU cu ESP8266 System-on-Chip (SoC).

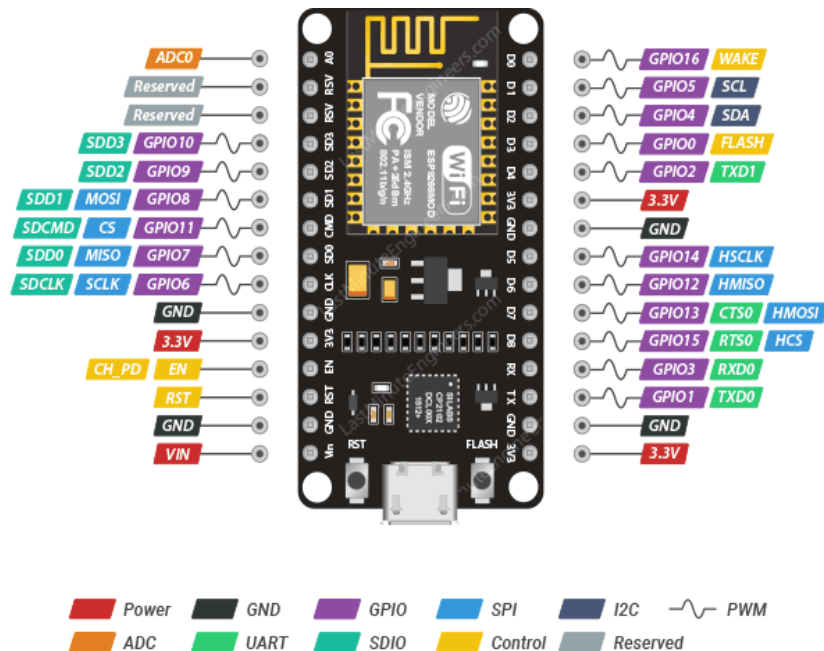


Figura 1.1 NodeMCU – ESP8266

Placa de dezvoltare NodeMCU se bazează pe modulul ESP8266 prin integrarea tuturor circuitelor de suport necesare:

- Interfață USB pentru alimentare și programare, conector micro USB
- Convertor USB-to-serial (chipset CP2102 sau CH340G)
- Regulator de tensiune
- Buton de reset (RST)
- Buton de flash
- LED încorporat (GPIO16)

Tabel 1. Pini plăcii NodeMCU

Pin	GPIO	Observații
D0	GPIO16	HIGH la boot , folosit pentru trezire din deep sleep
D1	GPIO5	
D2	GPIO4	
D3	GPIO0	conectat la butonul FLASH , bootează greșit dacă este LOW
D4	GPIO2	HIGH la boot , bootează greșit dacă este LOW
D5	GPIO14	
D6	GPIO12	
D7	GPIO13	
D8	GPIO15	necesar pentru boot, bootează greșit dacă este HIGH
RX	GPIO3	pin Rx , folosit pentru flashing și debugging
TX	GPIO1	pin Tx , folosit pentru flashing și debugging
CLK	GPIO6	conectat la memoria Flash
SD0	GPIO7	conectat la memoria Flash
CMD	GPIO11	conectat la memoria Flash
SD1	GPIO8	conectat la memoria Flash
SD2	GPIO9	conectat la memoria Flash
SD3	GPIO10	conectat la memoria Flash
A0	ADC0	pin de intrare analogică , nu poate fi configurat ca ieșire

3.3 Senzor de temperatura și umiditate DHT22 vs DHT11

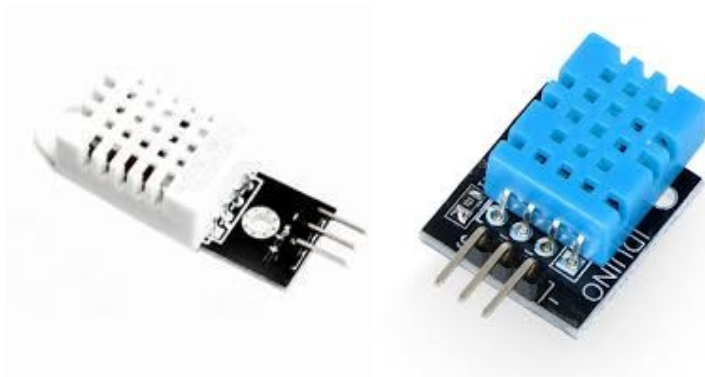


Figura 1.2 Senzor de temperatura și umiditate DHT22 (stânga) și DHT11 (dreapta)

Număr pin	Denumire	Descriere	Conectare la NodeMCU
1	VCC / +	Alimentare (3.3V - 5V)	Conectare la pinul 3.3V
2	DATE / OUT / S	Semnal de date	Conectat la orice pin digital
3	GND / -	Masă (Ground)	Conectat la pinul GND

3.4 Alte componente:

- Breadboard
- 3 fire pentru conexiunile senzorului
- Cablu alimentare/programare micro USB

4. Desfășurarea lucrării

Pașii de urmat:

4.1 Realizarea schemei hardware

Circuitul hardware este simplu, se conectează senzorul la placa de dezvoltare prin 3 pini: $V_{CC} +3.3V$, GND și portul digital D6 pentru citirea valorilor de temperatura și umiditate.

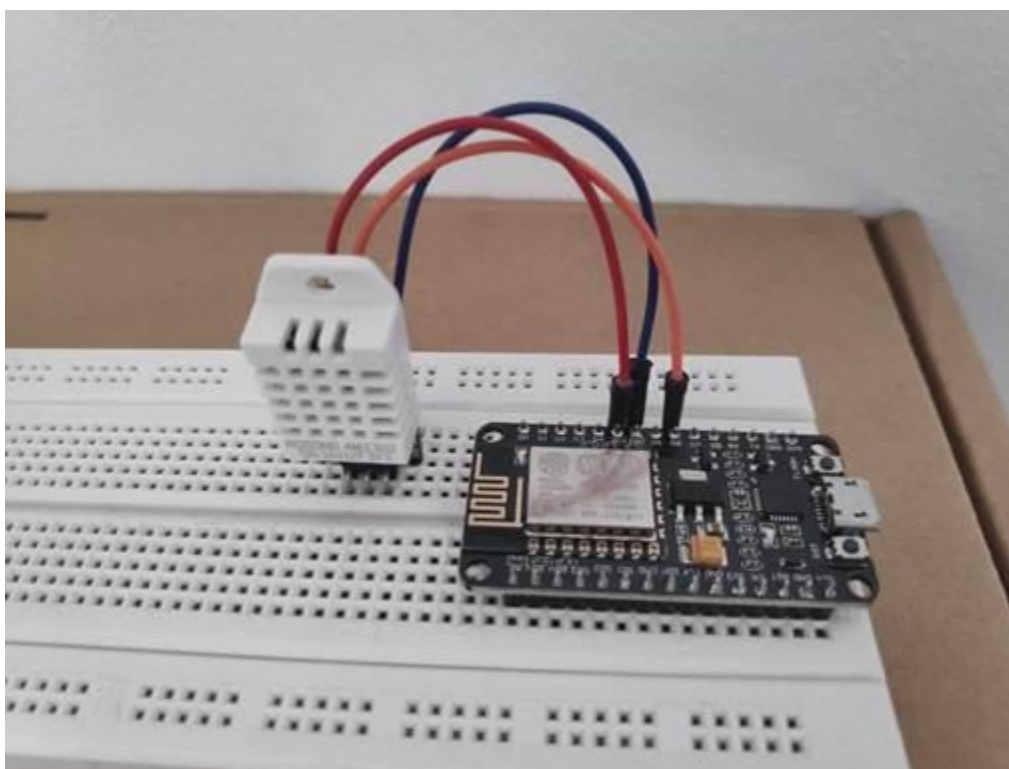


Figura 1.3 Circuitul hardware: senzor temperatura și NodeMCU

4.2 Utilizarea mediului de programare Arduino

4.2.1. Instalarea bibliotecilor necesare

Pentru a programa circuitul se folosește mediul de programare Arduino IDE; la acest mediu trebuie să se adauge biblioteca ESP8266 (pentru microcontroler) și biblioteca *DHT sensor library* (Adafruit) pentru senzorii folosiți

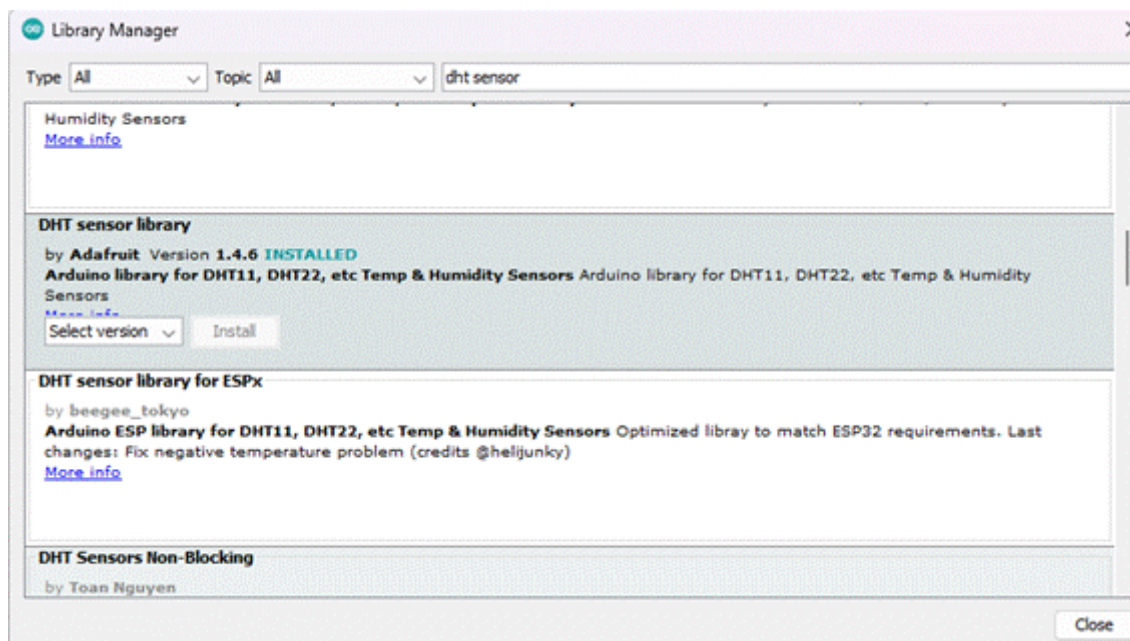


Figura 1.4 Librăria pentru senzorul DHT22

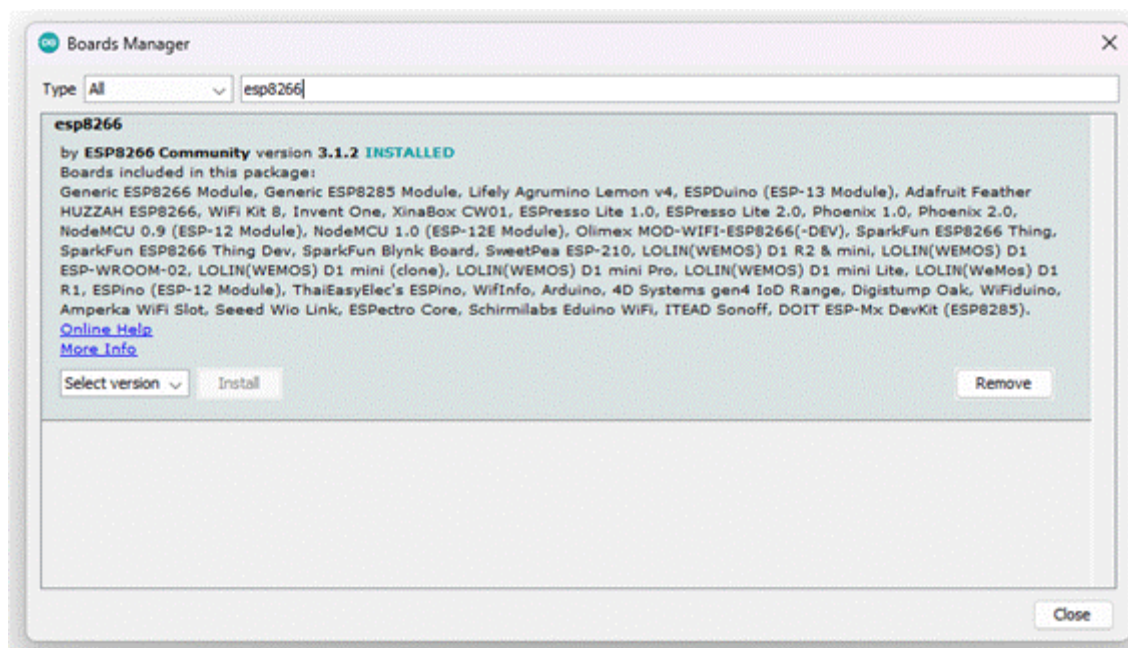


Figura 1.5 Librăria pentru microcontrolerul ESP8266*

*Observație: Pentru a instala librăria pentru ESP8266, trebuie setat URL-urile de mai jos în Arduino IDE -> File -> Preferences -> Additional boards manager URLs”. Se separă cele 2 URL-uri de mai jos cu virgulă.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

https://dl.espressif.com/dl/package_esp32_index.json

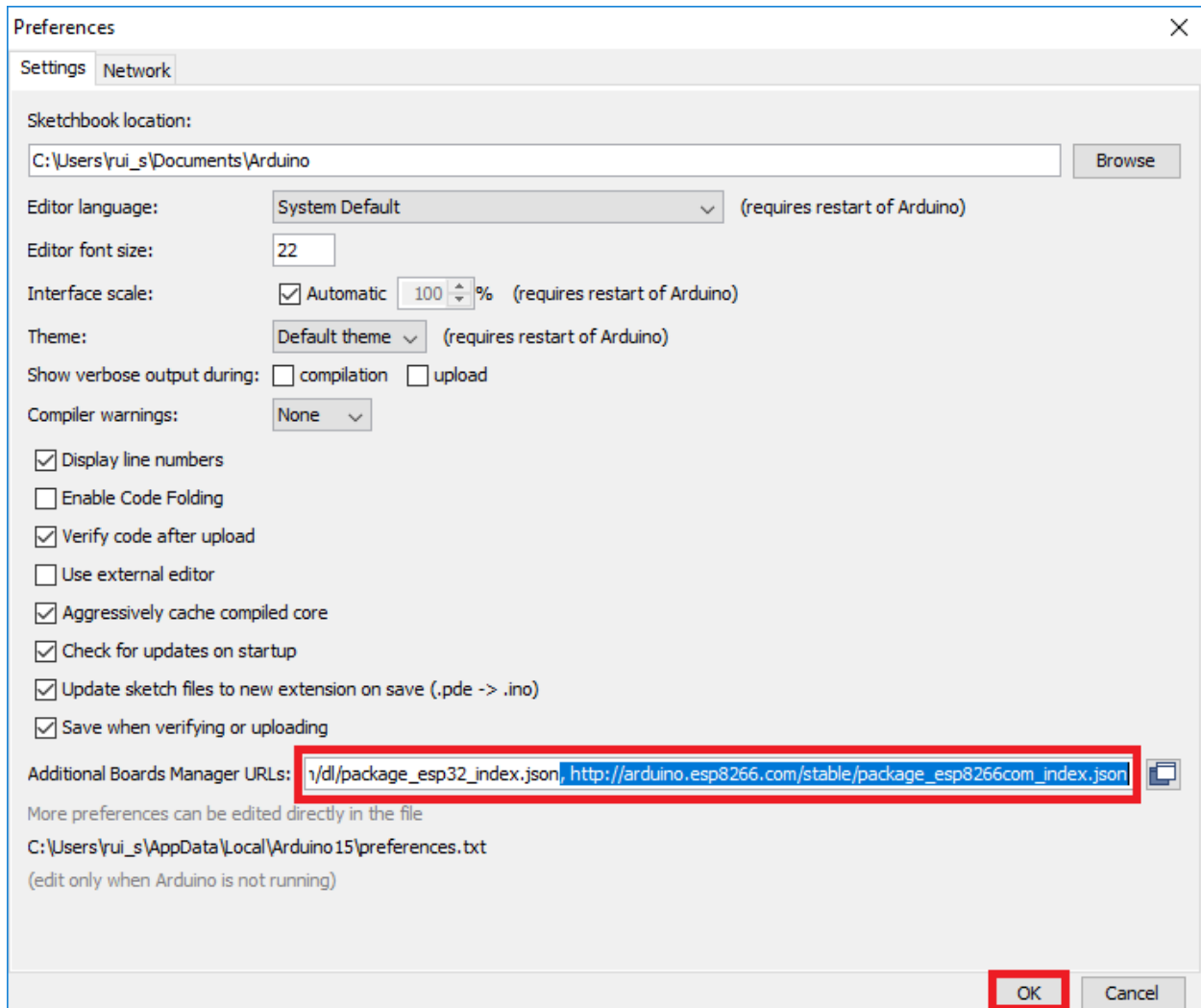


Figura 1.6 Setarea URL-urilor pentru librăria ESP8266

4.2.3 Programul de pe placa ESP

Pentru a citi și vizualiza datele de la senzor, acesta trebuie declarat prin specificarea pin-ului de date (D6) și tipul de senzor (DHT22 vs DHT11). Apoi, senzorul este activat în funcția *setup()* prin apelarea procedurii *dht.begin()*.

```

#include "DHT.h"
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
uint8_t DHTPin = D6;
DHT dht(DHTPin, DHTTYPE);
void setup()
{
pinMode(DHTPin, INPUT);
dht.begin();}

```

Valorile de temperatura și umiditate sunt citite prin intermediul procedurilor *dht.readTemperature()* și *dht.readHumidity()* și trimitem valorile către server prin funcția *SendHTML*.

```

float Temperature;
float Humidity;
void handle_OnConnect()
{
Temperature = dht.readTemperature();
Humidity     = dht.readHumidity();
server.send(200, "text/html", SendHTML(Temperature, Humidity));}

```

4.2.4 Comunicația pe interfața Wi-Fi

Vizualizarea datelor se va face într-o pagină web, iar pentru aceasta folosim capacitățile Wi-Fi ale SoC-ului ESP8266, disponibile prin bibliotecile *ESP8266WiFi* și *ESP8266WebServer*. Se specifică în cod numele rețelei wireless (SSID) și parola, iar conectarea se face prin apelarea funcției *Wifi.begin()*. Se creează un web server HTTP printr-o variabilă *ESP8266WebServer* și se pornește serverul prin procedura *server.begin()*.

```

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
/*Put your SSID & Password*/
const char* ssid = ""; // Enter SSID here
const char* password = ""; //Enter Password here
ESP8266WebServer server(80);
void setup()
{
//connect to your local wi-fi network
WiFi.begin(ssid, password);}

```

4.2.5 crearea paginii web pentru citirea datelor

Apoi cream o pagina HTML simpla în care sunt afișate în timp real temperatura și umiditatea.

```
String ptr = "<!DOCTYPE html> <html>\n";
ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-
scalable=no\">\n";
ptr += "<title>ESP8266 Weather Report</title>\n";
ptr += "<body>\n";
ptr += "<div id=\"webpage\">\n";
ptr += "<h1>ESP8266 NodeMCU Weather Report</h1>\n";
ptr += "<p>Temperature: ";
ptr += (float)Temperaturestat;
ptr += "\u00B0\"C</p>";
ptr += "<p>Humidity: ";
ptr += (float)Humiditystat;
ptr += "%</p>";
ptr += "</div>\n";
ptr += "</body>\n";
ptr += "</html>\n";
```

Pagina respectivă poate fi accesata de oricine din aceeași rețea prin introducerea adresei IP a serverului (Figura 1.8) într-un browser web. Codul complet este disponibil în **Anexa 1**.

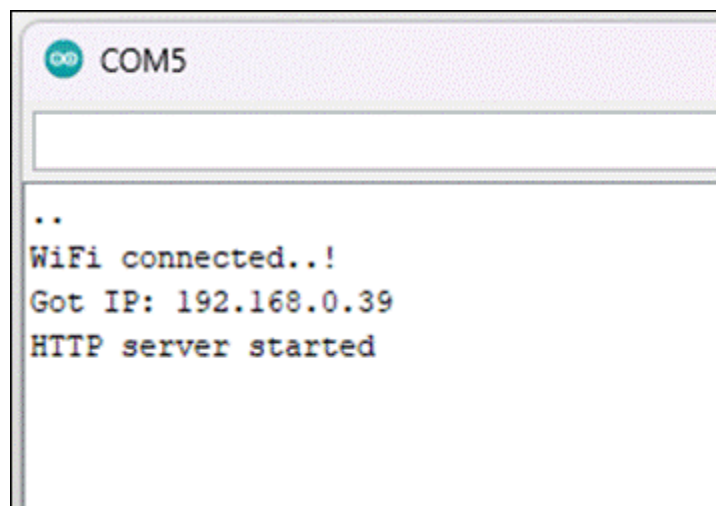


Figura 1.7 Afișarea adresei IP în Serial Monitor din Arduino IDE

ESP8266 NodeMCU Weather Report

Temperature: 29.10°C

Humidity: 44.50%

Figura 1.8 Pagina HTML pentru vizualizarea temperaturii și umidității

5. Exerciții

5.1. Crearea unui sistem care monitorizează temperatura și umiditatea, afișând datele pe o interfață web (tutorialul de la pagina 1).

- Conectați senzorul DHT22/DHT11 la ESP8266
- Creați o pagină web care afișează citirile senzorilor în timp real (codul sursa din Anexa 1)
- Implementați reîmprospătare automată la fiecare 5 secunde
- Adăugați stilizare CSS interfeței web

5.2. Se vor conecta diferite tipuri de componente și senzori la ESP8266, cum ar fi:

- Senzor de gaz
- Senzor ultrasonic
- Senzor de lumina
- Senzor audio (microfon)
- Senzor magnetic Hall
- LED-uri

Se va modifica codul sursa din **Anexa 1** pentru a afișa valorile colectate de la toți senzorii.

5.3. Se va implementa codul din **Anexa 2** pentru a verifica transferul bidirecțional de date. Circuit folosește un singur LED conectat la portul digital D1, printr-o rezistență de 220 - 500 Ohm.

5.4. Se va folosi platforma Arduino IoT Cloud (<https://cloud.arduino.cc/>) pentru a conecta placa NodeMCU la un serviciu de tip Cloud. Se folosește circuitul din exercițiul 4.1.

Pași de urmat:

- Din meniul din stânga se alege „Devices”, apoi “New Device”, “Compatible device”. Aici se configurează placa (Figura 1.9):
 - Device type: ESP8266 - NodeMCU 1.0
 - Nume
 - Se salvează PDF-ul cu Device ID și Secret Key.

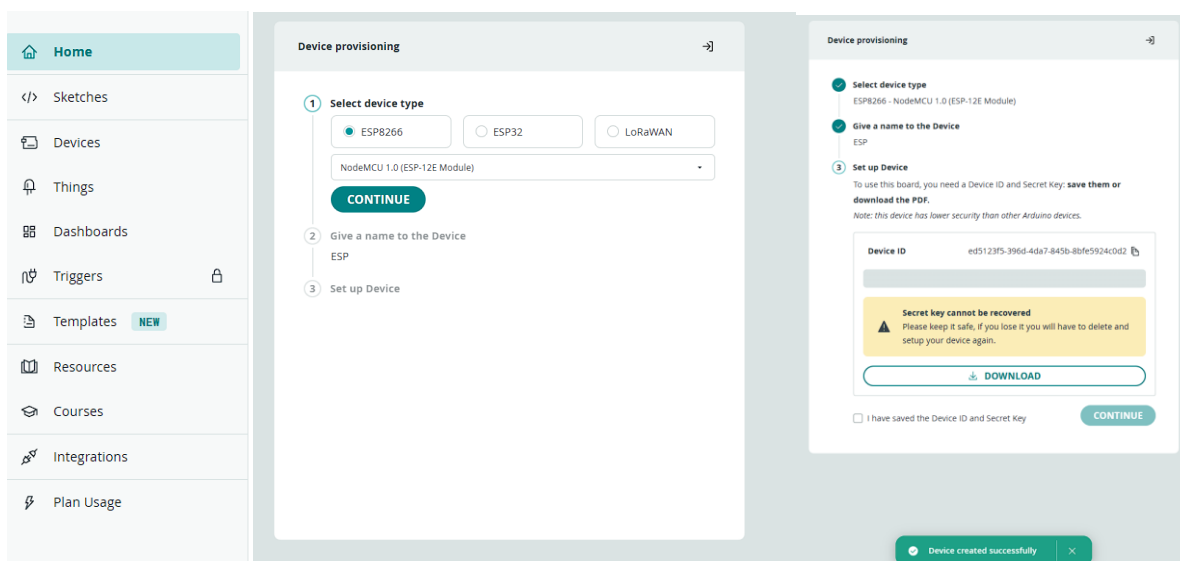


Figura 1.9 Setarea dispozitivului NodeMCU

2. Valorile pe care vrem sa le monitorizam (temperatura și umiditate) se adaugă în meniul „Things” -> “New Thing” (Figura 1.10). Se adaugă 2 variabile noi pentru temperatură și umiditate (butonul “Add”), se introduce numele, tip de date *float*, citite periodic la 1 secundă. În aceeași fereastră, se asociază dispozitivul (opțiunea “Associated Device” din partea dreaptă).

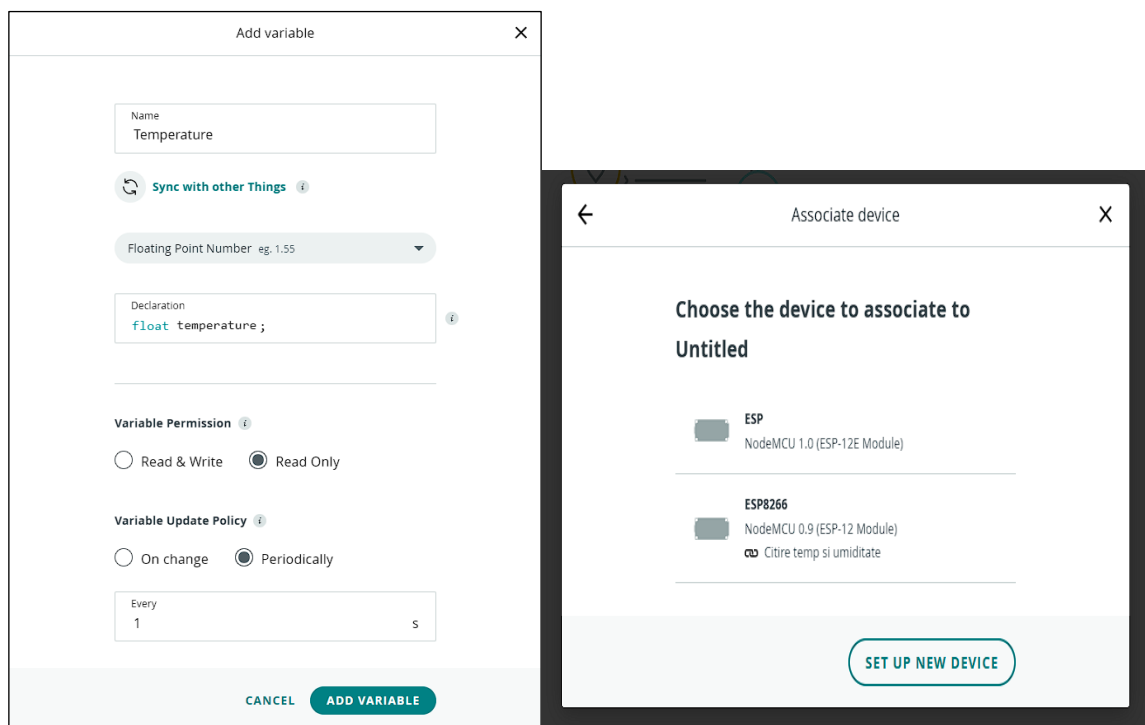


Figura 1.10 Adăugare variabilă în meniu „Thing”

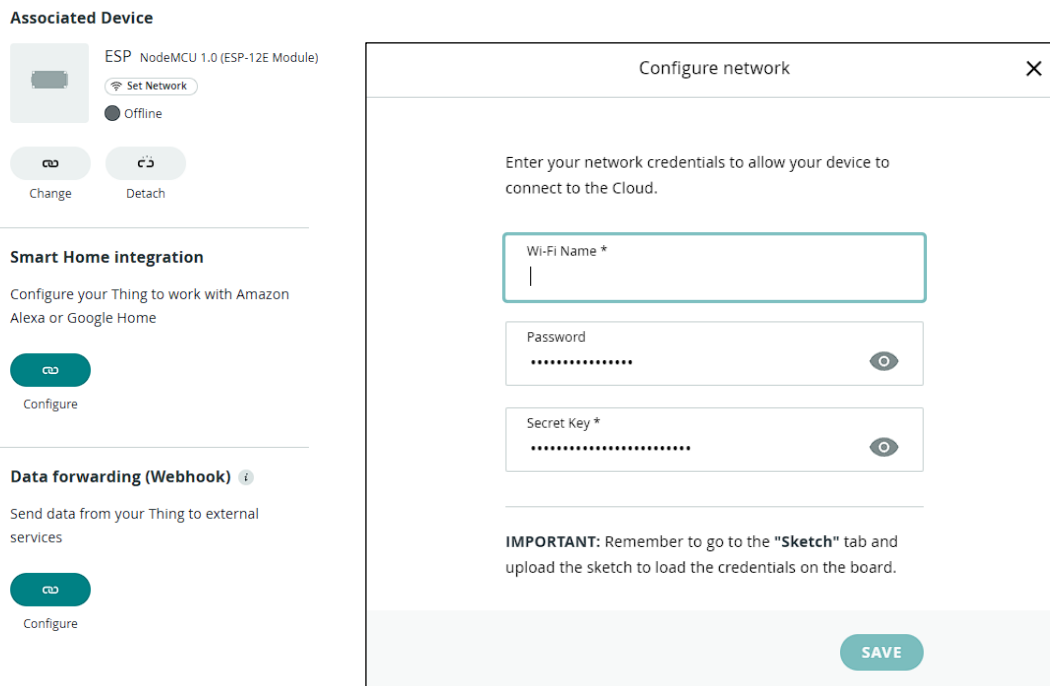


Figura 1.11 Setarea rețelei WiFi la care se conectează NodeMCU

3. În aceeași fereastră, se alege “Set Network” și se setează rețeaua Wi-Fi la care se conectează dispozitivul în meniul (Figura 1.11)
4. Codul sursa se adaugă din meniul „Sketch”. Se integrează în codul generat automat instrucțiunile pentru senzorul DHT: librăria DHT și definirea senzorului, apelarea procedurii de activare a senzorului și citirea valorilor de temperatura și umiditate. Tot aici va apărea o notificare despre Cloud Agent care va trebui instalat pentru a putea încărca codul sursa din browser-ul web pe dispozitivul ESP8266 (<https://cloud.arduino.cc/download-agent>)

```
#include <DHT.h>
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
uint8_t DHTPin = D6;
DHT dht(DHTPin, DHTTYPE);
void setup() {
  pinMode(DHTPin, INPUT);
  dht.begin();
}
void loop() {
  temperatura = dht.readTemperature();
  umiditate = dht.readHumidity();}
```

Codul sursa complet este în **Anexa 3**.

5. În meniul „Dashboard” se pot vizualiza datele colectate de la senzor. Adăugăm aici 2 widget-uri, pentru temperatura și umiditate. Fiecare widget trebuie asociat cu un “Thing” (variabilă) de la pasul 2. Apoi se încarcă codul sursa (meniul „Sketch”) și datele vor fi afișate în timp real în dashboard (Figura 1.12).

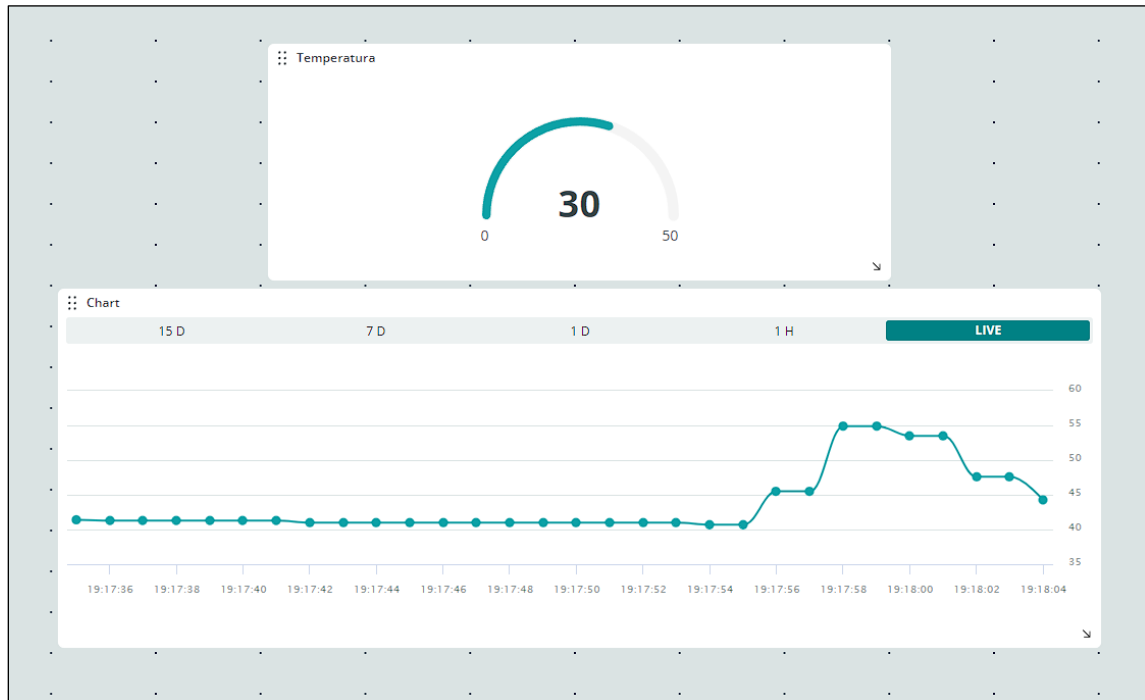


Figura 1.12 Vizualizarea valorilor colectate de la senzor în timp real

Anexa 1

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "DHT.h"

// #define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

const char* ssid = ""; // WiFi SSID
const char* password = ""; // WiFi Password
ESP8266WebServer server(80);

uint8_t DHTPin = D6;
DHT dht(DHTPin, DHTTYPE);

float Temperature;
float Humidity;

void setup() {
  Serial.begin(115200);
```

```

delay(100);
pinMode(DHTPin, INPUT);
dht.begin();
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print("."); }
Serial.println("");
Serial.println("WiFi connected..!");
Serial.print("Got IP: "); Serial.println(WiFi.localIP());
server.on("/", handle_OnConnect);
server.onNotFound(handle_NotFound);
server.begin();
Serial.println("HTTP server started");}

void loop() {
  server.handleClient();}

void handle_OnConnect() {
  Temperature = dht.readTemperature();
  Humidity = dht.readHumidity();
  server.send(200, "text/html", SendHTML(Temperature,Humidity));}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");}

String SendHTML(float Temperaturestat,float Humiditystat){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta   name=\"viewport\"   content=\"width=device-width,   initial-scale=1.0,   user-
scalable=no\">\n";
  ptr += "<title>ESP8266 Weather Report</title>\n";
  ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
  ptr += "p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  ptr += "</style>\n";
  ptr += "<script>\n";
  ptr += "setInterval(loadDoc,200);\n";
  ptr += "function loadDoc() {\n";
  ptr += "var xhttp = new XMLHttpRequest();\n";

```

```

ptr += "xhttp.onreadystatechange = function() {\n";
ptr += "if (this.readyState == 4 && this.status == 200) {\n";
ptr += "document.getElementById(\"webpage\").innerHTML =this.responseText}\n";
ptr += "};\n";
ptr += "xhttp.open(\"GET\", \"/\", true);\n";
ptr += "xhttp.send();\n";
ptr += "}\n";
ptr += "</script>\n";
ptr += "</head>\n";
ptr += "<body>\n";
ptr += "<div id=\"webpage\">\n";
ptr += "<h1>ESP8266 NodeMCU Weather Report</h1>\n";
ptr += "<p>Temperature: ";
ptr += (float)Temperaturestat;
ptr += "\u00B0\"C</p>";
ptr += "<p>Humidity: ";
ptr += (float)Humiditystat;
ptr += "%</p>";
ptr += "</div>\n";
ptr += "</body>\n";
ptr += "</html>\n";
return ptr;

```

Anexa 2

```

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
const char* ssid = ""; // WiFi SSID
const char* password = ""; // WiFi Password
ESP8266WebServer server(80);
uint8_t LEDPin = D1;
bool ledState = false;
void setup() {
  Serial.begin(115200);
  delay(100);
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
  Serial.println("Connecting to ");

```

```

Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");}
Serial.println("");
Serial.println("WiFi connected..!");
Serial.print("Got IP: "); Serial.println(WiFi.localIP());
server.on("/", handle_OnConnect);
server.on("/led", handle_LED);
server.onNotFound(handle_NotFound);
server.begin();
Serial.println("HTTP server started");}

void loop() {
  server.handleClient();}

void handle_OnConnect() {
  server.send(200, "text/html", SendHTML(ledState));}

void handle_LED() {
  ledState = !ledState;
  digitalWrite(LEDPin, ledState ? HIGH : LOW);
  server.sendHeader("Location", "/");
  server.send(303);}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");}

String SendHTML(bool ledStatus){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta  name=\"viewport\"  content=\"width=device-width,  initial-scale=1.0,  user-
scalable=no\">\n";
  ptr += "<title>ESP8266 LED Control</title>\n";
  ptr += "<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr += "body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
  ptr += "p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  ptr += ".button {display: block;width: 200px;background-color: #1abc9c;border: none;color: white;padding: 13px
30px;text-decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
  ptr += ".button-on {background-color: #1abc9c;}\n";
  ptr += ".button-on:active {background-color: #16a085;}\n";
  ptr += ".button-off {background-color: #e74c3c;}\n";
  ptr += ".button-off:active {background-color: #c0392b;}\n";

```

```

ptr += ".led-status {font-size: 20px;color: #34495e;margin-bottom: 20px;}\n";
ptr += "</style>\n";
ptr += "</head>\n";
ptr += "<body>\n";
ptr += "<div id=\"webpage\">\n";
ptr += "<h1>ESP8266 LED Control</h1>\n";
ptr += "<div class=\"led-status\">LED is currently: ";
ptr += (ledStatus ? "ON" : "OFF");
ptr += "</div>\n";
ptr += "<a class=\"button \"";
ptr += (ledStatus ? "button-off" : "button-on");
ptr += "\" href=\"/led\">Turn ";
ptr += (ledStatus ? "OFF" : "ON");
ptr += "</a>\n";
ptr += "</div>\n";
ptr += "</body>\n";
ptr += "</html>\n";
return ptr;

```

Anexa 3

```

#include "thingProperties.h"
#include <DHT.h>
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
uint8_t DHTPin = D6;
DHT dht(DHTPin, DHTTYPE);
void setup() {
  Serial.begin(9600);
  delay(1500);
  pinMode(DHTPin, INPUT);
  dht.begin();
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}
void loop() {
  ArduinoCloud.update();
  temperatura = dht.readTemperature();
  umiditate = dht.readHumidity();
}

```

Lucrare de laborator 2: Rețea de Senzori Fără Fir cu Raspberry Pi 5

1. Introducere

Această lucrare de laborator introduce dezvoltarea unei rețele cuprinzătoare de senzori fără fir (WSN) utilizând Raspberry Pi 5 ca nod central. Proiectul se concentrează pe integrarea diversilor senzori pentru a crea un sistem inteligent de monitorizare, permițând o înțelegere practică a arhitecturii IoT, a protocoalelor de comunicație wireless, achiziției de date de la senzori și procesării datelor în timp real.

Obiective:

- Înțelegerea protocoalelor de comunicație wireless și a aplicațiilor lor practice
- Implementarea sistemelor de achiziție a datelor de la senzori
- Dezvoltarea abilităților de programare pe platforma Raspberry Pi 5
- Învățarea protocolului MQTT și utilizarea acestuia pentru comunicarea în IoT
- Crearea și configurarea unei rețele complete de senzori fără fir, cu vizualizarea datelor

2. Specificații

Componente Hardware:

- Raspberry Pi 5 8GB RAM, flash drive USB pentru sistemul de operare.

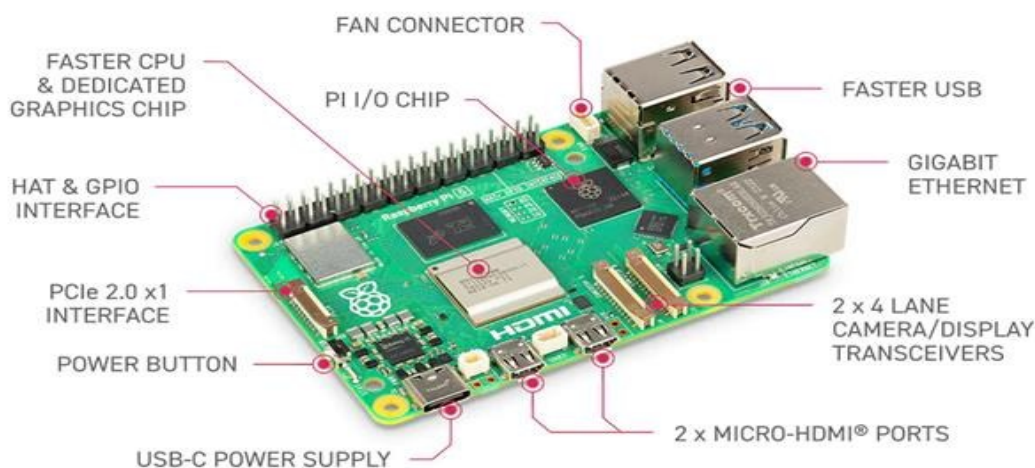


Figura 2.1 Raspberry Pi 5

Categorie	Detalii
Procesor (CPU)	Broadcom BCM2712, quad-core Arm Cortex-A76 pe 64 de biți, 2.4GHz
Memorie (RAM)	8GB LPDDR4X-4267 SDRAM
Grafică (GPU)	VideoCore VII GPU cu suport pentru OpenGL ES 3.1, Vulkan 1.2
Video	Două ieșiri HDMI® dual 4Kp60 cu suport HDR; Decodor 4Kp60 HEVC
Conectivitate	Wi-Fi® dual-band 802.11ac; Bluetooth 5.0 / BLE; Ethernet Gigabit
Porturi USB	2x porturi USB 3.0 (5Gbps) 2x porturi USB 2.0
Expansiune si periferice	Interfață PCIe 2.0 x1 pentru periferice rapide (ex. SSD-uri M.2); Header GPIO standard cu 40 de pini
Interfețe multimedia	2x transceivere MIPI cu 4 benzi pentru camere/afișaje
Stocare	Slot pentru card microSD cu suport pentru modul de mare viteză SDR104
Alimentare	5V/5A DC prin USB-C, cu suport pentru Power Delivery
Caracteristici adiționale	Buton de pornire; Ceas în timp real (RTC); Port UART pentru depanare

- Senzori: senzor de Temperatură/Umiditate DHT22/DHT11 conectat la modulul ESP8266 (Wi-Fi)
- Module Wireless: Modul ESP8266-NodeMCU
- Componente Adiționale: breadboard, fire de legătură

Cerințe Software:

- Raspberry Pi OS (64-bit)
- Python 3.9+
- Librării Python necesare: paho-mqtt
- Broker MQTT Mosquitto

3. Tutorial: Rețea IoT cu Raspberry Pi 5 și ESP8266-NodeMCU

Pasul 1: Configurarea Inițială a Raspberry Pi 5

1. Conectați Raspberry Pi 5 la un monitor (prin adaptor microHDMI), conectați o tastatură și un mouse. Asigurați-vă că unitatea USB este conectată într-un port albastru înainte de a alimenta placa.

2. Actualizați sistemul de operare

```
sudo apt update && sudo apt upgrade -y
```

3. Instalați pachetele necesare

```
sudo apt install mosquitto mosquitto-clients
```

4. Instalați librăriile Python.

```
pip3 install paho-mqtt
```

Pasul 2: Configurarea Broker-ului MQTT

a. Configurați fișierul de configurare MQTT:

```
sudo nano /etc/mosquitto/conf.d/default.conf
```

b. Adăugați următorul conținut:

```
listener 1883
```

```
allow_anonymous true
```

c. Reporniți serviciul:

```
sudo systemctl restart mosquitto
```

d. Testați broker-ul MQTT și conexiunea.
Deschideți un terminal și rulați următoarea comandă "subscribe":

```
mosquitto_sub -d -t "testTopic"
```

Deschideți un alt terminal și rulați următoarea comandă "publish":

```
mosquitto_pub -d -t "testTopic" -m "Hello world!"
```

e. Într-o fereastră de terminal, rulați următoarea comandă pentru a afla adresa IP a Raspberry Pi:

```
sudo hostname -I
```

Pasul 3: Integrarea Senzorului Wi-Fi (DHT cu ESP8266)

a. În Arduino IDE, instalați următoarea librărie din Managerul de Librării: PubSubClient.

b. Utilizați codul din Anexa 1 pentru ESP8266-NodeMCU.

c. Încărcați codul în ESP8266-NodeMCU.

d. Pe Raspberry Pi, creați un fișier nou cu extensia .py (de ex. main.py) și adăugați codul din Anexa 2.

e. Rulați codul .py într-o fereastră de terminal cu următoarea comandă:

```
python3 main.py
```

Pasul 4: Testarea Sistemului

a. Încărcați codul pentru ESP8266 în modulul dvs. (pasul 3.c).

b. Rulați scriptul Python pe Raspberry Pi (pasul 3.e).

c. Monitorizați mesajele MQTT într-o fereastră de terminal:

```
mosquitto_sub -h localhost -t "sensors/#"
```

4. Exerciții

4.1. Uurmați tutorialul și conectați ESP8266-NodeMCU la broker-ul MQTT de pe Raspberry Pi 5.

4.2. Configurați un nume de utilizator și parolă pentru broker-ul MQTT mosquitto. Actualizați codul Arduino pentru ESP8266-NodeMCU cu noile credențiale de autentificare.

Pași:

Pe Raspberry Pi, creați o parolă pentru un utilizator specific:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd your_username
```

Editați fișierul de configurare mosquitto:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Adăugați următoarele linii de configurare:

```
allow_anonymous false
```

```
password_file /etc/mosquitto/passwd
```

Reporniți serviciul mosquitto:

```
sudo systemctl restart mosquitto
```

Testați noua configurație în 2 ferestre de terminal:

```
mosquitto_sub -h localhost -t "test/topic" -u "your_username" -P "your_password"
```

```
mosquitto_pub -h localhost -t "test/topic" -m "Hello, Secure MQTT!" -u "your_username" -P "your_password"
```

Actualizați codul Arduino pentru ESP8266-NodeMCU:

```
const char* mqtt_user = ""; // Nume utilizator MQTT
```

```
const char* mqtt_password = ""; // Parolă MQTT
```

```
client.connect("ESP8266", mqtt_user, mqtt_password)
```

4.3. Modificați codul sursă existent în Arduino IDE astfel încât fiecare mesaj să fie formatat ca un JSON (Indiciu: veți avea nevoie de o librărie JSON, de exemplu: ArduinoJson).

Anexa 1

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
#include <DHT.h>
```

```
#define DHT_PIN 6
```

```
#define DHT_TYPE DHT11 // sau DHT22
```

```
const char* ssid = ""; // WiFi SSID
```

```
const char* password = ""; // WiFi password
```

```
const char* mqtt_server = ""; // IP Raspberry Pi - MQTT Broker
```

```
WiFiClient espClient;
```

```
PubSubClient client(espClient);
```

```

DHT dht(DHT_PIN, DHT_TYPE);
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
            client.subscribe("sensors/commands");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
void setup() {
    Serial.begin(115200);
    dht.begin();
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    float temp = dht.readTemperature();
    float humidity = dht.readHumidity();
    if (isnan(temp) || isnan(humidity)) {

```

```

Serial.println("Failed to read from DHT sensor!");
delay(2000);
return;}

String data = "{" + "temperature\":" + String(temp) + ",\\" + "humidity\":" + String(humidity) + "}";
// Debug output
Serial.print("Sensing: Temp=");
Serial.print(temp);
Serial.print("°C, Humidity=");
Serial.print(humidity);
Serial.println("%");
Serial.print("Publishing: ");
Serial.println(data);
// Publish to MQTT topic
client.publish("sensors/dht", data.c_str());
delay(10000); // Wait 10 seconds between readings}

```

Anexa 2

```

import paho.mqtt.client as mqtt
import json
import time

class SensorHub:
    def __init__(self):
        self.client = mqtt.Client()
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.sensor_data = {}

    def on_connect(self, client, userdata, flags, rc):
        print("Connected with result code " + str(rc))
        client.subscribe("sensors/#")

    def on_message(self, client, userdata, msg):
        try:
            topic = msg.topic
            payload = json.loads(msg.payload.decode())
            self.sensor_data[topic] = payload
            print(f"Received: {topic} - {payload}")
        except Exception as e:
            print(f"Error processing message: {e}")

```

```
def start(self):
    self.client.connect("localhost", 1883, 60)
    self.client.loop_start()
def display_data(self):
    while True:
        print("Current Sensor Data:")
        for sensor, data in self.sensor_data.items():
            print(f'{sensor}: {data}')
            time.sleep(5)
if __name__ == "__main__":
    hub = SensorHub()
    hub.start()
    hub.display_data()
```

Lucrare de laborator 3: Stocarea și analiza datelor de proces în baze de date Time-Series

1. Obiectivele lucrării

- Înțelegerea conceptului de baze de date orientate pe serii temporale (*Time-Series Databases – TSDB*) utilizate în sistemele IIoT;
- Configurarea unei surse de date de tip Arduino/ESP8266 în platforma **InfluxDB Cloud** (găzduită pe infrastructura Amazon Web Services);
- Crearea unui depozit de date (*bucket*) și generarea token-ului de securitate pentru autorizarea scrierilor;
- Dezvoltarea și încărcarea programului C++ pe placa NodeMCU (ESP8266) pentru transmiterea periodică a temperaturii și umidității citite de la senzorul DHT22;
- Interogarea și vizualizarea seriilor de timp sub formă tabelară și grafică în interfața **Data Explorer** a platformei InfluxDB.

2. Fundament teoretic: modelul de date în InfluxDB

Spre deosebire de bazele de date relaționale (SQL), InfluxDB stochează datele sub formă de **puncte temporale** (*Data Points*) organizate conform modelului *Line Protocol*:

- **Measurement (Măsurătoare):** definește contextul general al tabelii (de exemplu, "masuratori");
- **Tags (Etichete de metadata):** perechi cheie-valoare de tip text, indexate în memorie pentru căutare și filtrare rapidă (de exemplu, device = "ESP8266" sau locație = "Laborator");
- **Fields (Câmpuri de date):** valorile numerice măsurate efectiv de senzori, neindexate (de exemplu, Temperatura = 24.5, Umiditate = 55.0);
- **Timestamp (Marcaj temporal):** timpul exact al achiziției, adăugat automat de platformă sau generat prin sincronizare NTP.

3. Desfășurarea lucrării

Pasul 1: Configurarea sursei de date în InfluxDB Cloud

1. Conectați-vă la contul de InfluxDB Cloud;
2. În meniul din stânga, selectați „Load Data” → „Sources”, apoi alegeți platforma Arduino (Figura 3.1);
3. Creați un container de date nou (bucket), care va reprezenta depozitul logic pentru toate datele colectate de la senzori (Figura 3.2);
4. Denumiți noul bucket *laborator_senzori* (sau numele specificat de cadrul didactic) și setați politica de retenție a datelor (*retention period*).

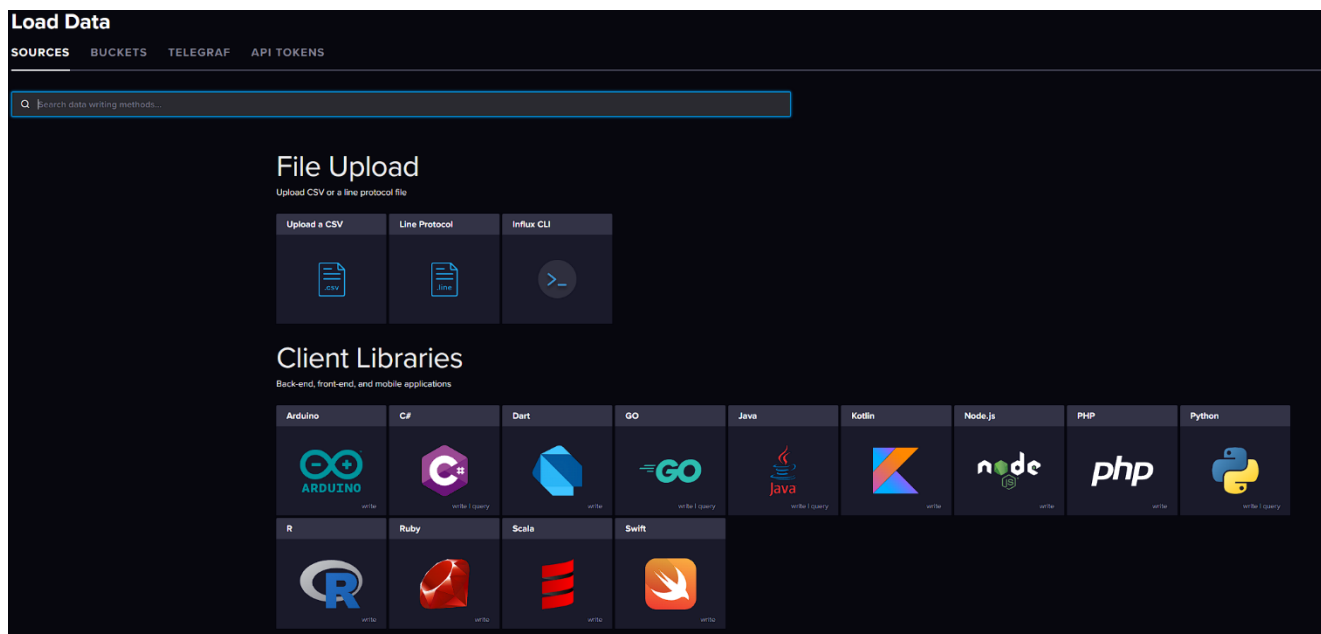


Figura 3.1 Configurare sursa de date in InfluxDB

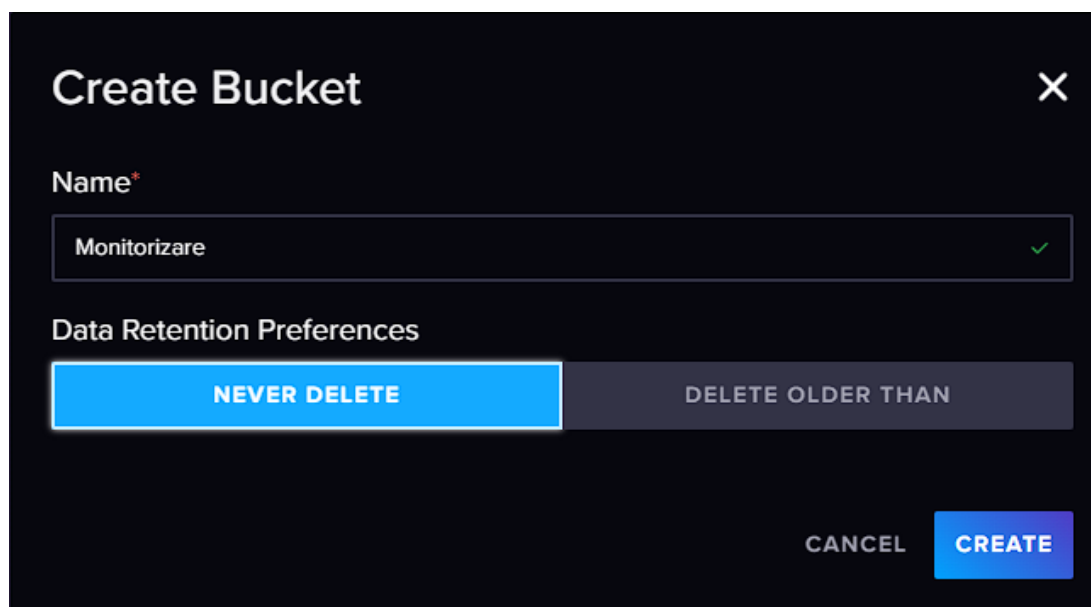


Figura 3.2 Crearea containerului de date (en. bucket)

Pasul 2: Preluarea parametrilor de conectare și a șablonului de cod

Platforma InfluxDB va genera automat un cod sursă șablon (*template*) ce conține credențialele de acces (Figura 3.3 și Figura 3.4).

Setting Up

Arduino

5 minutes

- Overview
- Prepare Arduino IDE
- Install Dependencies
- Initialize Client
- Write Data
- Finish

A **bucket** is used to store time-series data. Here is a list of your existing buckets. You can select one to use for the rest of the tutorial, or create one below.

BUCKET

+ CREATE BUCKET

Temp si umiditate

Configure an InfluxDB profile

Next we'll need to configure the client and its initial connection to InfluxDB. InfluxDB Cloud uses Tokens to authenticate API access. We've created an all-access token for you for this set up process.

Paste the following snippet into a blank Arduino sketch file.

```

if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

```

COPY TO CLIPBOARD

Note: you will need to set the `WIFI_SSID` and `WIFI_PASSWORD` variables to the correct values for your wifi router.

PREVIOUS

NEXT

Figura 3.3 Cod generat automat de InfluxDB pentru configurarea sursei de date

Setting Up

Arduino

5 minutes

- Overview
- Prepare Arduino IDE
- Install Dependencies
- Initialize Client
- Write Data
- Finish

Write Data

To start writing data, append the lines of code to add tags to the Point at the end of the void `setup()` function.

```

void setup() {
// ... code in setup() from Initialize Client

// Add tags to the data point
sensor.addTag("device", DEVICE);
sensor.addTag("ssid", WiFi.SSID());
}

```

COPY TO CLIPBOARD

Add the following `loop()` code snippet to your sketch

```

void loop() {
// Clear fields for reusing the point. Tags will remain the same as set above.
sensor.clearFields();

// Store measured value into point
// Report RSSI of currently connected network
sensor.addField("rssi", WiFi.RSSI());

// Print what are we exactly writing
Serial.print("Writing: ");
Serial.println(sensor.toLineProtocol());
}

```

COPY TO CLIPBOARD

In the above code snippet, we retrieve the RSSI (Received Signal Strength Indicator) of your wifi connection and write it to InfluxDB using the client.

Figura 3.4 Cod generat automat pentru stocarea datelor in InfluxDB

Copiați din această pagină cei patru parametri de configurare:

- INFLUXDB_URL – adresa URL a serverului InfluxDB Cloud (ex. <https://eu-central-1-1.aws.cloud2.influxdata.com>);
- INFLUXDB_TOKEN – token-ul API generat pentru autentificare;
- INFLUXDB_ORG – identificatorul sau adresa de e-mail a organizației;
- INFLUXDB_BUCKET – numele bucket-ului creat (laborator_senzori).

Pasul 3: Instalarea bibliotecilor în Arduino IDE

Pentru compilarea programului, deschideți **Arduino IDE** și instalați din meniul *Library Manager* (Sketch → Include Library → Manage Libraries...):

1. Biblioteca DHT sensor library (Adafruit) – pentru citirea senzorului DHT22;
2. Biblioteca ESP8266 Influxdb (de Markus Schulz) – pentru gestionarea conexiunii și scrierea punctelor de date în InfluxDB (Figura 3.5).

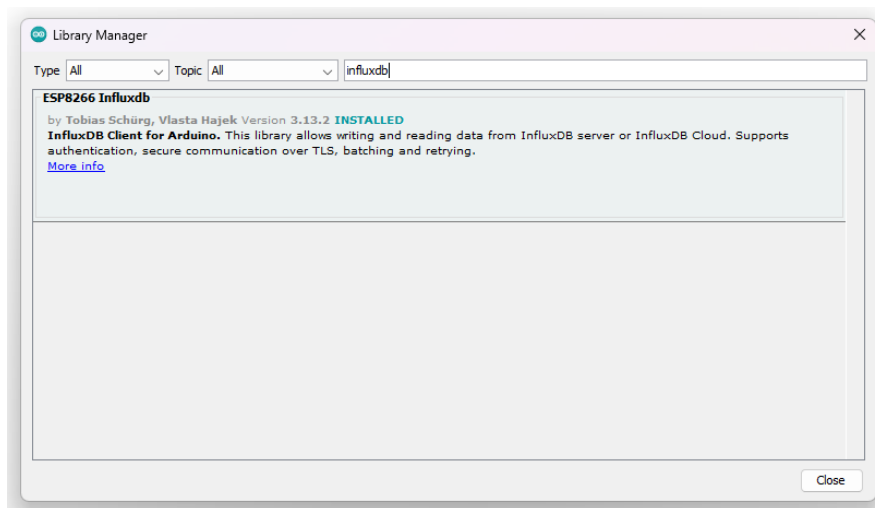


Figura 3.5 Biblioteca pentru InfluxDB

Pasul 4: Implementarea și încărcarea codului sursă

Conectați senzorul DHT22 la placa NodeMCU conform schemei hardware din Lucrarea 1 (pinul de date conectat la **D6 / GPIO12**).

Pentru crearea unui nou tabel în baza de date folosim tipul de date Point, iar pentru scrierea valorilor numerice măsurate folosim funcția `sensor.addField()`:

```
#include "DHT.h"
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// DHT Sensor
uint8_t DHTPin = D6;
// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);
float Temperature;
```

```
float Humidity;
void setup() {
  pinMode(DHTPin, INPUT);
  dht.begin(); }
```

```
// Data point
Point sensor("masuratori");
void setup() {
  // Add tags
  sensor.addTag("device", DEVICE);}
void loop() {
  Temperature = dht.readTemperature(); // Gets the values of the temperature
  Humidity = dht.readHumidity(); // Gets the values of the humidity
  // Store measured value into point
  sensor.clearFields();
  // add columns for temperature and humidity
  sensor.addField("Temperatura", Temperature);
  sensor.addField("Umiditate", Humidity); }
```

Platforma adaugă automat o coloană pentru marcajul de timp (timestamp) la fiecare punct inserat. Programul complet și gata de încărcare pe placă este disponibil în Anexa 1.

Pasul 5: Vizualizarea și analiza datelor în InfluxDB Cloud

După încărcarea programului pe placă, deschideți interfața web InfluxDB și accesați meniul „Data Explorer”:

1. Vizualizarea tabelară (Figura 3.6):

- Selectați bucket-ul laborator_senzori, măsurătoarea masuratori și câmpul Temperatura;
- Schimbați modul de afișare din colțul din dreapta-sus în **Table** pentru a vedea fiecare înregistrare cu timestamp-ul adăugat automat de platformă.

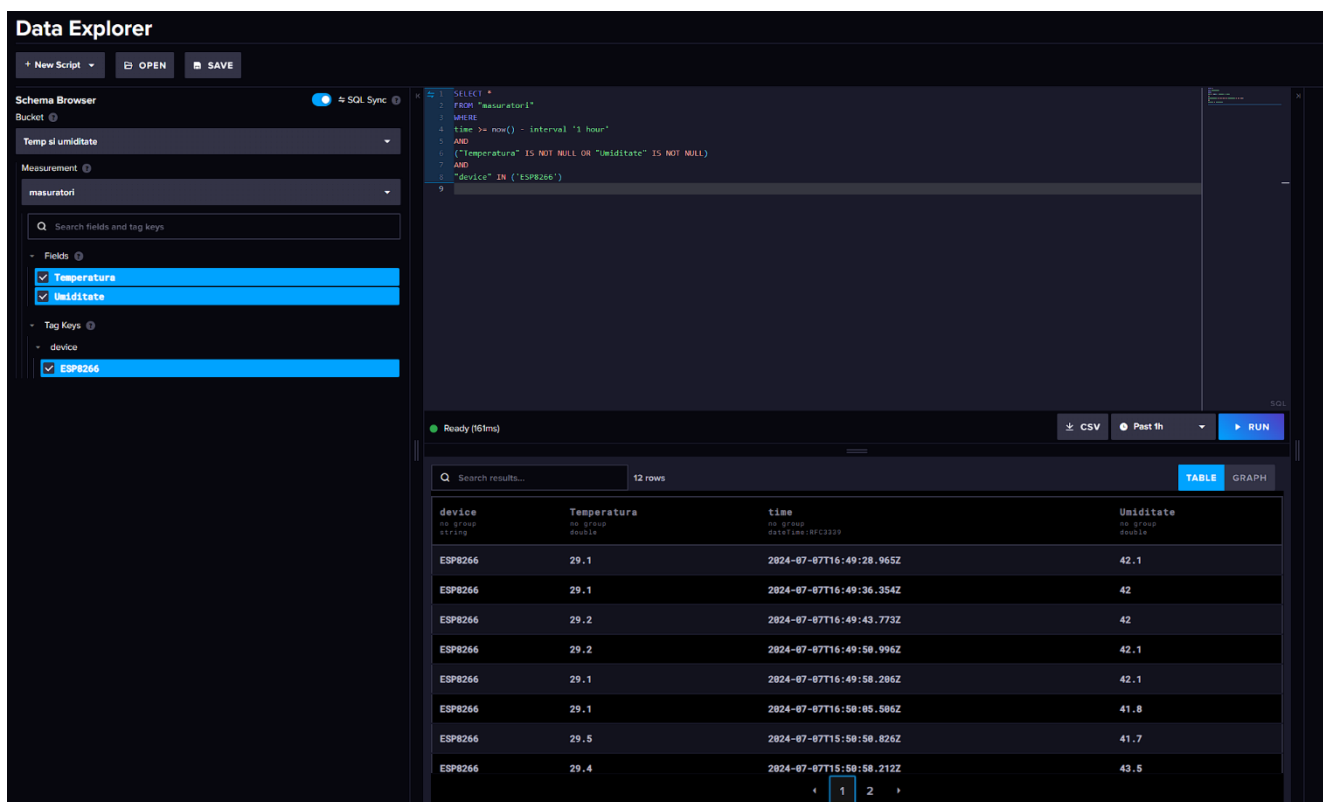


Figura 3.6 Vizualizarea datelor sub forma de tabel

2. Vizualizarea grafică în timp real (Figura 3.7):

- Selectați modul de afișare **Graph**;
- Selectați ambele câmpuri (Temperatura și Umiditate) pentru a urmări evoluția simultană a celor două mărimi în timp real;
- Salvați graficul într-un **Dashboard** dedicat prin butonul „Save as Dashboard Cell”.



Figura 3.7 Vizualizarea datelor sub forma de grafic

4. Exerciții și teme practice

- 4.1. Adăugarea unui tag de stare: Modificați codul sursă astfel încât să adăugați un tag dinamic *stare_climat*, care ia valoarea "Normal" dacă temperatura este $\leq 26^{\circ}\text{C}$ și "Cald" dacă temperatura depășește 26°C . Observați în *Data Explorer* cum se pot filtra datele pe baza acestui tag.
- 4.2. Construirea unui panou complet (Dashboard): În secțiunea *Dashboards*, creați un panou ce conține:
 - Un grafic liniar pentru temperatură și umiditate;
 - Un indicator de tip ceas (*gauge*) pentru valoarea instantanee a temperaturii, cu praguri colorate;
 - O celulă de tip *Single Stat* care afișează calitatea conexiunii Wi-Fi (Semnal_WiFi).
- 4.3. Răspunde-ți la următoarele întrebări:
 - a) De ce este necesară apelarea funcției `timeSync()` pe microcontroler înainte de a transmite date către InfluxDB?
 - b) Care este diferența dintre o etichetă (*Tag*) și un câmp (*Field*) în InfluxDB? De ce valorile numerice continue se salvează ca *Fields*?
 - c) Ce avantaje oferă o bază de date de serii temporale (TSDB) față de o bază de date relațională clasică (MySQL / PostgreSQL) la monitorizarea utilajelor industriale?

Anexa 1

```
#if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>
#include "DHT.h"

// 1. Configurare retea Wi-Fi
#define WIFI_SSID "NUME_RETEA_WIFI"
#define WIFI_PASSWORD "PAROLA_RETEA_WIFI"

// 2. Configurare parametri InfluxDB Cloud
#define INFLUXDB_URL "https://eu-central-1-1.aws.cloud2.influxdata.com"
```

```

#define INFLUXDB_TOKEN "TOKEN_GENERAT_DE_INFLUXDB"
#define INFLUXDB_ORG "ORGANIZATIA_TA"
#define INFLUXDB_BUCKET "laborator_senzori"
// Configurare senzor DHT22 pe pinul D6 (GPIO12)
#define DHTPIN D6
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
float Temperature;
float Humidity;
// Fus orar pentru sincronizare marcaj temporal (EET - Europa de Est)
#define TZ_INFO "EET-2EEST,M3.5.0/3,M10.5.0/4"
// Initializare client InfluxDB cu certificat SSL preconfigurat
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN,
InfluxDbCloud2CACert);
// Declarare punct de date in tabelul "masuratori"
Point sensor("masuratori");
void setup() {
  Serial.begin(115200);
  pinMode(DHTPIN, INPUT);
  dht.begin();
  // Conectare la reseaua Wi-Fi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Conectare la Wi-Fi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);}
  Serial.println("\nWi-Fi conectat!");
  // Adaugare eticheta fixa (TAG) pentru identificarea dispozitivului
  sensor.addTag("device", DEVICE);
  sensor.addTag("locatie", "Laborator_IIoT");
  // Sincronizare ceas NTP (obligatorie pentru validare certificat SSL si timestamp)
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");
  // Verificare conexiune server InfluxDB
  if (client.validateConnection()) {
    Serial.print("Conectat cu succes la InfluxDB: ");
    Serial.println(client.getServerUrl());
  }
}

```

```

} else {
    Serial.print("Eroare conexiune InfluxDB: ");
    Serial.println(client.getLastErrorMessage());}
void loop() {
    // Citire valori de la senzorul DHT22
    Temperature = dht.readTemperature();
    Humidity = dht.readHumidity();
    // Verificare erori de citire senzor
    if (isnan(Temperature) || isnan(Humidity)) {
        Serial.println("Eroare la citirea senzorului DHT22!");
        delay(2000);
        return;}
    // Resetare campuri anterioare si adaugare valori curente (FIELDS)
    sensor.clearFields();
    sensor.addField("Temperatura", Temperature);
    sensor.addField("Umiditate", Humidity);
    sensor.addField("Semnal_WiFi", WiFi.RSSI());
    // Afisare pachet in format Line Protocol pe Serial Monitor
    Serial.print("Scriere punct de date: ");
    Serial.println(client.pointToLineProtocol(sensor));
    // Verificare reconectare Wi-Fi
    if (wifiMulti.run() != WL_CONNECTED) {
        Serial.println("Conexiune Wi-Fi pierduta!");}
    // Transmite catre InfluxDB Cloud
    if (!client.writePoint(sensor)) {
        Serial.print("Eroare la scrierea in InfluxDB: ");
        Serial.println(client.getLastErrorMessage());}
    // Interval de esantionare: 5 secunde
    delay(5000);}

```

Lucrare de laborator 4: Procesarea semnalelor industriale - filtrare în domeniul timp și analiză spectrală FFT pentru detecția anomaliilor

1. Introducere

În sistemele de informatică industrială și de monitorizare a stării echipamentelor (*Condition Monitoring*), semnalele achiziționate de la traductoare (distanță, vibrații, curenți, presiuni sau zgomot acustic) sunt în mod frecvent afectate de perturbații fizice, zgomot de fond și valori aberante (*outliers*). Un sistem de control sau diagnosticare care utilizează date neprelucrate și nefiltrate poate genera alarme false sau poate ignora defectele timpurii.

Această lucrare de laborator abordează cele două mari paradigme de prelucrare a semnalelor discutate în Capitolele 4 și 5 din partea teoretică:

1. **Prelucrarea în domeniul timp:** eliminarea valorilor aberante și a zgomotului de măsurare printr-un lanț de filtrare hibrid (*Filtru Median + Medie Mobilă*) aplicat pe un senzor ultrasonic de distanță, urmat de un detector statistic de schimbare a regimului (**EWMA** - *Exponentially Weighted Moving Average* / **CUSUM** - *Cumulative Sum*);
2. **Prelucrarea în domeniul frecvență:** achiziția semnalului acustic al unui motor prin intermediul unui modul microfon analogic, urmată de calculul **Transformatei Fourier Rapide (FFT)** în timp real pe calculator, pentru identificarea armonicilor anormale generate de frecarea sau blocajul mecanic.

2. Obiectivele lucrării

- Înțelegerea diferenței dintre zgomotul alb de măsurare și valorile aberante de tip impuls (*glitches / outliers*);
- Implementarea directă pe microcontroler (Arduino) a algoritmilor de filtrare în timp real: **Filtrul Median** și **Filtrul cu Medie Mobilă (*Moving Average*)**;
- Aplicarea metodei statistice **EWMA** pentru detecția salturilor anormale în semnal;
- Eșantionarea la o frecvență fixă a unui semnal analogic dinamic (microfon acustic) și transmiterea pachetelor de date prin conexiune serială USB;
- Dezvoltarea unui script Python pe PC pentru calculul spectrului de frecvență **FFT** și vizualizarea armonicilor în timp real;
- Monitorizarea energiei pe benzi spectrale de frecvență înaltă pentru detecția automată a anomaliilor mecanice la un motor.

3. Materiale și componente necesare

3.1. Componente Hardware

- Placă de dezvoltare **Arduino Uno** (cu cablu USB de conectare la PC);
- Senzor ultrasonic de distanță **HC-SR04**;
- Modul senzor de sunet cu microfon analogic (ex. **KY-037** sau **KY-038**);

- Motor electric DC, servomotor sau ventilator mic (utilizat ca sursă fizică de vibrație/sunet);
- 1x LED + 1x rezistență de 220 Ω (indicator optic de alarmă);
- 1x breadboard și cabluri de conexiune.

3.2. Resurse Software

- **Arduino IDE** (pentru scrierea, compilarea și încărcarea programelor pe microcontrolerul Arduino Uno);
- Interpretator și mediu de dezvoltare **Python**: se recomandă utilizarea uneia dintre următoarele distribuții portabile (care nu necesită instalare și rulează direct dintr-un folder dezarhivat sau de pe stick USB):
 - **WinPython** (arhivă .zip descărcată de pe site-ul oficial: <https://winpython.github.io/>) – conține Python gata configurat împreună cu pachetele științifice de bază; pentru deschiderea și rularea scripturilor se utilizează utilitarul **IDLE.exe** sau consola dedicată **WinPython Command Prompt.exe**;
 - **Thonny Portable** (descărcabil de pe site-ul oficial: <https://thonny.org/>) – oferă un mediu de dezvoltare grafic (GUI) complet, simplu și intuitiv, având interpretatorul Python integrat.

Biblioteci Python necesare: *pyserial*, *numpy*, *scipy*, *matplotlib*. Instalarea pachetelor suplimentare se realizează direct din WinPython Command Prompt (sau din meniul „Tools→Manage packages” în Thonny) prin comanda:

```
pip install pyserial numpy scipy matplotlib
```

4. Desfășurarea lucrării

PARTEA I: Achiziția, filtrarea în domeniul timp și detecția statistică pe senzorul ultrasonic

4.1. Conectarea hardware a senzorului ultrasonic HC-SR04

Senzorul ultrasonic HC-SR04 măsoară timpul de zbor al undelor acustice reflectate de un obstacol. Realizați conexiunile pe breadboard conform Tabelului 4.1:

Tabelul 4.1. Conexiunile senzorului HC-SR04 și ale LED-ului de alarmă la Arduino Uno

Componentă	Pin Componentă	Pin Arduino Uno	Rol funcțional
HC-SR04	VCC	5V	Alimentare senzor (+5 V)
	GND	GND	Masă comună (0 V)
	Trig	Pin Digital 9	Declanșare impuls ultrasonic (10 μ s)
	Echo	Pin Digital 10	Recepție durată impuls reflectat
LED Alarmă	Anod (+)	Pin Digital 13 (prin rezistență 220 Ω)	Semnalizare optică anomalie
	Catod (-)	GND	Închiderea circuitului

4.2. Implementarea filtrului median și a mediei mobile

Un senzor ultrasonic generează periodic valori eronate (valori nule 0 cm când unda se reflectă oblic sau valori maxime de 400 cm la pierderea ecoului).

- **Filtrul Median (fereastră $N = 5$):** sortează ultimele 5 eșantioane achiziționate și alege valoarea din mijloc. Acest filtru **elimină complet vârfulurile aberante (outliers)**, fără a deforma tranziția reală a semnalului.
- **Media Mobilă:** calculează media aritmetică a valorilor rezultate după filtrarea mediană, netezind zgomotul alb de măsurare.

- **Detecția statistică EWMA:** calculează media mobilă ponderată exponențial:

$$S_k = \alpha \cdot y_k + (1 - \alpha) \cdot S_{k-1}$$

Dacă abaterea absolută dintre valoarea filtrată curentă și media istorică depășește un prag critic ($|y_k - S_k| > \text{Prag}$), sistemul detectează o anomalie de mișcare bruscă și activează LED-ul de alarmă.

Încărcați pe placa Arduino programul din **Anexa 1**. Deschideți meniul **Tools** → **Serial Plotter** (Ctrl + Shift + L) la viteza de **115200 baud**. Mișcați mâna în fața senzorului și observați comparativ:

1. Linia albastră: semnalul brut cu perturbații;
2. Linia roșie: semnalul curățat prin filtrul median și media mobilă;
3. Linia verde: pragul de alarmă EWMA.

PARTEA A II-A: Analiza spectrală FFT în domeniul frecvență pe semnalul acustic al motorului

4.3. Conectarea hardware a modului de microfon analogic

Modulul cu microfon analogic (KY-037 / KY-038) transformă undele de presiune acustică într-un semnal electric de tensiune variabilă. Realizați conexiunile conform Tabelului 4.2:

Tabelul 4.2. Conexiunile modului de microfon la Arduino Uno

Pin Modul Microfon	Pin Arduino Uno	Rol funcțional
+ (VCC)	5V	Alimentare modul (+5 V)
G (GND)	GND	Masă comună (0 V)
A0 (Analog Out)	Pin Analogic A0	Semnal analogic de tensiune proporțional cu sunetul

Observație: Așezați microfonul la o distanță de 3-5 cm de corpul motorului sau ventilatorului din laborator.

4.4. Eșantionarea pe Arduino

Pentru a analiza frecvențe acustice de până la 500 Hz, conform teoremei de eșantionare Nyquist-Shannon ($f_s \geq 2 \cdot f_{\max}$), este necesară o frecvență de eșantionare de minimum 1.000 Hz ($T_s = 1$ ms).

Încărcați pe Arduino programul din **Anexa 2**. Acesta configurează convertorul analog-digital (ADC) pentru eșantionare la pas fix de 1 ms și transmite pachete de câte **128 de eșantioane brute** prin portul serial către calculator.

4.5. Calculul și vizualizarea FFT în timp real cu Python

Pe calculator, deschideți mediul Python (WinPython IDLE, Thonny) și rulați scriptul din **Anexa 3**.

Scriptul execută următorii pași algoritmici:

1. Citește pachetul de 128 de eșantioane de la portul serial (ex. COM1 pe Windows sau /dev/ttyUSB0 pe Linux);
2. Îndepărtează componenta continuă (*DC Offset*) prin scăderea mediei semnalului;
3. Aplică o fereastră de ponderare **Hanning** pentru reducerea dispersiei spectrale (*spectral leakage*);
4. Calculează **Transformata Fourier Rapidă (FFT)** cu funcția `numpy.fft.rfft()`;
5. Calculează spectrul de amplitudine și energia pe banda de frecvențe înalte (250–500 Hz);
6. Afișează grafic semnalul în domeniul timp (sus) și spectrul de frecvență (jos) în timp real.

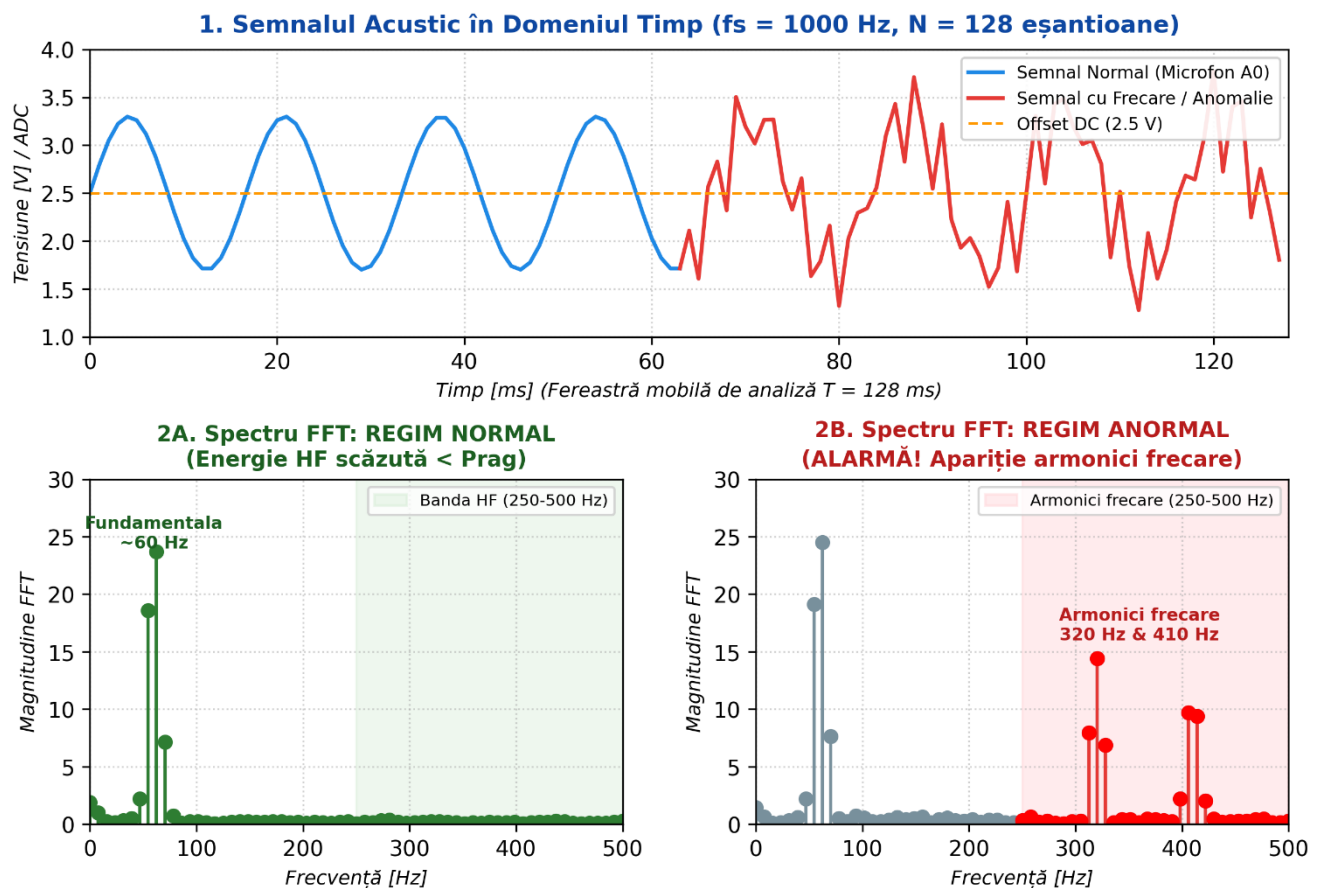


Figura 4.1 Semnalul acustic în domeniul timp și spectrul FFT corespunzător în regim normal vs. regim de frecare mecanică

4.6. Testarea experimentală a anomaliilor mecanice

1. **Regim Normal:** Porniți motorul să se rotească liber. Observați pe spectrul FFT că energia este concentrată la frecvența joasă de rotație a axului (40–80 Hz);
2. **Regim Anormal (Frecare / Forțare):** Aplicați o forță ușoară de frecare mecanică pe axul motorului (folosind o bucățică de carton sau un creion). Observați cum pe graficul FFT apar instantaneu **armonici de frecvență înaltă** (300–450 Hz), iar indicatorul de energie spectrală depășește pragul de avarie, generând alerta de defect în consolă.

5. Exerciții și teme practice

- 5.1. Optimizarea mărimii ferestrei la filtrul median: Modificați dimensiunea ferestrei filtrului median din codul Arduino de la $N = 5$ la $N = 3$ și apoi la $N = 9$. Analizați pe *Serial Plotter* compromisul dintre capacitatea de eliminare a perturbațiilor mari și întârzierea introdusă de filtru (*latența de răspuns*).
- 5.2. Ajustarea pragului de sensibilitate EWMA: Modificați factorul de ponderare α (0.1 vs. 0.5) în algoritmul EWMA pentru senzorul ultrasonic. Determinați experimental valoarea optimă a parametrului α care detectează mișcarea rapidă a mâinii, dar ignoră trepidațiile lente ale aerului.
- 5.3. Monitorizarea energiei pe benzi spectrale specifice în Python: Modificați scriptul Python din Anexa 3 pentru a calcula separat energia în două benzi de frecvență:
 - Banda 1 (Frecvențe joase: 20–100 Hz): corespunde dezechilibrului maselor rotative;
 - Banda 2 (Frecvențe înalte: 250–500 Hz): corespunde frecărilor uscate și șocurilor din lagăre.
Afișați pe ecran un mesaj specific de diagnosticare în funcție de banda în care crește energia.
- 5.4. Fuziune senzorială de bază: Scrieți o condiție logică pe Arduino care să declanșeze alarma critică de avarie doar dacă atât distanța măsurată de senzorul ultrasonic scade sub 10 cm, cât și nivelul de zgomot acustic depășește pragul mediu de lucru.

6. Întrebări de verificare

- 6.1. De ce un filtru bazat pe media aritmetică simplă eșuează în eliminarea unui vârf aberant izolat (*outlier* de 0 cm), în timp ce filtrul median îl elimină complet?
- 6.2. Ce condiție impune Teorema Nyquist-Shannon asupra frecvenței de eșantionare f_s și ce fenomen nedorit (*aliasing*) apare dacă eșantionăm un semnal de 400 Hz cu o frecvență $f_s = 500$ Hz?
- 6.3. De ce este necesară scăderea componentei continue (*DC Offset*) înainte de aplicarea Transformatei Fourier (FFT)?
- 6.4. Care este avantajul utilizării analizei spectrale FFT față de măsurarea simplă a amplitudinii de vârf a semnalului în diagnoza utilajelor industriale?

Anexa 1: Cod Arduino C++ pentru Partea I (Filtrare Mediană, Medie Mobilă și EWMA)

```
// Pini conectare HC-SR04 si LED
const int TRIG_PIN = 9;
const int ECHO_PIN = 10;
const int LED_PIN = 13;
// Parametri filtrare
const int WINDOW_SIZE = 5;
float rawBuffer[WINDOW_SIZE];
int bufferIndex = 0;
// Parametri EWMA
float ewmaAverage = 0.0;
const float ALPHA = 0.2; // Factor de netezire (0.0 - 1.0)
const float THRESHOLD = 15.0; // Prag de abatere pentru detectie anomalie (cm)
// Functie auxiliara pentru sortare (necesara la filtrul median)
float computeMedian(float arr[], int size) {
    float temp[size];
    for (int i = 0; i < size; i++) temp[i] = arr[i];
    // Sortare simpla prin selectie
    for (int i = 0; i < size - 1; i++) {
        for (int j = i + 1; j < size; j++) {
            if (temp[i] > temp[j]) {
                float aux = temp[i];
                temp[i] = temp[j];
                temp[j] = aux;}}}
    return temp[size / 2]; // Returneaza elementul median}
void setup() {
    Serial.begin(115200);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(LED_PIN, OUTPUT);
    // Initializare buffer cu o valoare initiala valida
    for (int i = 0; i < WINDOW_SIZE; i++) rawBuffer[i] = 30.0;
    ewmaAverage = 30.0;}
void loop() {
    // 1. Masurare distanta bruta (durata ecou in microsecunde)
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
```

```

digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
long duration = pulseIn(ECHO_PIN, HIGH, 30000); // Timeout 30ms
float rawDistance = (duration == 0) ? 400.0 : (duration * 0.0343) / 2.0;
// 2. Salvare in buffer circular
rawBuffer[bufferIndex] = rawDistance;
bufferIndex = (bufferIndex + 1) % WINDOW_SIZE;
// 3. Aplicare Filtru Median
float medianDistance = computeMedian(rawBuffer, WINDOW_SIZE);
// 4. Aplicare detector statistic EWMA
ewmaAverage = ALPHA * medianDistance + (1.0 - ALPHA) * ewmaAverage;
float deviation = abs(medianDistance - ewmaAverage);
// 5. Verificare prag anomalie
bool anomaly = (deviation > THRESHOLD);
digitalWrite(LED_PIN, anomaly ? HIGH : LOW);
// 6. Transmite date catre Serial Plotter (format etichetat)
Serial.print("Brut:");
Serial.print(rawDistance);
Serial.print(" ");
Serial.print("Filtrat_Median:");
Serial.print(medianDistance);
Serial.print(" ");
Serial.print("EWMA_Ref:");
Serial.print(ewmaAverage);
Serial.print(" ");
Serial.print("Alarma_Anomalie:");
Serial.println(anomaly ? 50.0 : 0.0);
delay(50); // Perioada de esantionare Ts = 50 ms (20 Hz)}

```

Anexa 2: Cod Arduino C++ pentru Partea a II-a (Eșantionare rapidă microfon port A0)

```

const int MIC_PIN = A0;
const int SAMPLES = 128;
const unsigned long SAMPLING_PERIOD_US = 1000; // 1 ms -> Frecventa fs = 1000 Hz
int signalBuffer[SAMPLES];
void setup() {
    Serial.begin(115200);

```

```

pinMode(MIC_PIN, INPUT);}
void loop() {
    unsigned long nextSampleTime = micros();
    // 1. Achizitie sincronizata a 128 de esantioane
    for (int i = 0; i < SAMPLES; i++) {
        while (micros() < nextSampleTime) {
            // Asteptare activa pentru respectarea pasului strict de 1 ms}
            signalBuffer[i] = analogRead(MIC_PIN);
            nextSampleTime += SAMPLING_PERIOD_US;}
    // 2. Transmitere pachet pe Serial sub forma de linie separata prin virgula
    Serial.print("START,");
    for (int i = 0; i < SAMPLES; i++) {
        Serial.print(signalBuffer[i]);
        if (i < SAMPLES - 1) Serial.print(",");}
    Serial.println("END");
    delay(20); // Pauza intre cadre de analiza}

```

Anexa 3: Script Python pe PC pentru Calculul și Afișarea Spectrului FFT

```

import serial
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import rfft, rfftfreq
# 1. Configurare port serial (ajustati portul conform sistemului vostru: 'COM1', 'COM2' etc.)
SERIAL_PORT = 'COM1'
BAUD_RATE = 115200
SAMPLES = 128
FS = 1000.0 # Frecventa de esantionare (Hz)
try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=2)
    print(f"Conectat cu succes la {SERIAL_PORT}")
except Exception as e:
    print(f"Eroare la deschiderea portului serial: {e}")
    exit()
# Configurare grafice Matplotlib interactive
plt.ion()
fig, (ax_time, ax_freq) = plt.subplots(2, 1, figsize=(10, 6))
time_axis = np.linspace(0, (SAMPLES - 1) / FS * 1000, SAMPLES) # Axa timpului in ms

```

```

freq_axis = rfftfreq(SAMPLES, 1.0 / FS) # Axa frecventelor (0 - 500 Hz)
line_time, = ax_time.plot(time_axis, np.zeros(SAMPLES), 'b-', label='Semnal Microfon (A0)')
ax_time.set_ylim(0, 1023)
ax_time.set_xlabel('Timp (ms)')
ax_time.set_ylabel('Amplitudine ADC')
ax_time.set_title('Semnal Acustic în Domeniul Timp')
ax_time.grid(True)
line_freq, = ax_freq.plot(freq_axis, np.zeros(len(freq_axis)), 'r-', label='Spectru FFT')
ax_freq.set_ylim(0, 200)
ax_freq.set_xlabel('Frecvență (Hz)')
ax_freq.set_ylabel('Magnitudine')
ax_freq.set_title('Spectrul de Frecvență (FFT) - Detecție Armonici')
ax_freq.grid(True)
plt.tight_layout()

print("Pornire procesare semnal... Aplicati frecare pe axul motorului pentru a vedea anomalia.")
while True:
    try:
        raw_line = ser.readline().decode('utf-8', errors='ignore').strip()
        if raw_line.startswith("START,") and raw_line.endswith(",END"):
            parts = raw_line.split(",")
            values = [float(p) for p in parts[1:-1]]
            if len(values) == SAMPLES:
                signal_data = np.array(values)
                # 1. Eliminare componenta continua (DC Offset)
                signal_zero_mean = signal_data - np.mean(signal_data)
                # 2. Aplicare fereastră Hanning
                windowed_signal = signal_zero_mean * np.hanning(SAMPLES)
                # 3. Calcul Transformata Fourier Rapida (FFT)
                fft_spectrum = np.abs(rfft(windowed_signal))
                # 4. Calcul energie pe banda inalta (250 Hz - 500 Hz)
                high_freq_indices = np.where((freq_axis >= 250) & (freq_axis <= 500))[0]
                high_freq_energy = np.sum(fft_spectrum[high_freq_indices])
                # 5. Actualizare grafice
                line_time.set_ydata(signal_data)
                line_freq.set_ydata(fft_spectrum)
                ax_freq.set_ylim(0, max(np.max(fft_spectrum) * 1.2, 50))
                # Verificare prag anomalie spectrala

```

```

    if high_freq_energy > 150.0:
        ax_freq.set_title(f"Spectrul FFT - ALARMĂ! Frecare Mecanică Detectată (Energie HF: {high_freq_energy:.1f})", color='red')
    else:
        ax_freq.set_title(f"Spectrul FFT - Regim Normal de Funcționare (Energie HF: {high_freq_energy:.1f})",
color='green')
    plt.pause(0.01)
except KeyboardInterrupt:
    print("Oprire program de catre utilizator.")
    ser.close()
    break
except Exception as e:
    continue

```

Lucrare de laborator 5: Integrare industrială și comunicații seriale cu PLC FX3U și Interfață HMI

1. Introducere

În sistemele industriale, controlerele logice programabile (PLC) execută buclele de reglare rapidă, comenzile secvențiale și interblocările fizice de securitate la nivelul mașinii (Nivelul 1 OT). Pentru a transmite datele de producție către operatori și sistemele de dispecerizare, este necesară realizarea unei punți de comunicație serială prin intermediul unui calculator industrial (IPC), al PC-ului de proces sau al unui nod edge dedicat (Raspberry Pi 5).

Această lucrare de laborator tratează integrarea unui controler fizic real **Mitsubishi FX3U**, prezentând două modalități practice de monitorizare și comandă:

- **Varianta A (Conectare directă la PC și HMI nativ în Python):** utilizează cablul serial nativ (Mini-DIN 8 pini la mufă DB9 RS-232 conectată la portul COM1/COM2 pe PC). Interfața grafică HMI este rulată direct din **WinPython / Thonny IDE** folosind biblioteca standard *tkinter*;
- **Varianta B (Conectare la Raspberry Pi 5 și Dashboard Web în Node-RED):** utilizează un adaptor USB-Serial conectat la portul `/dev/ttyUSB0` al plăcii Raspberry Pi 5, unde rulează serverul Node-RED ce expune tabloul de bord în rețeaua locală a fabricii.

2. Obiectivele lucrării

- Însușirea arhitecturii hardware a gamei Mitsubishi FX3U (intrări digitale X, ieșiri pe releu Y, relee auxiliare de memorie M, registre numerice de date D);
- Dezvoltarea, compilarea și încărcarea programului în limbaj Ladder folosind mediul **GX Works2 / Developer**;
- Realizarea conexiunii fizice conform uneia dintre cele două variante disponibile pe bancul de lucru (Varianta A pe PC sau Varianta B pe Raspberry Pi 5);
- Instalarea și configurarea mediului Node-RED pe Raspberry Pi 5 folosind scriptul oficial;
- Configurarea parametrilor de transmisie asincronă pe portul serial (9600 baud, paritate pară, 7 biți de date, 1 bit de stop);
- Dezvoltarea unui tablou de bord (**HMI**) pentru monitorizarea stării utilajului, contorizarea pieselor și transmiterea comenzilor de pornire/oprire/resetare;
- Verificarea principiului de autonomie locală a PLC-ului: programul de control din teren continuă să ruleze neafectat chiar dacă legătura serială cu calculatorul este deconectată.

3. Materiale și componente necesare

3.1. Componente Hardware

- Controler logic programabil **Mitsubishi FX3U** (alimentat la tensiunea nominală specificată pe carcasă, 24V DC sau 230V AC);

- Cablu de comunicație serială FX (mufă rotundă Mini-DIN 8 pini la mufă DB9 mamă RS-232);
- **Pentru Varianta A:** PC de laborator echipat cu port serial nativ RS-232 (conector tată DB9 pe panoul din spate);
- **Pentru Varianta B:** Placă de calcul **Raspberry Pi 5 8GB** + adaptor de conversie Serial RS-232 la USB;
- 2x Butoane fizice normal deschise (NO) conectate la intrările X0 (Start local) și X1 (Stop local);

3.2. Resurse Software

- **GX Works2** (sau **GX Developer**) – mediul oficial de programare și monitorizare online PLC;
- **Pentru Varianta A (PC):** Interpretator Python 3 portabil (**WinPython** dezarhivat sau **Thonny Portable**) cu pachetul *pyserial* instalat local;
- **Pentru Varianta B (Raspberry Pi 5):** **Node-RED** instalat cu modulul de interfață grafică *node-red-dashboard*.

4. Desfășurarea lucrării

4.1. Conexiunile hardware de proces

Conectați butonul de *Start* la borna X0, butonul de *Stop* la borna X1, iar polul comun al butoanelor la borna COM a PLC-ului. Lampa de semnalizare a motorului va fi ieșirea Y0 și lampa de avarie va fi Y1. Conectați mufa rotundă Mini-DIN cu 8 pini în portul frontal de programare al PLC-ului FX3U.

Arhitectura completă de integrare este ilustrată în Figura 5.1.

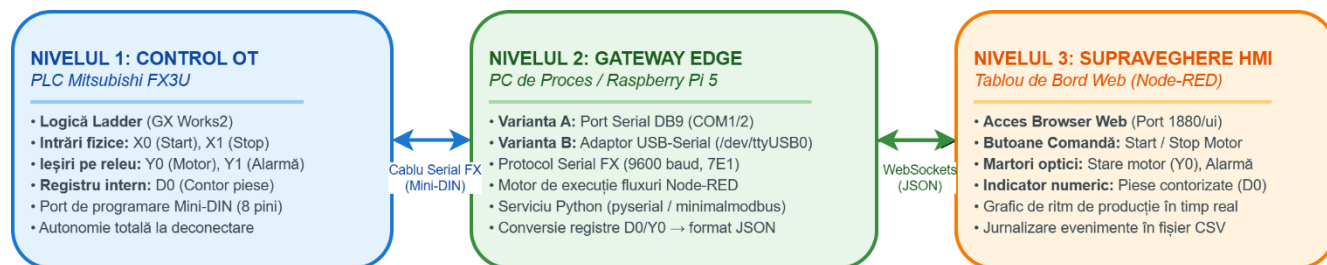


Figura 5.1 Arhitectura de integrare a PLC-ului FX3U cu gateway-ul de comunicație și interfața HMI

4.2. Selectarea variantei de conectare la calculator

Varianta A: Conectare directă la portul serial DB9 al PC-ului

1. Conectați mufa DB9 a cablului serial direct în portul serial al calculatorului (COM1 sau COM2);
2. În mediul GX Works2, selectați *Connection Destination* → *Connection1* → *Serial USB* → *COM1*, setați rata la **9600 bps**, paritate **Even**, **7 biți de date**, **1 bit de stop** și apăsați butonul *Connection Test*;
3. După confirmarea conexiunii, calculatorul de laborator va rula aplicația grafică HMI direct prin scriptul Python din **Anexa 2**.

Varianta B: Conectare la Raspberry Pi 5 și configurare Node-RED

1. Conectați mufa DB9 a cablului serial în adaptorul USB-Serial și introduceți conectorul USB într-un port al plăcii Raspberry Pi 5;
2. Pe Raspberry Pi, verificați recunoașterea portului în terminal prin comanda:

```
ls -l /dev/ttyUSB*
```

(Portul alocat va fi `/dev/ttyUSB0` sau `/dev/ttyUSB1`)

3. Acordați permisiunile de acces:

```
sudo chmod 666 /dev/ttyUSB0
```

Instalarea Node-RED pe Raspberry Pi 5: Instalarea se realizează folosind scriptul oficial recomandat de Fundația Node-RED (documentația oficială completă este disponibilă la adresa: <https://nodered.org/docs/getting-started/raspberrypi>):

1. Deschideți un terminal pe Raspberry Pi și rulați comanda de instalare:

```
bash < (curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

2. Răspundeți cu y la întrebările privind instalarea pachetelor recomandate pentru Raspberry Pi;
3. Porniți serviciul Node-RED prin comanda:

```
node-red-start
```

4. (Opțional) Pentru ca Node-RED să pornească automat la fiecare pornire a plăcii Raspberry Pi, rulați comanda:

```
sudo systemctl enable nodered.service
```

5. Deschideți editorul Node-RED în browser la adresa <http://localhost:1880> (sau http://IP_RASPBERRY_PI:1880);
6. Instalați pachetul de tablouri de bord din meniul *Manage Palette* → *Install* → *node-red-dashboard*.

4.3. Implementarea și încărcarea logicii Ladder (GX Works2)

1. Deschideți mediul **GX Works2**, creați un proiect nou (*Series: FXCPU, Type: FX3U, Language: Ladder*);
2. Introduceți următoarea diagramă de contacte:

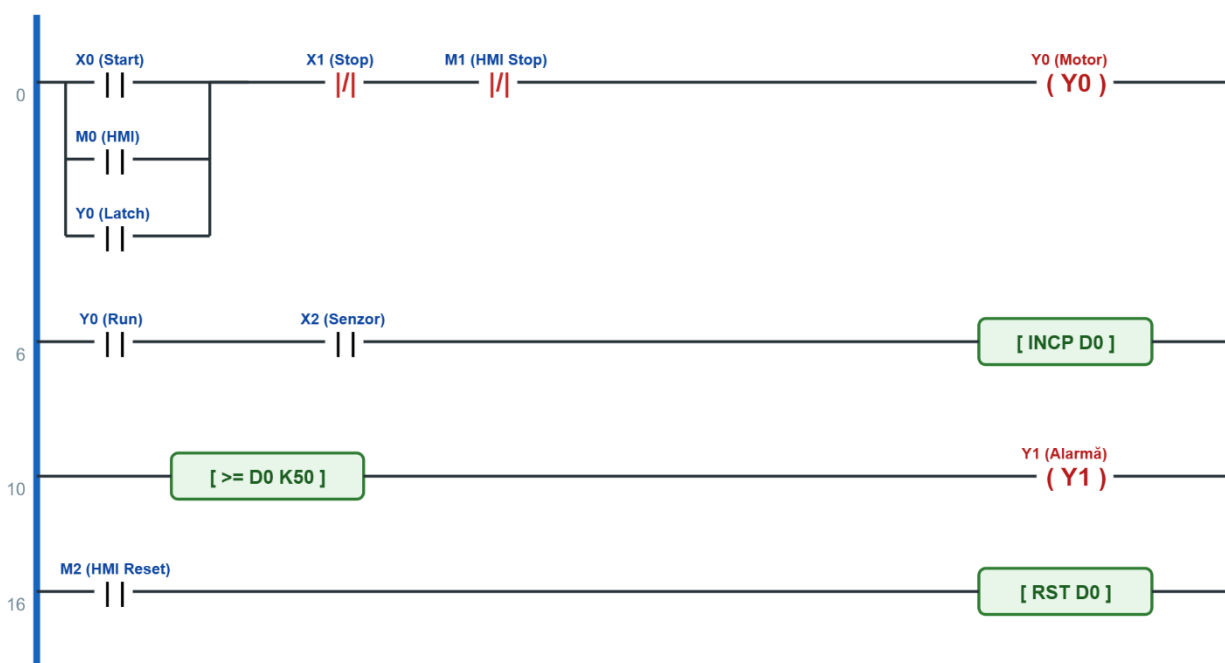


Figura 5.2 Diagrama LD

3. Compilați programul prin tasta **F4** (*Rebuild All*);
4. Descărcați programul în memoria PLC-ului selectând **Online** → **Write to PLC**;
5. Comutați cheia fizică a PLC-ului pe **RUN**. Testați funcționarea secvenței apăsând butoanele fizice X0 și X1.

4.4. Rularea interfeței HMI pe calculator (Varianta A – Python Tkinter)

1. Deschideți mediul **IDLE** din folder-ul portabil WinPython sau Thonny IDE;
2. Creați un fișier nou *hmi_plc_fx3u.py* și introduceți codul complet din **Anexa 2**;
3. Rulați scriptul apăsând tasta **F5**;
4. Pe ecran se va deschide o fereastră grafică HMI ce conține:
 - Un martor luminos (LED virtual) ce indică starea motorului Y0 (Verde = Pornit, Gri = Oprit);
 - Un afișaj numeric mare ce arată în timp real numărul de piese contorizate în registrul D0;
 - Trei butoane grafice: [**START (M0)**], [**STOP (M1)**] și [**RESET (M2)**].

4.5. Testarea experimentală a sistemului integrat

1. **Comandă din interfața grafică:** Apăsăți butonul *START* din aplicația Python (sau Node-RED). Verificați aprinderea simultană a indicatorului de pe ecran și a ieșirii Y0;
2. **Prioritatea comenzilor de siguranță locale:** În timp ce motorul este pornit din HMI, apăsați butonul fizic de oprire X1 de pe banc. Constatați oprirea imediată a mașinii, confirmând regula inginerească conform căreia interblocările hardware locale au prioritate asupra comenzilor venite prin software;

3. **Simularea producției:** Generați impulsuri pe intrarea X2 (prin atingerea bornelor sau apăsarea unui microîntrerupător). Urmăriți incrementarea în timp real a contorului D0 pe ecranul calculatorului;
4. **Verificarea decuplării comunicației:** Deconectați cablul serial dintre PLC și calculator. Observați că procesul fizic își continuă ciclul automat, demonstrând reziliența și autonomia controlerului de câmp.

5. Exerciții și teme practice

- 5.1. Adăugarea unui prag dinamic de avertizare: Modificați programul Ladder astfel încât valoarea maximă a lotului de piese să fie citită dintr-un registru intern D10. În aplicația Python HMI, adăugați un câmp de introducere numerică (*Entry*) și un buton de setare prin care operatorul poate modifica mărimea lotului direct de pe ecran.
- 5.2. Contorizarea orelor de funcționare a motorului: Implementați în PLC un cronometru (*Timer* pe 100 ms, ex. T0) care măsoară timpul total în care ieșirea Y0 este activă și stochează minutele cumulate în registrul D2. Afișați timpul de funcționare în interfața HMI.
- 5.3. Calculul ritmului de fabricație în Python: Extindeți scriptul Python pentru a calcula numărul de piese produse în ultimele 60 de secunde și afișați viteza instantanee a liniei exprimată în piese/minut.
- 5.4. Jurnalizarea automată a evenimentelor în format CSV: Scrieți o secvență în Python care adaugă o linie nouă într-un fișier raport_schimb.csv la fiecare pornire, oprire sau atingere a pragului de 50 de piese, înregistrând data, ora și starea procesului.

6. Întrebări de verificare

- 6.1. Care este diferența dintre o intrare fizică X, o ieșire fizică Y și un releu intern de memorie M în arhitectura PLC-urilor Mitsubishi FX?
- 6.2. De ce comenzile de pornire/oprire transmise dintr-o interfață HMI se mapează pe relele interne M și nu pot forța direct intrările fizice X?
- 6.3. Ce parametri de comunicație serială trebuie să fie perfect identici între PLC și calculator pentru ca pachetele de date să fie recepționate fără erori de paritate sau de cadru (*framing errors*)?
- 6.4. De ce este periculos ca butonul de oprire de urgență al unei mașini să fie implementat exclusiv pe un ecran tactil HMI conectat prin rețea, în loc să fie un buton fizic cablat pe circuitul de comandă?

Anexa 1: Lista instrucțiunilor Ladder (Format Text Mnemonic GX Works2 / Developer)

```

0 LD   X000
1 OR   M0
2 OR   Y000
3 ANI  X001
4 ANI  M1
5 OUT  Y000
6 LD   Y000

```

```

7 AND X002
8 INCP D0
10 LD>= D0 K50
15 OUT Y001
16 LD M2
17 RST D0
19 END

```

Anexa 2: Script Python complet pentru aplicația HMI pe PC (WinPython / Thonny)

```

import serial
import time
import sys
import tkinter as tk
from tkinter import messagebox

# 1. Configurare Port Serial
# Pe Windows: 'COM1', 'COM2' (sau 'COM3'/'COM4' daca e adaptor USB)
# Pe Raspberry Pi: '/dev/ttyUSB0'
SERIAL_PORT = 'COM1' if sys.platform.startswith('win') else '/dev/ttyUSB0'
BAUD_RATE = 9600

try:
    ser = serial.Serial(
        port=SERIAL_PORT,
        baudrate=BAUD_RATE,
        bytesize=serial.SEVENBITS,
        parity=serial.PARITY_EVEN,
        stopbits=serial.STOPBITS_ONE,
        timeout=0.5
    )
    print(f"Conexiune seriala stabilita cu succes pe {SERIAL_PORT}")
except Exception as e:
    print(f"Eroare port serial: {e}")
    ser = None

# 2. Functii Protocol Serial Mitsubishi FX

```

```

def read_d_register(d_num):
    """Citeste valoarea pe 16 biti din registrul D (D0 = adresa 0x1000)"""
    if not ser: return 0
    addr = 0x1000 + (d_num * 2)
    cmd = f"\x020{addr:04X}02\x03"
    chk = sum(ord(c) for c in cmd[1:]) % 256
    try:
        ser.write(f"{cmd}{chk:02X}".encode('ascii'))
        time.sleep(0.04)
        resp = ser.read(11)
        if len(resp) >= 7 and resp[0] == 0x02:
            hex_val = resp[1:5].decode('ascii')
            low = int(hex_val[0:2], 16)
            high = int(hex_val[2:4], 16)
            return (high << 8) | low
    except:
        pass
    return 0

def force_m_relay(m_num, state):
    """Fortezza starea releului intern M (M0=0x0800, M1=0x0801, M2=0x0802)"""
    if not ser: return
    cmd_code = '7' if state else '8' # '7' = Force ON, '8' = Force OFF
    addr = 0x0800 + m_num
    cmd = f"\x02{cmd_code}{addr:04X}\x03"
    chk = sum(ord(c) for c in cmd[1:]) % 256
    try:
        ser.write(f"{cmd}{chk:02X}".encode('ascii'))
        time.sleep(0.04)
        ser.read(2) # Raspuns ACK
    except:
        pass

# 3. Creare Interfata Grafica HMI (Tkinter)
root = tk.Tk()
root.title("HMI Industrial - PLC Mitsubishi FX3U")
root.geometry("460x430")
root.configure(bg="#eceff1")

```

```

# Titlu HMI
lbl_title = tk.Label(root, text="SUPRAVEGHERE & COMANDĂ PLC", font=("Helvetica",
13, "bold"), bg="#1565c0", fg="white", pady=10)
lbl_title.pack(fill="x")

# Panou Stare Motor & LED Indicator
frame_motor = tk.LabelFrame(root, text=" Stare Motor (Ieșire Y0) ",
font=("Helvetica", 10, "bold"), bg="#eceff1", padx=15, pady=10)
frame_motor.pack(pady=12, fill="x", padx=20)

canvas_led = tk.Canvas(frame_motor, width=35, height=35, bg="#eceff1",
highlightsthickness=0)
canvas_led.pack(side="left", padx=10)
led_circle = canvas_led.create_oval(4, 4, 31, 31, fill="#b0bec5",
outline="#37474f", width=2)

lbl_status = tk.Label(frame_motor, text="MOTOR OPRIT", font=("Helvetica", 11,
"bold"), fg="#c62828", bg="#eceff1")
lbl_status.pack(side="left", padx=10)

# Panou Contor Piese D0
frame_count = tk.LabelFrame(root, text=" Producție / Contor Piese (Registru D0) ",
font=("Helvetica", 10, "bold"), bg="#eceff1", padx=15, pady=10)
frame_count.pack(pady=5, fill="x", padx=20)

lbl_piese = tk.Label(frame_count, text="0 piese", font=("Consolas", 24, "bold"),
fg="#0d47a1", bg="#eceff1")
lbl_piese.pack()

# Functii butoane HMI
def cmd_start():
    force_m_relay(0, True) # Impuls Start pe M0
    time.sleep(0.1)
    force_m_relay(0, False)

def cmd_stop():
    force_m_relay(1, True) # Impuls Stop pe M1
    time.sleep(0.1)
    force_m_relay(1, False)

```

```

def cmd_reset():
    force_m_relay(2, True) # Impuls Reset pe M2
    time.sleep(0.1)
    force_m_relay(2, False)

# Panou Butoane de Comanda
frame_btns = tk.Frame(root, bg="#eceff1")
frame_btns.pack(pady=15)

btn_start = tk.Button(frame_btns, text="▶ START (M0)", font=("Helvetica", 9,
"bold"), bg="#2e7d32", fg="white", width=12, height=2, command=cmd_start)
btn_start.grid(row=0, column=0, padx=5)

btn_stop = tk.Button(frame_btns, text="■ STOP (M1)", font=("Helvetica", 9,
"bold"), bg="#c62828", fg="white", width=12, height=2, command=cmd_stop)
btn_stop.grid(row=0, column=1, padx=5)

btn_reset = tk.Button(frame_btns, text="↺ RESET (M2)", font=("Helvetica", 9,
"bold"), bg="#f57c00", fg="white", width=12, height=2, command=cmd_reset)
btn_reset.grid(row=0, column=2, padx=5)

# 4. Bucla de Actualizare Date (la fiecare 400 ms)
def update_hmi():
    piese_d0 = read_d_register(0)
    lbl_piese.config(text=f"{piese_d0} piese")

    if piese_d0 >= 50:
        lbl_status.config(text="LOT FINALIZAT (Y1)", fg="#f57c00")
        canvas_led.itemconfig(led_circle, fill="#ff9800")

    root.after(400, update_hmi)

root.after(400, update_hmi)
root.mainloop()

if ser: ser.close()

```

Lucrare de laborator 6: Inspecție optică industrială (AOI), control dimensional și detecție de defecte în timp real

1. Introducere

Inspecția optică automată (*Automated Optical Inspection* – AOI) reprezintă o tehnologie cheie în liniile industriale moderne pentru garantarea calității de 100% a pieselor fabricate, eliminând erorile și oboseala asociate controlului vizual manual. Așa cum s-a analizat în Capitolul 6 și Studiul de caz 2, o stație AOI industrială transformă fluxul video de la senzorul de imagine într-o decizie binară de conformitate (*Conform / Rebut*), calculează cotele metrologice ale piesei și acționează un mecanism fizic de sortare (cilindru pneumatic sau servomotor) într-un timp strict delimitat de ciclul liniei (*takt time*).

Această lucrare de laborator utilizează cel mai modern hardware de procesare edge:

- **Senzor optic de înaltă rezoluție:** camera **Raspberry Pi Camera Module 3** sau varianta **NoIR** (sensibilă în infraroșu pentru inspecții speciale);
- **Unitate de procesare edge:** placa **Raspberry Pi 5 8GB**, cu suport opțional pentru acceleratorul neuronal **Raspberry Pi AI Kit** (Hailo-8L NPU);
- **Conductă completă de procesare de imagine în Python cu OpenCV:** calibrare metrică (conversie pixeli în milimetri), filtrare spațială, binarizare adaptivă (prag Otsu), extragere de contururi, măsurători dimensionale, detecția defectelor de formă și comanda directă a semnalelor de sortare pe pinii GPIO.

2. Obiectivele lucrării

- Înțelegerea lanțului funcțional complet al unei stații de inspecție optică: iluminare → achiziție → preprocesare → metrologie → decizie → acționare fizică;
- Montarea și configurarea camerei Raspberry Pi Camera Module 3 prin stiva software *libcamera / Picamera2*;
- Implementarea procedurii de calibrare geometrică a camerei pentru determinarea factorului de scară (raportul pixeli/mm pe baza unei ținte etalon);
- Aplicarea tehnicilor de procesare de imagine: conversie la scară de gri, filtrare Gaussiană de netezire, binarizare automată Otsu și detecție de contururi (*cv2.findContours*);
- Măsurarea automată în timp real a dimensiunilor piesei (diametru exterior, diametru gaură interioară, arie, circularitate) și compararea cu toleranțele tehnologice;
- Comanda ieșirilor digitale GPIO pentru semnalizarea stării (*LED Verde = OK, LED Roșu / Releu = Rebut*) și salvarea automată a imaginilor cu defecte pentru trasabilitate conform normelor de calitate.

3. Materiale și componente necesare

3.1. Componente Hardware

- Placă de dezvoltare **Raspberry Pi 5 8GB** (cu sursă de alimentare USB-C de 27W);
- Modul cameră **Raspberry Pi Camera Module 3** (sau Camera Module 3 NoIR), conectat prin cablu panglică FPC la portul CAM/DISP 0 sau 1;
- Modul accelerator neuronal **Raspberry Pi AI Kit** (opțional, montat pe slotul PCIe M.2 HAT);
- 2x LED (1x Verde, 1x Roșu) + 2x rezistențe de 220 Ω (sau un modul cu releu de 5V pentru comanda cilindrului de sortare);
- Suport mecanic sau trepied pentru fixarea verticală a camerei deasupra bancului de lucru la o înălțime de 15–20 cm;
- Sursă de iluminare uniformă (lampă LED circulară sau lumină albă difuză);
- Piese de test industriale pe banc (șaibe metalice, piulițe, garnituri, capace sau șabloane din carton decupate cu diferite diametre și defecte de tip bavură sau fisură);
- Țintă etalon de calibrare (un pătrat alb sau negru cu dimensiunea exactă de 20 × 20 mm imprimat pe hârtie).

3.2. Resurse Software

- **Raspberry Pi OS** cu interfață grafică Desktop;
- Interpretator **Python** cu bibliotecile: *opencv-python* (cv2), *numpy*, *picamera2* (sau *cv2.VideoCapture*), RPi.GPIO (sau *gpiozero*). Instalarea bibliotecilor pe Raspberry Pi se realizează prin comanda:

```
sudo apt update  
sudo apt install -y python3-opencv python3-numpy python3-picamera2 python3-gpiozero
```

4. Desfășurarea lucrării

4.1. Arhitectura stației AOI și schema de conectare hardware

Arhitectura funcțională a stației este prezentată în Figura 6.1.



Figura 6.1 Lanțul de procesare a imaginilor și arhitectura stației AOI cu Raspberry Pi 5

Realizați conexiunile electrice pe pinii GPIO ai plăcii Raspberry Pi 5 conform Tabelului 6.1:

Tabelul 6.1. Conexiunile componentelor de semnalizare și sortare la pinii GPIO

Componentă	Pin Componentă	Pin Fizic Raspberry Pi 5	Nume Pin GPIO	Rol funcțional
LED Verde (Piesă OK)	Anod (+) prin 220 Ω	Pin 11	GPIO 17	Semnalizare optică piesă conformă
LED Roșu (Rebut)	Anod (+) prin 220 Ω	Pin 13	GPIO 27	Semnalizare piesă neconformă
Modul Releu Sortare	Pin Semnal (IN)	Pin 15	GPIO 22	Comandă cilindru pneumatic de deviere
Masa comună (GND)	Catozi (-) LED / GND Releu	Pin 6, 9 sau 14	GND	Închiderea circuitului de masă (0 V)

4.2. Testarea camerei și fixarea geometriei de inspecție

1. Fixați camera pe suportul vertical, orientată perpendicular în jos spre suprafața bancului de lucru;
2. Poziționați sub cameră o suprafață de fundal contrastant (de exemplu, o coală albă mată dacă piesele sunt metalice închise la culoare);
3. Deschideți un terminal pe Raspberry Pi și testați funcționarea camerei prin comanda:

```
rpicam-hello -t 5000
```

4. Asigurați-vă că piesa se află complet în centrul cadrului și că iluminarea este uniformă, fără reflexii orbitoare sau umbre puternice pe margini.

4.3. Calibrarea spațială: determinarea factorului de scară (Pixels to Millimeters)

Pentru a transforma măsurătorile din pixeli în unități metrice reale (milimetri), sistemul trebuie calibrat folosind un obiect de referință cu dimensiuni exact cunoscute.

1. Așezați în câmpul vizual ținta etalon (pătratul etalon de 20 × 20 mm);
2. Algoritmul detectează conturul etalonului și determină lățimea acestuia în pixeli (W_{pix});
3. Calculează factorul de scară metrică:

$$\text{Factor de scară } (K) = \frac{\text{Lățime reală etalon [mm]}}{\text{Lățime măsurată [pixeli]}} = \frac{20.0}{W_{\text{pix}}} [\text{mm/pixel}]$$

4. Toate dimensiunile calculate ulterior pentru piese vor rezulta prin înmulțirea distanței în pixeli cu această constantă K .

4.4. Algoritmul de inspecție dimensională și detecție a defectelor

Fluxul de procesare implementat în scriptul Python din **Anexa 1** execută ciclic următorii pași:

1. **Preluare cadru video:** captează imaginea color și o redimensionează la rezoluția de inspecție (640 × 480 sau 1280 × 720 pixeli);

2. **Conversie în tonuri de gri și filtrare:** convertește imaginea prin `cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)` și aplică o filtrare Gaussiană (`cv2.GaussianBlur`) cu mască 5×5 pentru eliminarea zgomotului de senzor;
3. **Segmentare și binarizare:** aplică pragul adaptiv Otsu (`cv2.threshold` cu parametrul `THRESH_OTSU`) pentru separarea clară a piesei (alb) de fundal (negru);
4. **Extragerea conturilor:** apelează `cv2.findContours` pentru identificarea marginilor exterioare și interioare ale piesei;
5. **Metrologie dimensională:**
 - Calculează cercul minim circumscris piesei (`cv2.minEnclosingCircle`), obținând raza și diametrul exterior în pixeli și milimetri ($D_{\text{ext}} = 2 \cdot R \cdot K$);
 - Calculează aria conturului (`cv2.contourArea`) și perimetrul (`cv2.arcLength`);
 - Determină circularitatea: $\text{Circularitate} = \frac{4 \cdot \pi \cdot \text{Arie}}{\text{Perimetru}^2}$ (o valoare apropiată de 1.0 indică un cerc perfect; o valoare sub 0.85 semnalează deformări sau bavuri);
 - Detectează prezența găurii interioare prin verificarea conturilor de tip copil (*hierarchy*).
6. **Luarea deciziei de calitate:**
 - Dacă dimensiunile se încadrează în limitele prescrise (de exemplu, diametru nominal $D_{\text{nom}} = 30.0 \text{ mm} \pm 1.0 \text{ mm}$ și circularitate > 0.88):
 - Activează **LED-ul Verde (GPIO 17)**;
 - Desenează pe ecran conturul verde și cotele măsurate;
 - Dacă oricare dintre condiții este încălcată (piesă spartă, ovalizată, diametru greșit sau bavură):
 - Activează **LED-ul Roșu și Releul de sortare (GPIO 27 / GPIO 22)**;
 - Desenează pe ecran conturul roșu cu textul „*REBUT: Dimensiune neconformă*”;
 - Salvează automat imaginea cadrului în folder-ul local `/home/pi/rebuturi/` cu timestamp în nume.

4.5. Rularea și testarea experimentală a stației AOI

1. Creați pe Raspberry Pi fișierul `inspectie_aoi.py` și introduceți codul complet din **Anexa 1**;
2. Rulați scriptul din terminal:

```
python3 inspectie_aoi.py
```
3. Așezați pe rând sub cameră o piesă conformă (etalon) și observați aprinderea LED-ului Verde și afișarea cotelor în fereastra grafică;
4. Introduceți o piesă defectă (sau acoperiți parțial marginea unei piese cu o bucătică de bandă adezivă neagră pentru a simula o fisură/ruptură). Constați comutarea instantanee a deciziei în *REBUT*, aprinderea LED-ului Roșu și salvarea fișierului imagine în directorul de rebuturi.

5. Exerciții și teme practice

- 5.1. Calibrarea automată la pornire: Modificați scriptul Python astfel încât la apăsarea tastei 'C' de la tastatură, programul să captureze cadrul curent, să identifice automat pătratul etalon de 20×20 mm și să recalculeze factorul de scară K , salvând valoarea într-o variabilă globală.
- 5.2. Detecția găurilor lipsă pe reper: Extindeți logica de clasificare pentru inspecția unei șaibe sau a unei flanșe: dacă conturul exterior este conform, dar algoritmul nu detectează niciun contur interior (gaura centrală este absentă sau înfundată), piesa trebuie declarată *REBUT*: *Gaură lipsă*.
- 5.3. Jurnalizarea automată a calității în fișier CSV: Scrieți o funcție care adaugă o înregistrare într-un fișier raport_inspecție.csv pentru fiecare piesă inspectată, salvând: *Data și Ora*, *ID piesă*, *Diametru [mm]*, *Circularitate*, *Decizie (OK/NOK)* și *Timpul de procesare [ms]*.
- 5.4. Calculul ratei de rebuturi și integrare Dashboard: Calculați în timp real procentul de rebuturi din lot ($\text{Rata Rebut} = \frac{\text{Piese Rebutate}}{\text{Piese Totale}} \times 100\%$) și afișați această valoare pe ecranul video cu culoare galbenă dacă depășește pragul de 5%.

6. Întrebări de verificare

- 6.1. De ce este necesară aplicarea unui filtru Gaussian înainte de binarizarea imaginii prin metoda Otsu?
- 6.2. Cum influențează variația înălțimii camerei față de bancul de lucru factorul de scară K și de ce în stațiile industriale de măsurare precisă se utilizează lentile telecentrice?
- 6.3. Care este diferența dintre inspecția dimensională clasică pe bază de contururi (metrologie) și detecția defectelor de suprafață bazată pe rețele neuronale profunde (Deep Learning)?
- 6.4. Ce măsuri de siguranță hardware și software trebuie implementate pentru a garanta că o piesă rebutată nu trece mai departe pe linie dacă procesorul de imagine se blochează temporar?

Anexa 1: Script Python complet pentru inspecție optică AOI și control GPIO pe Raspberry Pi 5

```
import cv2
import numpy as np
import time
import os
from datetime import datetime
from gpiozero import LED

# 1. Configurare pini GPIO pentru semnalizare si sortare
led_ok = LED(17) # GPIO 17: LED Verde (Piesa Conformă)
led_rebut = LED(27) # GPIO 27: LED Rosu (Piesa Neconformă)
releu_sortare = LED(22) # GPIO 22: Releu comanda cilindru respingere

# 2. Parametri metrologici si tolerante (valori in mm)
```

```

DIAMETRU_NOMINAL_MM = 30.0 # Diametrul dorit al piesei
TOLERANTA_MM = 1.5      # Toleranta admisa (+/- 1.5 mm)
CIRCULARITATE_MINIMA = 0.85 # Prag minim de circularitate (1.0 = cerc perfect)

# Factor de calibrare initial (mm per pixel) - se ajusteaza la calibrare
SCALE_FACTOR = 0.12 # 1 pixel = 0.12 mm (exemplu pentru camera la 18 cm inaltime)

# Creare director pentru salvarea imaginilor cu defecte
DIR_REBUTURI = "/home/pi/rebuturi"
if not os.path.exists(DIR_REBUTURI):
    os.makedirs(DIR_REBUTURI)

# 3. Initializare camera video (OpenCV VideoCapture sau Picamera2)
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

if not cap.isOpened():
    print("Eroare: Nu s-a putut deschide camera video!")
    exit()

print("Statia AOI a pornit. Apasati 'q' pentru a opri, 'c' pentru calibrare.")

while True:
    start_time = time.time()
    ret, frame = cap.read()
    if not ret:
        print("Eroare la capturarea cadrului video.")
        break

    # Creare copie pentru afisare adnotata
    display_frame = frame.copy()

    # 4. Preprocesare imagine
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

```

```

# Binarizare automata prin metoda Otsu (obiect alb, fundal negru)
_, thresh = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

# 5. Extragere contururi si ierarhie
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)

piesa_detectata = False
status_piesa = "ASTEPTARE PIESA"
culoare_status = (200, 200, 200)

for i, cnt in enumerate(contours):
    area_pixels = cv2.contourArea(cnt)

    # Filtrare contururi mici cauzate de zgomot (prag minim 1000 pixeli)
    if area_pixels > 1500 and hierarchy[0][i][3] == -1:
        piesa_detectata = True
        perimeter = cv2.arcLength(cnt, True)

        # Calcul cerc minim circumscris
        (x, y), radius_pixels = cv2.minEnclosingCircle(cnt)
        center = (int(x), int(y))
        radius = int(radius_pixels)

        # Conversie metrologica in milimetri
        diametru_masurat_mm = (radius_pixels * 2.0) * SCALE_FACTOR
        circularitate = (4 * np.pi * area_pixels) / (perimeter * perimeter) if perimeter > 0 else 0

# 6. Luarea deciziei de calitate
eroare_dimensiune = abs(diametru_masurat_mm - DIAMETRU_NOMINAL_MM)

if eroare_dimensiune <= TOLERANTA_MM and circularitate >= CIRCULARITATE_MINIMA:
    status_piesa = "CONFORM (OK)"
    culoare_status = (0, 255, 0)
    led_ok.on()
    led_rebut.off()
    releu_sortare.off()
else:

```

```

status_piesa = "REBUT (NOK)"
culoare_status = (0, 0, 255)
led_ok.off()
led_rebut.on()
releu_sortare.on() # Actionare cilindru sortare

# Salvare imagine rebut cu timestamp
timestamp_str = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
nume_fisier = f"{DIR_REBUTURI}/rebut_{timestamp_str}.jpg"
cv2.imwrite(nume_fisier, frame)

# Desenare elemente grafice pe ecran
cv2.drawContours(display_frame, [cnt], -1, culoare_status, 2)
cv2.circle(display_frame, center, radius, (255, 255, 0), 1)
cv2.circle(display_frame, center, 3, (0, 0, 255), -1)

# Afisare cote numerice masurate
text_cote = f"D: {diametru_masurat_mm:.1f} mm | Circ: {circularitate:.2f}"
cv2.putText(display_frame, text_cote, (center[0] - 80, center[1] - radius - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, culoare_status, 2)
break

if not piesa_detectata:
    led_ok.off()
    led_rebut.off()
    releu_sortare.off()

# Calcul timp de procesare per cadru (buget temporal)
processing_time_ms = (time.time() - start_time) * 1000

# Afisare panou informativ pe ecran
cv2.rectangle(display_frame, (10, 10), (320, 80), (40, 40, 40), -1)
cv2.putText(display_frame, f"STATUS: {status_piesa}", (20, 35),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, culoare_status, 2)
cv2.putText(display_frame, f"Timp procesare: {processing_time_ms:.1f} ms", (20, 65),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

```

```

# Afisare fereastra video
cv2.imshow("Statie Inspectie Optica AOI - Raspberry Pi 5", display_frame)

key = cv2.waitKey(1) & 0xFF
if key == ord('q'):
    break
elif key == ord('c'):
    print("Functie de calibrare etalon declansata...")

# Eliberare resurse
led_ok.off()
led_rebut.off()
releu_sortare.off()
cap.release()
cv2.destroyAllWindows()

```

Lucrare de laborator 7: Simularea și programarea unei celule robotizate industriale de manipulare

1. Introducere

În informatica industrială și mecatronică, programarea roboților industriali direct pe celula fizică reală implică riscuri majore de coliziune mecanică, timpuri morți de producție și costuri ridicate de echipare. Așa cum s-a detaliat în Capitolul 9 (*Robotică industrială, cinematică și coboți*) și Studiul de caz 7, practica inginerescă modernă se bazează pe simularea 3D și punerea în funcțiune virtuală (*Virtual Commissioning*).

Această lucrare de laborator utilizează simulatorul de robotică **Webots** (platformă profesională gratuită și open-source dezvoltată de Cyberbotics):

1. **Mediu fizic 3D complet:** Webots include un motor fizic rigid avansat (ODE – *Open Dynamics Engine*) care simulează forțele de gravitație, inerția brațului, frecarea la contact și dinamica gripper-ului;
2. **Model industrial standardizat:** se utilizează manipulatorul serial articulat cu 6 grade de libertate (6 DOF) **Universal Robots UR5e**, echipat cu un efector final de tip gripper paralel cu două degete (Robotiq 2F-85);
3. **Control software în Python:** dezvoltarea unui controller autonom bazat pe o mașină de stări finită care preia piese de pe un conveyer de alimentare și le poziționează ordonat într-o cutie de ambalare (*Pick & Place*), integrând o buclă de oprire de securitate la detectarea obstacolelor.

2. Obiectivele lucrării

- Înțelegerea structurii cinemate a unui manipulator serial cu 6 grade de libertate (bază, umăr, cot și cele trei articulații ale încheieturii: *wrist 1*, *wrist 2*, *wrist 3*);
- Însușirea funcțiilor API-ului Webots pentru comanda motoarelor articulare (*Position Control / Velocity Control*) și citirea traductoarelor de poziție (*PositionSensors*);
- Implementarea unei mașini de stări secvențiale pentru ciclul industrial de manipulare: *Home* → *Abordare piesă* → *Coborâre* → *Strângere gripper* → *Ridicare* → *Deplasare* → *Eliberare*;
- Coordonarea mișcării gripper-ului cu dinamica de contact pentru asigurarea unei prinderi stabile fără alunecarea piesei;
- Integrarea unui senzor de distanță / barieră optică pentru oprirea automată a mișcării în cazul apariției unui obstacol neprevăzut pe traiectorie;
- Măsurarea și optimizarea timpului de ciclu (*takt time*) al operației de paletizare.

3. Materiale și resurse software necesare

3.1. Componente Hardware

- Calculator de laborator (sistem de operare Windows, cu cerințe hardware modeste).

3.2. Resurse Software

- **Webots** (versiunea recentă R2023b / R2024a sau ulterioară) – descărcabil gratuit de pe site-ul oficial: <https://cyberbotics.com/>;
- **Python 3** (integrat nativ în Webots sau instalat local).

4. Desfășurarea lucrării

4.1. Arhitectura de control a celulei robotizate în Webots

Arhitectura de control și fluxul informațional sunt ilustrate în Figura 7.1.



Figura 7.1 Arhitectura de control și fluxul de date în simulatorul Webots

4.2. Instalarea și deschiderea scenei industriale în Webots

1. Descărcați și rulați fișierul de instalare Webots pentru Windows (webots-setup.exe);
2. Porniți programul Webots;
3. În meniul superior, navigați la **File → Open Sample World...**;
4. În lista de exemple, accesați categoria *robots* → *universal_robots* → *ur5e.wbt* (sau deschideți fișierul de laborator furnizat de cadrul didactic ce conține celula completă: robot UR5e, masă de lucru, cutie și piese prismatice);
5. Lumea virtuală 3D se va deschide afișând brațul robotizat UR5e gata echipat cu gripper-ul Robotiq.

4.3. Identificarea motoarelor și a senzorilor robotului

Robotul UR5e conține 6 servomotoare de rotație corespunzătoare celor 6 axe de mișcare. În scriptul Python, aceste motoare sunt adresate prin numele lor din arborele scenei:

- *shoulder_pan_joint* – rotația bazei în plan orizontal ($[-, 2\pi + 2\pi]$ rad);
- *shoulder_lift_joint* – mișcarea umărului ($[-, 2\pi + 2\pi]$ rad);
- *elbow_joint* – mișcarea cotului ($[-, 2\pi + 2\pi]$ rad);

- *wrist_1_joint*, *wrist_2_joint*, *wrist_3_joint* – cele 3 axe de orientare a încheieturii și a efectorului final;
- *finger_1_joint_1*, *finger_2_joint_1* – motoarele degetelor paralele ale gripper-ului.

Fiecare motor dispune de un senzor de poziție asociat (*PositionSensor*) care returnează unghiul curent al articulației în radiani.

4.4. Crearea și configurarea controller-ului Python

1. În meniul Webots, selectați **Wizards** → **New Robot Controller...**;
2. Alegeți limbajul **Python** și denumiți controller-ul `ur5e_pick_and_place`;
3. În arborele din stânga (*Scene Tree*), selectați nodul **UR5e**, derulați la câmpul controller și selectați din listă `ur5e_pick_and_place`;
4. În editorul de text integrat din partea dreaptă a ferestrei Webots, înlocuiți codul existent cu scriptul complet din **Anexa 1**;
5. Salvați fișierul (Ctrl + S).

4.5. Implementarea mașinii de stări pentru secvența Pick & Place

Pentru a asigura o mișcare fluidă și reproductibilă, secvența de manipulare este împărțită în 6 stări discrete:

1. **Starea 0 (HOME):** robotul se deplasează într-o poziție de repaus sigură, deschizând complet gripper-ul;
2. **Starea 1 (ABORD):** brațul se poziționează deasupra piesei de pe masa de alimentare;
3. **Starea 2 (COBORARE):** efectorul coboară vertical pe axa Z până când piesa se află între degetele gripper-ului;
4. **Starea 3 (PRINDERE):** motoarele gripper-ului se închid exercitând o forță controlată asupra piesei;
5. **Starea 4 (RIDICARE):** brațul ridică piesa pe verticală pentru a evita coliziunea cu masa;
6. **Starea 5 (DEPLASARE & DESCARCARE):** robotul se rotește către cutia de destinație, coboară și deschide gripper-ul, eliberând piesa.

După eliberare, secvența se reia automat pentru următoarea piesă.

4.6. Rularea simulării și testarea interblocărilor de securitate

1. Apăsați butonul **Play / Real-Time Run** din bara de unelte superioară a simulatorului Webots;
2. Urmăriți mișcarea 3D a brațului și jurnalele de stare afișate în consola inferioară (*Console*);
3. **Testul de oprire de urgență:** În timpul deplasării brațului, introduceți un obstacol virtual în fața senzorului de proximitate (sau modificați variabila `obstacol_detectat = True` în consolă). Constatați oprirea imediată a motoarelor și menținerea poziției curente până la eliberarea zonei.

5. Exerciții și teme practice

- 5.1. Paletizare în matrice 2×2 : Modificați pozițiile articulare de descărcare din Starea 5 astfel încât robotul să nu așeze toate piesele în același punct, ci să le distribuie succesiv în 4 poziții distincte pe un caroi de tip matrice 2×2 (reprezentând un palet industrial).
- 5.2. Optimizarea profilului de viteză și a timpului de ciclu: Ajustați viteza unghiulară maximă a motoarelor de la 1.0 rad/s la 2.5 rad/s prin funcția `motor.setVelocity()`. Măsurați timpul total necesar manipulării a 5 piese și identificați viteza maximă la care piesa nu scapă din gripper din cauza forțelor de inerție la viraje.
- 5.3. Detecția scăpării piesei din gripper: Implementați o verificare de calitate: după comanda de închidere a gripper-ului (Starea 3), citiți senzorul de poziție al degetelor. Dacă degetele s-au închis complet (poziție apropiată de 0 mm), înseamnă că piesa lipsește (prindere ratată); în acest caz, robotul trebuie să sară peste etapa de deplasare și să revină direct în starea *HOME*.
- 5.4. Semnalizare optică HMI a stării robotului: Adăugați în scenă un nod luminos de tip LED virtual (sau afișați mesaje colorate în consolă) care semnalizează: Verde = *Ciclu în execuție*, Galben = *Apropiere piesă*, Roșu = *Oprire de securitate*.

6. Întrebări de verificare

- 6.1. Care este diferența dintre cinematica directă (calculul poziției X, Y, Z a vârfului pornind de la unghiurile articulațiilor) și cinematica inversă în comanda unui robot industrial?
- 6.2. De ce mișcarea de ridicare a piesei după prindere trebuie executată pe o traiectorie strict verticală și nu direct pe diagonală către punctul de descărcare?
- 6.3. Ce forțe fizice acționează asupra piesei transportate la viteze mari și cum influențează coeficientul de frecare al degetelor gripper-ului stabilitatea prinderii?
- 6.4. Ce avantaje oferă utilizarea unui simulator precum Webots în etapa de proiectare a unei celule robotizate înainte de achiziția echipamentelor fizice reale?

Anexa 1: Script Python complet pentru controller-ul robotului UR5e în Webots

```
from controller import Robot, Motor, PositionSensor, DistanceSensor
import time
```

```
# Initializare instantă robot
```

```
robot = Robot()
```

```
# Pasul de timp al simulării (milisecunde)
```

```
TIME_STEP = int(robot.getBasicTimeStep())
```

```
# 1. Numele motoarelor articulare ale robotului UR5e
```

```
joint_names = [
    "shoulder_pan_joint",
    "shoulder_lift_joint",
```

```
"elbow_joint",  
"wrist_1_joint",  
"wrist_2_joint",  
"wrist_3_joint"
```

```
]
```

2. Initializare motoare brat

```
motors = []  
for name in joint_names:  
    motor = robot.getDevice(name)  
    motor.setVelocity(1.2) # Viteza nominala de rotatie (rad/s)  
    motor.setAcceleration(2.0)  
    motors.append(motor)
```

3. Initializare motoare gripper Robotiq

```
left_finger = robot.getDevice("finger_1_joint_1")  
right_finger = robot.getDevice("finger_2_joint_1")  
left_finger.setVelocity(0.5)  
right_finger.setVelocity(0.5)
```

4. Initializare senzori de pozitie articulara

```
sensors = []  
for name in joint_names:  
    sensor = robot.getDevice(name + "_sensor")  
    if sensor:  
        sensor.enable(TIME_STEP)  
        sensors.append(sensor)
```

Initializare senzor de proximitate pentru securitate (optional in scena)

```
dist_sensor = robot.getDevice("distance_sensor")  
if dist_sensor:  
    dist_sensor.enable(TIME_STEP)
```

5. Definirea pozitiilor articulare cheie [radiani] pentru fiecare stare

Format: [shoulder_pan, shoulder_lift, elbow, wrist_1, wrist_2, wrist_3]

```
POS_HOME = [0.00, -1.57, 0.00, -1.57, 0.00, 0.00]
```

```
POS_PICK_ABOVE = [0.00, -1.00, 1.20, -1.77, -1.57, 0.00]
```

```

POS_PICK_DOWN = [ 0.00, -0.65, 1.60, -2.52, -1.57, 0.00]
POS_PLACE_ABOVE= [ 1.57, -1.00, 1.20, -1.77, -1.57, 0.00]
POS_PLACE_DOWN = [ 1.57, -0.65, 1.60, -2.52, -1.57, 0.00]

```

```

def set_arm_position(target_positions):
    """Trimite comenzile de unghi catre cele 6 servomotoare"""
    for i in range(6):
        motors[i].setPosition(target_positions[i])

```

```

def set_gripper(position):
    """Comanda deschiderea (0.0) sau inchiderea (0.045) degetelor gripper-ului"""
    left_finger.setPosition(position)
    right_finger.setPosition(position)

```

```

# Variabile masina de stari

```

```

STATE_HOME = 0
STATE_APPROACH_PICK = 1
STATE_LOWER_PICK = 2
STATE_GRASP = 3
STATE_LIFT_PICK = 4
STATE_MOVE_PLACE = 5
STATE_LOWER_PLACE = 6
STATE_RELEASE = 7

```

```

current_state = STATE_HOME
timer_counter = 0
cycle_count = 0

```

```

print("Controller-ul UR5e a pornit. Incepe secventa Pick & Place...")

```

```

# Bucla principala de control

```

```

while robot.step(TIME_STEP) != -1:

```

```

    # 6. Verificare conditie de securitate (oprire la obstacol)

```

```

    if dist_sensor and dist_sensor.getValue() < 100.0:

```

```

        print("[ALERTA SECURITATE] Obstacol detectat in zona celulei! Miscare oprita.")

```

```

        for m in motors:

```

```

        m.setVelocity(0.0)
    continue
else:
    for m in motors:
        m.setVelocity(1.2)

# 7. Executie masina de stari finite
timer_counter += 1

if current_state == STATE_HOME:
    set_arm_position(POS_HOME)
    set_gripper(0.0) # Gripper deschis
    if timer_counter > 60: # Temporizare pentru stabilizare miscare (~2 secunde)
        print("Stare: Deplasare deasupra piesei (Pick Above)")
        current_state = STATE_APPROACH_PICK
        timer_counter = 0

elif current_state == STATE_APPROACH_PICK:
    set_arm_position(POS_PICK_ABOVE)
    if timer_counter > 60:
        print("Stare: Coborare pe piesa (Pick Down)")
        current_state = STATE_LOWER_PICK
        timer_counter = 0

elif current_state == STATE_LOWER_PICK:
    set_arm_position(POS_PICK_DOWN)
    if timer_counter > 50:
        print("Stare: Prindere piesa (Grasp)")
        current_state = STATE_GRASP
        timer_counter = 0

elif current_state == STATE_GRASP:
    set_gripper(0.045) # Inchidere degete gripper
    if timer_counter > 40:
        print("Stare: Ridicare piesa (Lift)")
        current_state = STATE_LIFT_PICK
        timer_counter = 0

```

```

elif current_state == STATE_LIFT_PICK:
    set_arm_position(POS_PICK_ABOVE)
    if timer_counter > 50:
        print("Stare: Deplasare catre zona de descarcare (Place Above)")
        current_state = STATE_MOVE_PLACE
        timer_counter = 0

elif current_state == STATE_MOVE_PLACE:
    set_arm_position(POS_PLACE_ABOVE)
    if timer_counter > 60:
        print("Stare: Coborare in cutie (Place Down)")
        current_state = STATE_LOWER_PLACE
        timer_counter = 0

elif current_state == STATE_LOWER_PLACE:
    set_arm_position(POS_PLACE_DOWN)
    if timer_counter > 50:
        print("Stare: Eliberare piesa (Release)")
        current_state = STATE_RELEASE
        timer_counter = 0

elif current_state == STATE_RELEASE:
    set_gripper(0.0) # Deschidere gripper
    if timer_counter > 40:
        cycle_count += 1
        print(f"[SUCCES] Piesa {cycle_count} paletizata cu succes! Revenire Home.")
        current_state = STATE_HOME
        timer_counter = 0

```

Lucrare de laborator 8: Modelarea unui Geamăn Digital (Digital Twin) și Punere în Funcțiune Virtuală Hardware-in-the-Loop (HIL)

1. Introducere

În ingineria sistemelor cyber-fizice (CPS), testarea și calibrarea algoritmilor de reglare direct pe echipamente industriale reale (cuptoare termice de mare putere, reactoare chimice sau prese hidraulice) implică riscuri majore de avarie, consum inutil de energie și timpi mari de oprire a producției. Așa cum s-a detaliat în Capitolul 7 (*Identificarea sistemelor, MPC și Gemeni Digitali*), soluția modernă constă în crearea unui **Geamăn Digital (Digital Twin)** și realizarea unei puneri în funcțiune virtuale în buclă închisă de tip **Hardware-in-the-Loop (HIL)**.

Această lucrare de laborator realizează o punte sincronă în timp real între cele două lumi:

1. **Lumea fizică (Hardware-ul real):** placa **Arduino Uno** joacă rolul controlerului industrial real din fabrică. Aceasta citește referința dorită (*SetPoint*) de la un potențiometru fizic pe intrarea analogică A0, rulează algoritmul de reglare în timp real și generează comanda pe un LED fizic (modulație PWM);
2. **Lumea virtuală (Geamănul Digital):** calculatorul de proces rulează un script **Python** care integrează numeric ecuația diferențială a procesului termic, simulând inerția termică, pierderile spre mediu și perturbațiile exterioare;
3. **Bucă închisă HIL:** la fiecare pas fix de 100 ms, simulatorul transmite temperatura calculată T_{sim} către Arduino prin cablul USB Serial, iar Arduino calculează și transmite înapoi comanda de încălzire $u(t) \in [0, 100\%]$.

2. Obiectivele lucrării

- Înțelegerea conceptului de Geamăn Digital activ și a arhitecturii de testare Hardware-in-the-Loop (HIL);
- Formularea și discretizarea modelului matematic al unui proces termic industrial de ordinul 1 cu pierderi:

$$C \frac{dT}{dt} = P_{\text{in}}(u) - k(T - T_{\text{amb}})$$

- Implementarea simulatorului de proces fizic în Python cu integrare numerică Euler și afișare grafică în timp real;
- Programarea algoritmului de reglare pe microcontrolerul fizico-experimental Arduino Uno (regulator bipozițional cu histerezis și regulator PID discret);
- Stabilirea protocolului de sincronizare serială deterministă la pas fix de eșantionare ($T_s = 100$ ms);
- Analiza răspunsului în timp: evaluarea timpului de stabilire, a erorii staționare și a capacității controlerului de a respinge perturbațiile simulate (ex. deschiderea unei uși a incintei termice).

3. Materiale și componente necesare

3.1. Componente Hardware

- Placă de dezvoltare **Arduino Uno** (cu cablu USB de conectare la PC);
- 1x Potențiomtru rotativ liniar (pentru reglarea manuală a temperaturii de referință);
- 1x LED + 1x rezistență de 220 Ω (reprezintă matorul fizic al rezistenței de încălzire);
- 1x breadboard și cabluri de conexiune.

3.2. Resurse Software

- **Arduino IDE** (pentru programarea algoritmului de control pe Arduino Uno);
- **Python 3** (distribuția portabilă WinPython, Thonny sau Python standard) cu bibliotecile: *pyserial*, *numpy*, *matplotlib*.

4. Desfășurarea lucrării

4.1. Arhitectura buclei închise Hardware-in-the-Loop (HIL)

Arhitectura funcțională a Geamănului Digital și fluxul bidirecțional de date sunt prezentate în Figura 8.1.

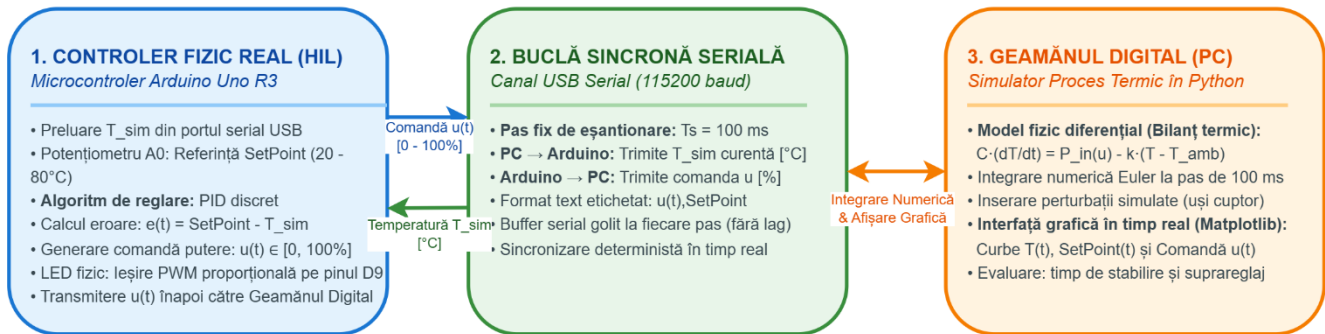


Figura 8.1 Arhitectura buclei închise Hardware-in-the-Loop (HIL) între Geamănul Digital din Python și Controlerul Fizic Arduino

Realizați conexiunile pe breadboard conform Tabelului 8.1:

Tabelul 8.1. Conexiunile potențiometrului și ale LED-ului la placa Arduino Uno

Componentă	Pin Componentă	Pin Arduino Uno	Rol funcțional
Potențiomtru	Pin 1 (Extremitate)	5V	Alimentare referință (+5 V)
Potențiomtru	Pin 2 (Cursor central)	Pin Analogic A0	Citire referință SetPoint (0–100°C)
Potențiomtru	Pin 3 (Extremitate)	GND	Masă (0 V)
LED Încălzire	Anod (+) prin 220 Ω	Pin Digital 9 (PWM)	Comandă proporțională cu puterea $u(t)$
LED Încălzire	Catod (-)	GND	Închiderea circuitului de masă

4.2. Modelul matematic al Geamănului Digital (Simulatorul Python)

Procesul simulat pe calculator reprezintă o încălțat termic industrială încălțat de un element rezistiv comandat în putere și răcit pasiv prin convecție spre mediul ambiant. Ecuația diferențială continuă este:

$$C \frac{dT(t)}{dt} = P_{\max} \cdot \frac{u(t)}{100} - k \cdot (T(t) - T_{\text{amb}})$$

unde:

- $T(t)$ este temperatura instantanee a incintei [$^{\circ}\text{C}$];
- $T_{\text{amb}} = 20.0^{\circ}\text{C}$ este temperatura mediului ambiant;
- $C = 50.0 \text{ J}/^{\circ}\text{C}$ este capacitatea termică a incintei;
- $k = 1.2 \text{ W}/^{\circ}\text{C}$ este coeficientul de transfer termic spre exterior;
- $P_{\max} = 100.0 \text{ W}$ este puterea maximă a rezistenței de încălzire;
- $u(t) \in [0, 100\%]$ este comanda calculată de placa Arduino.

Discretizarea ecuației prin metoda Euler la pasul de eșantionare $T_s = 0.1 \text{ s}$ devine:

$$T_{k+1} = T_k + \frac{T_s}{C} \cdot \left[P_{\max} \cdot \frac{u_k}{100} - k \cdot (T_k - T_{\text{amb}}) \right]$$

4.3. Implementarea algoritmului de control pe Arduino Uno

Controlerul fizic Arduino primește prin portul serial temperatura curentă calculată de model (T_{sim}), citește tensiunea de pe potențiometrul A0 (scalată în intervalul 20°C – 80°C) și calculează eroarea de reglare:

$$e(t) = \text{SetPoint} - T_{\text{sim}}$$

Controlerul implementează un algoritm **PID discret** (sau bipozițional cu histerezis $\pm 1^{\circ}\text{C}$):

$$u(t) = K_p \cdot e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$$

Comanda calculată $u(t)$ este saturată în intervalul $[0, 100\%]$, este aplicată fizic pe LED prin semnal PWM (`analogWrite(9, u * 2.55)`) și este transmisă înapoi pe portul serial către simulatorul Python.

Încărcați pe placa Arduino codul din **Anexa 1**.

4.4. Rularea simulării HIL și analiza Geamănului Digital în Python

1. Conectați placa Arduino la calculator prin cablul USB;
2. Verificați portul serial alocat (ex. COM1 pe Windows sau `/dev/ttyUSB0` pe Linux);
3. Porniți scriptul Python din **Anexa 2**;
4. Pe ecran se va deschide o fereastră grafică interactivă *Matplotlib* cu două panouri:
 - **Graficul superior:** afișează evoluția în timp a temperaturii simulate $T(t)$ (linie roșie) în raport cu referința dorită $\text{SetPoint}(t)$ (linie albastră);
 - **Graficul inferior:** afișează comanda de putere $u(t)$ calculată de Arduino (linie verde);
5. Rotiți potențiometrul conectat la Arduino și observați cum referința se modifică instantaneu pe ecran, iar Arduino calculează comanda optimă pentru a aduce Geamănul Digital la noua temperatură;

6. **Testul de respingere a perturbațiilor:** Apăsăți tasta 'd' în fereastra Python pentru a simula „Deschiderea unei uși a cuptorului” (creșterea bruscă a coeficientului de pierderi k de la 1.2 la 3.5). Observați cum temperatura scade temporar, iar Arduino crește automat puterea $u(t)$ pentru a compensa pierderile.

5. Exerciții și teme practice

- 5.1. Acordarea parametrilor regulatorului PID pe Arduino: Modificați coeficienții K_p , K_i și K_d în codul Arduino. Analizați pe graficul Python influența fiecărui termen:

- Mărirea lui K_p : răspuns mai rapid, dar risc de oscilații;
- Introducerea termenului integral K_i : eliminarea completă a erorii staționare;
- Introducerea termenului derivativ K_d : amortizarea suprareglajului.

- 5.2. Simularea unui Geamăn Digital de nivel de rezervor hidraulic: Modificați ecuația diferențială din Python pentru a simula un rezervor de apă cu golire liberă:

$$\frac{dh(t)}{dt} = \frac{1}{A} \left(Q_{in}(u) - c_d \sqrt{2gh(t)} \right)$$

Verificați comportamentul nelinier al procesului la modificarea nivelului de referință.

- 5.3. Implementarea unei zone moarte / histerezis pe comandă: Modificați codul Arduino pentru a implementa o reglare bipozițională de tip termostat (On-Off) cu bandă moartă de $\pm 1.5^\circ\text{C}$ în jurul referinței. Comparați consumul energetic și frecvența de comutare a comenzii față de reglarea PID.

- 5.4. Evaluarea automată a indicilor de calitate ai reglării: Scrieți o funcție în Python care calculează la finalul simulării: *Timpul de creștere* (t_r), *Timpul de stabilire* (t_s la banda de $\pm 2\%$) și *Suprareglajul procentual* ($\sigma\% = \frac{T_{\max} - SP}{SP} \times 100\%$).

6. Întrebări de verificare

- 6.1. Care este diferența dintre o simulare pur software (Model-in-the-Loop – MIL) și o simulare Hardware-in-the-Loop (HIL)? De ce HIL este mai apropiată de realitatea industrială?
- 6.2. De ce pasul de eșantionare T_s trebuie să fie strict constant în ambele programe (Arduino și Python) pentru ca integrarea numerică a Geamănului Digital să fie corectă?
- 6.3. Ce fenomen nedorit (saturație integrală / *Windup*) apare la regulatoarele PID dacă comanda atinge limita de 100% și cum poate fi prevenit?
- 6.4. Cum contribuie utilizarea unui Geamăn Digital la reducerea costurilor de punere în funcțiune (*Commissioning*) a unei fabrici inteligente?

Anexa 1: Codul C++ pe microcontrolerul Arduino Uno (Regulator PID & Comunicație HIL)

```
// Pini hardware
const int POT_PIN = A0; // Potentiometru referinta (SetPoint)
const int LED_PIN = 9;  // LED comanda incalzire (PWM)
// Parametri temporali bucla HIL
unsigned long lastTime = 0;
```

```

const unsigned long SAMPLE_TIME_MS = 100; // Pas fix Ts = 100 ms
// Variabile de proces
float setPoint = 40.0; // Temperatura dorita (20 - 80 °C)
float tempSimulata = 20.0; // Temperatura primita de la Geamanul Digital
float controlOutput = 0.0; // Comanda putere u(t) in procente (0 - 100 %)
// Coeficienti regulator PID
float Kp = 4.0;
float Ki = 0.15;
float Kd = 1.2;
float integralTerm = 0.0;
float lastError = 0.0;

void setup() {
  Serial.begin(115200);
  pinMode(POT_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // 1. Receptie temperatura de la simulatorul Python (Geamanul Digital)
  if (Serial.available() > 0) {
    String inputStr = Serial.readStringUntil('\n');
    inputStr.trim();
    if (inputStr.length() > 0) {
      tempSimulata = inputStr.toFloat(); } }

  // 2. Executie bucla de reglare la fiecare 100 ms
  unsigned long now = millis();
  if (now - lastTime >= SAMPLE_TIME_MS) {
    lastTime = now;
    // Citire potentiometru si scalare SetPoint intre 20.0 si 80.0 °C
    int potRaw = analogRead(POT_PIN);
    setPoint = 20.0 + ((float)potRaw / 1023.0) * 60.0;
    // Calcul eroare
    float error = setPoint - tempSimulata;
    // Calcul termen integral cu protectie anti-windup
    integralTerm += (error * (SAMPLE_TIME_MS / 1000.0));
  }
}

```

```

if (integralTerm > 50.0) integralTerm = 50.0;
if (integralTerm < -50.0) integralTerm = -50.0;
// Calcul termen derivativ
float dError = (error - lastError) / (SAMPLE_TIME_MS / 1000.0);
lastError = error;
// Calcul comanda PID
controlOutput = Kp * error + Ki * integralTerm + Kd * dError;
// Saturare comanda intre 0% si 100%
if (controlOutput > 100.0) controlOutput = 100.0;
if (controlOutput < 0.0) controlOutput = 0.0;
// Aplicare comanda fizica pe LED prin PWM (0 - 255)
analogWrite(LED_PIN, (int)(controlOutput * 2.55));
// 3. Transmite pachet de raspuns catre Geamanul Digital
// Format: "COMANDA,SETPOINT"
Serial.print(controlOutput, 1);
Serial.print(",");
Serial.println(setPoint, 1); } }

```

Anexa 2: Script Python pe PC pentru Simulatorul Geamănului Digital (Model Termic & GUI)

```

import serial
import time
import numpy as np
import matplotlib.pyplot as plt

# 1. Configurare port serial (ajustati portul: 'COM1', 'COM2', etc. pe Windows sau '/dev/ttyUSB0' pe Linux)
SERIAL_PORT = 'COM1'
BAUD_RATE = 115200

try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.5)
    time.sleep(2) # Asteptare resetare Arduino
    print(f"Conectat la Arduino pe {SERIAL_PORT}")
except Exception as e:
    print(f"Eroare port serial: {e}")
    exit()

# 2. Parametri fizici ai Geamanului Digital (Proces Termic)

```

```

C = 50.0    # Capacitate termica (J/°C)
k = 1.2     # Coeficient pierderi (W/°C)
P_MAX = 100.0 # Putere maxima rezistenta (W)
T_AMB = 20.0 # Temperatura ambientala (°C)
TS = 0.1    # Pas de esantionare (100 ms)

# Stari curense
T_current = 20.0
u_current = 0.0
sp_current = 40.0

# Buffere pentru afisare grafica (ultimele 150 de puncte = 15 secunde)
BUFFER_LEN = 150
time_history = np.zeros(BUFFER_LEN)
temp_history = np.ones(BUFFER_LEN) * 20.0
sp_history = np.ones(BUFFER_LEN) * 40.0
u_history = np.zeros(BUFFER_LEN)

# Configurare interfata grafica Matplotlib
plt.ion()
fig, (ax_temp, ax_cmd) = plt.subplots(2, 1, figsize=(10, 6), sharex=True)

line_sp, = ax_temp.plot(time_history, sp_history, 'b--', label='Referință SetPoint (°C)')
line_temp, = ax_temp.plot(time_history, temp_history, 'r-', linewidth=2, label='Geamă Digital: Temperatură T(t)')
ax_temp.set_ylabel('Temperatură (°C)')
ax_temp.set_ylim(15, 85)
ax_temp.grid(True)
ax_temp.legend(loc='upper left')
ax_temp.set_title('Co-simulare HIL: Geamă Digital (PC) & Controler PID Fizic (Arduino)')

line_u, = ax_cmd.plot(time_history, u_history, 'g-', linewidth=1.5, label='Comandă Putere u(t) [%]')
ax_cmd.set_xlabel('Timp (s)')
ax_cmd.set_ylabel('Comandă u (%)')
ax_cmd.set_ylim(-5, 105)
ax_cmd.grid(True)
ax_cmd.legend(loc='upper left')

```

```

sim_time = 0.0
perturba_activata = False

def on_key(event):
    global k, perturba_activata
    if event.key == 'd':
        perturba_activata = not perturba_activata
        k = 3.5 if perturba_activata else 1.2
        print(f"[PERTURBAȚIE] Pierderi k = {k} (Ușă cuptor {'DESCHISĂ' if perturba_activata else 'ÎNCHISĂ'})")

fig.canvas.mpl_connect('key_press_event', on_key)

print("Incepe simularea HIL. Apasati tasta 'd' pe grafic pentru a simula deschiderea usii cuptorului.")

while True:
    step_start = time.time()

    # 1. Transmitere temperatura curenta T_sim catre Arduino
    msg_out = f"{T_current:.2f}\n"
    ser.write(msg_out.encode('ascii'))

    # 2. Receptie comanda u si SetPoint de la Arduino
    try:
        line_in = ser.readline().decode('ascii', errors='ignore').strip()
        if "," in line_in:
            parts = line_in.split(",")
            u_current = float(parts[0])
            sp_current = float(parts[1])
    except Exception:
        pass

    # 3. Integrare numerica Euler a Geamanului Digital:  $C \cdot (dT/dt) = P_{in} - k \cdot (T - T_{amb})$ 
    P_in = P_MAX * (u_current / 100.0)
    dT_dt = (P_in - k * (T_current - T_AMB)) / C
    T_current = T_current + dT_dt * TS

    # 4. Actualizare date grafice

```

```

sim_time += TS
time_history = np.roll(time_history, -1)
temp_history = np.roll(temp_history, -1)
sp_history = np.roll(sp_history, -1)
u_history = np.roll(u_history, -1)

time_history[-1] = sim_time
temp_history[-1] = T_current
sp_history[-1] = sp_current
u_history[-1] = u_current

line_temp.set_xdata(time_history)
line_temp.set_ydata(temp_history)
line_sp.set_xdata(time_history)
line_sp.set_ydata(sp_history)
line_u.set_xdata(time_history)
line_u.set_ydata(u_history)

ax_temp.set_xlim(time_history[0], time_history[-1])
plt.pause(0.001)

# 5. Respectare pas strict de 100 ms
elapsed = time.time() - step_start
if TS - elapsed > 0:
    time.sleep(TS - elapsed)

```

Lucrare de laborator 9: Sistem MES pentru calculul automatizat al indicatorului OEE și urmărirea producției

1. Introducere

În piramida de automatizare industrială definită prin standardul **ANSI/ISA-95** (analizată în Capitolul 10 și Studiul de caz 4), sistemul **MES (Manufacturing Execution System)** ocupă Nivelul 3 de autoritate, făcând legătura directă între comenzile comerciale din ERP (Nivelul 4) și utilajele fizice de pe linia de fabricație (Nivelul 1 OT). Metrica universală utilizată în fabrici pentru evaluarea performanței productive a unui utilaj sau a unei linii este **Eficiența Globală a Echipamentului (Overall Equipment Effectiveness – OEE)**, standardizată prin **ISO 22400-2**.

Această lucrare de laborator realizează un sistem MES industrial complet, oferind două modalități flexibile de implementare a interfeței operator:

- **Nivelul de proces (Simulator de mașină):** o placă **Arduino Uno** simulează senzorii de trecere ai unei linii de fabricație (detectarea pieselor conforme și a rebuturilor) și stările operaționale ale mașinii (*În Funcțiune / Avarie*);
- **Varianta A (Pe PC – Fără drepturi de administrator):** nucleul de calcul MES și panoul de bord HMI rulează direct în **WinPython / Thonny IDE** printr-un script grafic Python bazat pe biblioteca standard **tkinter**, calculând în timp real indicatorii OEE și afișând ceasul grafic de randament;
- **Varianta B (Pe Raspberry Pi 5 – Cu drepturi de administrator):** evenimentele sunt procesate în **Node-RED**, iar panoul de bord este expus ca pagină web în browser prin modulul **node-red-dashboard**.

2. Obiectivele lucrării

- Înțelegerea ierarhiei funcționale ISA-95 și a rolului sistemului MES în execuția și urmărirea comenzilor de fabricație;
- Însușirea formulei matematice a indicatorului OEE și a semnificației celor 3 componente fundamentale:

$$OEE = A \times P \times Q$$

- Implementarea pe microcontroler (Arduino Uno) a unui simulator de evenimente de mașină cu debounce hardware și emisie de pachete JSON pe portul serial;
- Configurarea algoritmului de cronometrare a timpilor de rulare și oprire (*Downtime*);
- Realizarea unei interfețe grafice de dispecerat (**HMI**) cu ceas vitezometru (*Gauge*), bare de progres pentru factorii *A, P, Q* și alerte la scăderea randamentului;
- Jurnalizarea automată a evenimentelor de oprire a liniei și exportul raportului de schimb în format CSV.

3. Materiale și componente necesare

3.1. Componente Hardware

- Placă de dezvoltare **Arduino Uno**;
- 2x Butoane fizice cu revenire:
 - Butonul 1: simulează senzorul optic pentru **Piesă Conformă (OK)**;
 - Butonul 2: simulează senzorul de inspecție pentru **Piesă Rebutată (NOK)**;
- 1x Comutator basculant cu două poziții (SPST) sau un al treilea buton: simulează starea mașinii (**RUN / AVARIE-STOP**);
- 2x LED (1x Verde = Mașină în funcțiune, 1x Roșu = Avarie/Oprită) + 2x rezistențe de 220 Ω ;
- 1x breadboard și cabluri de conexiune;
- PC de laborator (pentru Varianta A) sau placă **Raspberry Pi 5 8GB** (pentru Varianta B).

3.2. Resurse Software

- **Arduino IDE** (pentru programarea simulatorului de mașină pe Arduino);
- **Pentru Varianta A (PC):** Interpretator Python 3 portabil (**WinPython** dezarhivat sau **Thonny Portable**) cu pachetul pyserial;
- **Pentru Varianta B (Raspberry Pi 5):** **Node-RED** instalat cu modulul node-red-dashboard.

4. Desfășurarea lucrării

4.1. Arhitectura de sistem și conexiunile hardware

Arhitectura funcțională a sistemului MES și fluxul de evenimente sunt prezentate în Figura 9.1.

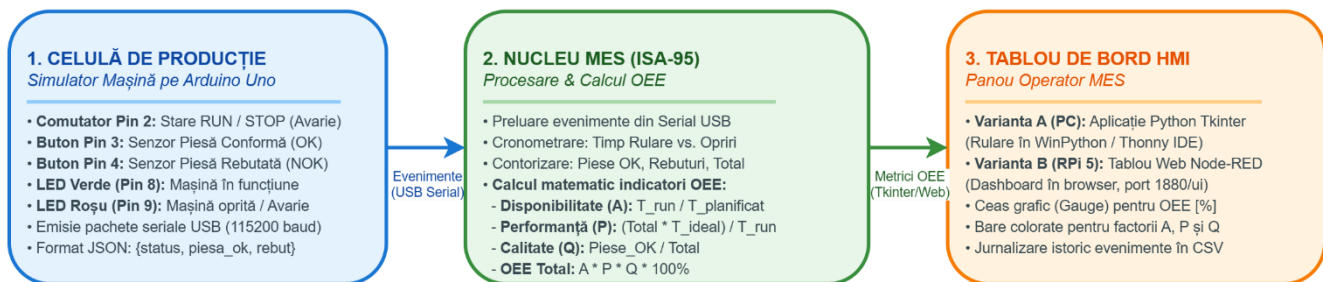


Figura 9.1 Arhitectura sistemului MES pentru achiziția evenimentelor de mașină și calculul indicatorului OEE

Realizați conexiunile pe breadboard conform Tabelului 9.1:

Tabelul 9.1. Conexiunile butoanelor și ale LED-urilor la placa Arduino Uno

Componentă	Pin Componentă	Pin Arduino Uno	Rol funcțional
Comutator Stare	Pin Contact (cu pull-up intern)	Pin Digital 2	Comutare stare: LOW = RUN, HIGH = AVARIE
Buton Piesă Bună	Pin Contact (cu pull-up intern)	Pin Digital 3	Simulare senzor trecere piesă conformă (OK)
Buton Rebut	Pin Contact (cu pull-up intern)	Pin Digital 4	Simulare senzor detecție piesă neconformă (NOK)
LED Verde (RUN)	Anod (+) prin 220 Ω	Pin Digital 8	Martor optic: Mașină în funcțiune normală
LED Roșu (AVARIE)	Anod (+) prin 220 Ω	Pin Digital 9	Martor optic: Mașină oprită / Avarie pe linie
Masa comună (GND)	Pinii comuni butoane / catodi LED	GND	Închiderea circuitului de masă (0 V)

4.2. Programarea simulatorului de celulă pe Arduino Uno

Programul Arduino monitorizează starea comutatorului și detectează apăsarea butoanelor cu filtrare software de debounce (50 ms). La fiecare eveniment, Arduino transmite către portul serial un mesaj text în format JSON structurat:

- La comutarea stării: {"status":"RUN"} sau {"status":"STOP"};
- La înregistrarea unei piese bune: {"piesa_ok":1};
- La înregistrarea unui rebut: {"rebut":1}.

Încărcați pe placa Arduino codul complet din **Anexa 1**.

4.3. Modelul matematic de calcul al indicatorului OEE

În sistemul MES, calculul OEE se bazează pe parametrii unei comenzi de producție:

- **Timpul total planificat de producție (T_{plan}):** durata totală alocată schimbului (pentru laborator = 600 secunde / 10 minute);
- **Timpul de funcționare efectivă (T_{run}):** timpul total cumulat în care mașina s-a aflat în starea RUN;
- **Timpul ideal de ciclu (T_{ideal}):** durata tehnologică minimă necesară producerii unei piese ($T_{\text{ideal}} = 3.0$ secunde/piesa);
- **Piese bune (Q_{bun}):** numărul de piese conforme contorizate;
- **Piese totale (Q_{total}):** suma pieselor bune și a rebuturilor ($Q_{\text{total}} = Q_{\text{bun}} + Q_{\text{rebut}}$).

Formulele de calcul aplicate la fiecare secundă sunt:

1. **Disponibilitatea (A):** $A = \frac{T_{run}}{T_{plan}}$ (saturat la maxim 1.0)
2. **Performanța (P):** $P = \frac{Q_{total} \cdot T_{ideal}}{T_{run}}$ (dacă $T_{run} > 0$ saturat la 1.0)
3. **Calitatea (Q):** $Q = \frac{Q_{bun}}{Q_{total}}$ (dacă $Q_{total} > 0$ altfel 1.0)
4. **Indicatorul OEE Total:** $OEE = A \times P \times Q \times 100\%$

4.4. Rularea interfeței HMI pe calculator (Varianta A – Python Tkinter)

1. Deschideți mediul **IDLE** sau **Thonny** din folder-ul portabil WinPython;
2. Creați fișierul `mes_oe_dashboard.py` și introduceți codul complet din **Anexa 2**;
3. Rulați scriptul apăsând tasta **F5**;
4. Pe ecran se va deschide tabloul de bord MES ce conține:
 - Indicatorul mare cu valoarea **OEE Total [%]** colorat dinamic: Verde ($> 85\%$), Galben ($60-85\%$), Roșu ($< 60\%$);
 - Barele grafice de progres pentru cei 3 factori (A, P, Q);
 - Contoarele numerice pentru piese bune, rebuturi și timpul de funcționare;
 - Butonul [**START SCHIMB NOU / RESET**].

(Pentru Varianta B pe Raspberry Pi 5, se configurează fluxul Node-RED utilizând codul din Anexa 3).

4.5. Testarea experimentală a scenariilor de producție

1. **Lansarea comenzii:** Apăsați butonul de *Reset / Start Schimb Nou*;
2. **Producție normală:** Treceți comutatorul în poziția RUN (LED-ul Verde se aprinde). Apăsați ritmic butonul de *Piesă Bună* la fiecare 3 secunde. Urmăriți cum factorii de Disponibilitate, Performanță și Calitate ating valori de 95–100%, iar OEE-ul indică zona verde;
3. **Simularea pierderilor de calitate:** Apăsați de câteva ori butonul de *Rebut*. Observați scăderea imediată a factorului de Calitate (Q) și impactul direct asupra OEE-ului total;
4. **Simularea opririlor neplanificate (Downtime):** Comutați pe poziția STOP (LED-ul Roșu se aprinde) timp de 1–2 minute. Observați degradarea rapidă a factorului de Disponibilitate (A) pe durata opririi.

5. Exerciții și teme practice

- 5.1. Declararea motivului de oprire în HMI: Adăugați în interfața Python (sau Node-RED) un meniu de selecție care se activează automat la trecerea mașinii în starea STOP, permițând operatorului să selecteze cauza opririi: 1. *Lipsă materie primă*, 2. *Blocaj mecanic*, 3. *Pauză tehnologică*.
- 5.2. Jurnalizarea automată a raportului în fișier CSV: Extindeți scriptul Python pentru a salva la fiecare 30 de secunde o linie într-un fișier istoric `oe.csv` cu formatul: *Timestamp, Disponibilitate, Performanță, Calitate, OEE_Total, Piese_Bune, Rebuturi*.

- 5.3. Calculul pierderilor financiare din rebuturi: Introduceți în calculul MES costul unitar al materiei prime (15 lei/rebut). Calculați și afișați pe ecran costul financiar total al pierderilor generate de piesele rebutate în schimbul curent.
- 5.4. Alarmă automată la degradarea indicatorului OEE: Implementați o regulă logică care afișează o notificare de avertizare vizuală dacă valoarea OEE scade sub pragul critic de 65% timp de mai mult de 30 de secunde consecutiv.

6. Întrebări de verificare

- 6.1. Care sunt cele „șase mari pierderi” (*Six Big Losses*) din mentenanța productivă totală (TPM) care afectează cei trei factori ai indicatorului OEE (Disponibilitate, Performanță, Calitate)?
- 6.2. De ce un utilaj care produce piese la o viteză dublă față de norma ideală ($P > 100\%$), dar generează 40% rebuturi, va avea un OEE scăzut?
- 6.3. Cum ajută standardul ANSI/ISA-95 la decuplarea responsabilităților dintre sistemul MES și controlerele logice din teren (PLC)?
- 6.4. Ce măsuri de securitate a datelor trebuie implementate pentru a garanta că operatorii nu pot modifica manual numărul de piese bune înregistrat de senzorii fizici?

Anexa 1: Codul C++ pe microcontrolerul Arduino Uno (Simulator Mașină de Producție)

```
// Pini conectare butoane si LED-uri
const int PIN_SWITCH_STATUS = 2; // Comutator RUN / STOP (cu INPUT_PULLUP)
const int PIN_BTN_GOOD    = 3; // Buton Piesa Buna OK (cu INPUT_PULLUP)
const int PIN_BTN_BAD     = 4; // Buton Piesa Rebut NOK (cu INPUT_PULLUP)

const int LED_RUN  = 8; // LED Verde (Masina in functiune)
const int LED_FAULT = 9; // LED Rosu (Masina oprita / Avarie)

// Variabile pentru debounce software
int lastSwitchState = HIGH;
int lastGoodBtnState = HIGH;
int lastBadBtnState = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long DEBOUNCE_DELAY = 50; // 50 ms

void setup() {
  Serial.begin(115200);

  pinMode(PIN_SWITCH_STATUS, INPUT_PULLUP);
  pinMode(PIN_BTN_GOOD, INPUT_PULLUP);
  pinMode(PIN_BTN_BAD, INPUT_PULLUP);
```

```

pinMode(LED_RUN, OUTPUT);
pinMode(LED_FAULT, OUTPUT);

// Stare initiala LED-uri
digitalWrite(LED_RUN, HIGH);
digitalWrite(LED_FAULT, LOW);

// Transmite stare initiala
Serial.println("{ \"status\": \"RUN\"}");
}

void loop() {
    int currentSwitch = digitalRead(PIN_SWITCH_STATUS);
    int currentGood = digitalRead(PIN_BTN_GOOD);
    int currentBad = digitalRead(PIN_BTN_BAD);

    // 1. Verificare schimbare stare masina (RUN / STOP)
    if (currentSwitch != lastSwitchState && (millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
        lastSwitchState = currentSwitch;
        lastDebounceTime = millis();

        if (currentSwitch == LOW) {
            digitalWrite(LED_RUN, HIGH);
            digitalWrite(LED_FAULT, LOW);
            Serial.println("{ \"status\": \"RUN\"}");
        } else {
            digitalWrite(LED_RUN, LOW);
            digitalWrite(LED_FAULT, HIGH);
            Serial.println("{ \"status\": \"STOP\"}");
        }
    }

    // 2. Verificare apasare buton Piesa Buna (doar daca masina este in RUN)
    if (currentGood == LOW && lastGoodBtnState == HIGH && (millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
        lastDebounceTime = millis();
        if (lastSwitchState == LOW) {

```

```

    Serial.println("{\\"piesa_ok\\":1}");
}
}

lastGoodBtnState = currentGood;

// 3. Verificare apasare buton Piesa Rebutata
if (currentBad == LOW && lastBadBtnState == HIGH && (millis() - lastDebounceTime) > DEBOUNCE_DELAY) {
    lastDebounceTime = millis();
    if (lastSwitchState == LOW) {
        Serial.println("{\\"rebut\\":1}");
    }
}
lastBadBtnState = currentBad;
}

```

Anexa 2: Script Python complet pentru aplicația MES / OEE pe PC (WinPython / Thonny)

```

import serial
import json
import time
import sys
import tkinter as tk
from tkinter import ttk

# 1. Configurare Port Serial
SERIAL_PORT = 'COM1' if sys.platform.startswith('win') else '/dev/ttyUSB0'
BAUD_RATE = 115200

try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.1)
    print(f"Conectat la Arduino pe {SERIAL_PORT}")
except Exception as e:
    print(f"Eroare port serial: {e}")
    ser = None

# 2. Variabile MES de productie
is_running = True
run_time_sec = 0

```

```

planned_time_sec = 600 # 10 minute
ideal_cycle_sec = 3.0 # 3 secunde per piesa
good_count = 0
bad_count = 0

# 3. Creare Interfata Grafica MES (Tkinter)
root = tk.Tk()
root.title("Sistem MES - Monitorizare OEE conform ISA-95")
root.geometry("520x560")
root.configure(bg="#eceff1")

# Titlu
lbl_title = tk.Label(root, text="DISPECERAT MES: CALCUL OEE ÎN TIMP REAL",
                    font=("Helvetica", 13, "bold"), bg="#1565c0", fg="white", pady=10)
lbl_title.pack(fill="x")

# Panou OEE Total
frame_oe = tk.LabelFrame(root, text="Eficiența Globală a Echipamentului (OEE) ",
                        font=("Helvetica", 11, "bold"), bg="#eceff1", padx=15, pady=10)
frame_oe.pack(pady=10, fill="x", padx=20)

lbl_oe_val = tk.Label(frame_oe, text="0.0 %", font=("Consolas", 32, "bold"), fg="#2e7d32", bg="#eceff1")
lbl_oe_val.pack()

lbl_oe_status = tk.Label(frame_oe, text="STANDBY", font=("Helvetica", 10, "bold"), fg="#546e7a", bg="#eceff1")
lbl_oe_status.pack()

# Panou Factori A, P, Q
frame_factors = tk.LabelFrame(root, text="Descompunerea Factorilor OEE (ISO 22400-2) ",
                            font=("Helvetica", 10, "bold"), bg="#eceff1", padx=15, pady=10)
frame_factors.pack(pady=5, fill="x", padx=20)

# Disponibilitate (A)
lbl_a = tk.Label(frame_factors, text="Disponibilitate (A): 0.0 %", font=("Helvetica", 10, "bold"), bg="#eceff1",
                anchor="w")
lbl_a.pack(fill="x")
prog_a = ttk.Progressbar(frame_factors, length=400, maximum=100)

```

```

prog_a.pack(pady=2)

# Performanta (P)
lbl_p = tk.Label(frame_factors, text="Performanță (P): 0.0 %", font=("Helvetica", 10, "bold"), bg="#eceff1",
anchor="w")
lbl_p.pack(fill="x")
prog_p = ttk.Progressbar(frame_factors, length=400, maximum=100)
prog_p.pack(pady=2)

# Calitate (Q)
lbl_q = tk.Label(frame_factors, text="Calitate (Q): 0.0 %", font=("Helvetica", 10, "bold"), bg="#eceff1", anchor="w")
lbl_q.pack(fill="x")
prog_q = ttk.Progressbar(frame_factors, length=400, maximum=100)
prog_q.pack(pady=2)

# Panou Contoare Piese si Timp
frame_stats = tk.Frame(root, bg="#eceff1")
frame_stats.pack(pady=8, fill="x", padx=20)

lbl_good = tk.Label(frame_stats, text="Piese Bune: 0", font=("Helvetica", 11, "bold"), fg="#2e7d32", bg="#eceff1")
lbl_good.grid(row=0, column=0, padx=10)

lbl_bad = tk.Label(frame_stats, text="Rebuturi: 0", font=("Helvetica", 11, "bold"), fg="#c62828", bg="#eceff1")
lbl_bad.grid(row=0, column=1, padx=10)

lbl_time_stat = tk.Label(frame_stats, text="Timp Rulare: 0 s", font=("Helvetica", 10), bg="#eceff1")
lbl_time_stat.grid(row=0, column=2, padx=10)

# Functie Reset
def reset_shift():
    global run_time_sec, good_count, bad_count, is_running
    run_time_sec = 0
    good_count = 0
    bad_count = 0
    is_running = True
    print("[MES] Schimb nou initializat!")

```

```

btn_reset = tk.Button(root, text="⏮ START SCHIMB NOU / RESET", font=("Helvetica", 10, "bold"),
                      bg="#f57c00", fg="white", command=reset_shift, pady=5)
btn_reset.pack(pady=10)

```

4. Bucla de calcul OEE si citire seriala

```
last_tick = time.time()
```

```
def update_mes():
```

```
    global is_running, run_time_sec, good_count, bad_count, last_tick
```

Citire pachete seriale de la Arduino

```
if ser and ser.in_waiting > 0:
```

```
    try:
```

```
        line = ser.readline().decode('utf-8', errors='ignore').strip()
```

```
        if line.startswith("{") and line.endswith("}"):

```

```
            data = json.loads(line)

```

```
            if "status" in data:

```

```
                is_running = (data["status"] == "RUN")

```

```
            if "piesa_ok" in data:

```

```
                good_count += data["piesa_ok"]

```

```
            if "rebut" in data:

```

```
                bad_count += data["rebut"]

```

```
except Exception:
```

```
    pass
```

Cronometrare timp de rulare la fiecare secunda

```
now = time.time()
```

```
if now - last_tick >= 1.0:
```

```
    last_tick = now

```

```
    if is_running:

```

```
        run_time_sec += 1

```

Calcul Formule OEE

```
total_pieces = good_count + bad_count
```

Factor A

```
A = (run_time_sec / planned_time_sec) if planned_time_sec > 0 else 0
```

```

A = min(A, 1.0)

# Factor P
P = ((total_pieces * ideal_cycle_sec) / run_time_sec) if run_time_sec > 0 else 0
P = min(P, 1.0)

# Factor Q
Q = (good_count / total_pieces) if total_pieces > 0 else 1.0

# OEE Total
OEE = A * P * Q * 100.0

# Actualizare Interfata Grafica
lbl_oe_val.config(text=f"{OEE:.1f} %")

# Colorare OEE
if OEE >= 85.0:
    lbl_oe_val.config(fg="#2e7d32")
    lbl_oe_status.config(text="EXCELENT (World Class OEE > 85%)", fg="#2e7d32")
elif OEE >= 60.0:
    lbl_oe_val.config(fg="#f57c00")
    lbl_oe_status.config(text="ACCEPTABIL (60% - 85%)", fg="#f57c00")
else:
    lbl_oe_val.config(fg="#c62828")
    lbl_oe_status.config(text="CRITIC - RANDAMENT SCĂZUT (< 60%)", fg="#c62828")

lbl_a.config(text=f"Disponibilitate (A): {A*100:.1f} %")
prog_a['value'] = A * 100

lbl_p.config(text=f"Performanță (P): {P*100:.1f} %")
prog_p['value'] = P * 100

lbl_q.config(text=f"Calitate (Q): {Q*100:.1f} %")
prog_q['value'] = Q * 100

lbl_good.config(text=f"Piese Bune: {good_count}")
lbl_bad.config(text=f"Rebuturi: {bad_count}")

```

```
lbl_time_stat.config(text=f"Timp Rulare: {run_time_sec} s ({'RUN' if is_running else 'STOP'})")
```

```
root.after(100, update_mes)
```

```
root.after(100, update_mes)
```

```
root.mainloop()
```

```
if ser: ser.close()
```

Anexa 3: Codul JavaScript pentru nodul Function din Node-RED (Varianta B pe Raspberry Pi 5)

```
// Initializare context persistent Node-RED
```

```
var state = context.get('state') || {
```

```
  running: true,
```

```
  runTimeSeconds: 0,
```

```
  plannedTimeSeconds: 600,
```

```
  idealCycleTimeSeconds: 3.0,
```

```
  goodCount: 0,
```

```
  badCount: 0
```

```
};
```

```
if (msg.payload === "RESET") {
```

```
  state.runTimeSeconds = 0;
```

```
  state.goodCount = 0;
```

```
  state.badCount = 0;
```

```
  state.running = true;
```

```
} else if (typeof msg.payload === 'object') {
```

```
  if (msg.payload.status === "RUN") state.running = true;
```

```
  if (msg.payload.status === "STOP") state.running = false;
```

```
  if (msg.payload.piesa_ok) state.goodCount += 1;
```

```
  if (msg.payload.rebut) state.badCount += 1;
```

```
} else if (msg.topic === "timer_tick") {
```

```
  if (state.running) {
```

```
    state.runTimeSeconds += 1;
```

```
  }
```

```
}
```

```
var totalCount = state.goodCount + state.badCount;
```

```

var A = (state.plannedTimeSeconds > 0) ? (state.runTimeSeconds / state.plannedTimeSeconds) : 0;
if (A > 1.0) A = 1.0;

var P = (state.runTimeSeconds > 0) ? ((totalCount * state.idealCycleTimeSeconds) / state.runTimeSeconds) : 0;
if (P > 1.0) P = 1.0;

var Q = (totalCount > 0) ? (state.goodCount / totalCount) : 1.0;

var OEE = A * P * Q * 100.0;

context.set('state', state);

return [
  { payload: parseFloat(OEE.toFixed(1)) },
  { payload: parseFloat((A * 100).toFixed(1)) },
  { payload: parseFloat((P * 100).toFixed(1)) },
  { payload: parseFloat((Q * 100).toFixed(1)) },
  { payload: state.goodCount },
  { payload: state.badCount }
];

```

Bibliografie

- [1] Gheorghe Sebestyen-Pal, *Informatica Industrială*. Bucuresti, Romania: Editura Albastra, 2006. Accessed: Sep. 22, 2026. [Online]. Available: https://biblioteca.utcluj.app:443/liberty/OpacLogin?mode=BASIC&openDetail=true&corporation=RO_UTCN&action=search&queryTerm=uuid%3D%22ece22547a9fed919668f003900056ac1%22&editionUuid=ece22547a9fed919668f003900056ac1&operator=OR&url=%2Fopac%2Fsearch.do
- [2] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 2011.
- [3] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, “Guide to Operational Technology (OT) Security,” National Institute of Standards and Technology, NIST Special Publication 800-82 Rev. 3, 2023. doi: 10.6028/NIST.SP.800-82r3.
- [4] International Electrotechnical Commission, “Security for Industrial Automation and Control Systems (IEC 62443).” IEC, Geneva, Switzerland, 2009. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standards-and-publications/iso-standards/isa-iec-62443-series-of-standards>
- [5] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao, and Zihé Liu, “Review on cyber-physical systems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 27–40, 2017, doi: 10.1109/JAS.2017.7510349.
- [6] N. Jazdi, “Cyber Physical Systems in the Context of Industry 4.0,” in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, 2014, pp. 1–4. doi: 10.1109/AQTR.2014.6857843.
- [7] International Electrotechnical Commission, “Enterprise-control system integration — Part 1: Models and terminology (IEC 62264-1).” IEC, Geneva, Switzerland, 2013. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iec:62264:-1:ed-2:v1:en,fr>
- [8] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson, “The industrial internet of things (IIoT): An analysis framework,” *Computers in Industry*, vol. 101, pp. 1–12, 2018, doi: 10.1016/j.compind.2018.04.015.
- [9] Kun Mean Hou, Xunxing Diao, Hongling Shi, Hao Ding, Haiying Zhou, and Christophe de Vault, “Trends and challenges in AIoT/IIoT/IoT implementation,” *Sensors*, vol. 23, no. 11, p. 5074, 2023, doi: 10.3390/s23115074.
- [10] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, “Framework for Cyber-Physical Systems: Volume 1, Overview,” National Institute of Standards and Technology, NIST SP 1500-201, 2017. doi: 10.6028/NIST.SP.1500-201.
- [11] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, “Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020, doi: 10.1109/COMST.2020.3009103.
- [12] M. Satyanarayanan, “The Emergence of Edge Computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017, doi: 10.1109/MC.2017.9.

- [13] K. Kang, C. Yu, and R. Y. Zhong, “13 - Cloud, fog, edge computing and 5G technologies for industrial automation,” in *Manufacturing from Industry 4.0 to Industry 5.0*, D. Mourtzis, Ed., Elsevier, 2024, pp. 379–430. doi: 10.1016/B978-0-443-13924-6.00013-2.
- [14] International Organization for Standardization, “Automation Systems and Integration — Key Performance Indicators (KPIs) for Manufacturing Operations Management — Part 2: Definitions and Descriptions (ISO 22400-2).” ISO, Geneva, Switzerland, 2014. Accessed: Jul. 19, 2026. [Online]. Available: <https://cdn.standards.iteh.ai/samples/54497/10e33d2a34c144558b5e3024ee5e1cb0/ISO-22400-2-2014.pdf>
- [15] L. Zhu, C. Johnsson, M. Varisco, and M. M. Schiraldi, “Key performance indicators for manufacturing operations management – gap analysis between process industrial needs and ISO 22400 standard,” *Procedia Manufacturing*, vol. 25, pp. 82–88, Jan. 2018, doi: 10.1016/j.promfg.2018.06.060.
- [16] International Organization for Standardization, “Energy Management Systems — Requirements with Guidance for Use (ISO 50001).” ISO, Geneva, Switzerland, 2018.
- [17] S. Nahavandi, “Industry 5.0—A Human-Centric Solution,” *Sustainability*, vol. 11, no. 16, 2019, doi: 10.3390/su11164371.
- [18] P. P. Groumpos, “A Critical Historical and Scientific Overview of all Industrial Revolutions,” *IFAC-PapersOnLine*, vol. 54, no. 13, pp. 464–471, Jan. 2021, doi: 10.1016/j.ifacol.2021.10.492.
- [19] M. Peruzzini, E. Prati, and M. Pellicciari, “A framework to design smart manufacturing systems for Industry 5.0 based on the human-automation symbiosis,” *International Journal of Computer Integrated Manufacturing*, vol. 37, no. 10–11, pp. 1426–1443, Nov. 2024, doi: 10.1080/0951192X.2023.2257634.
- [20] European Commission, “Industry 5.0: Towards a Sustainable, Human-Centric and Resilient European Industry,” Directorate-General for Research and Innovation, 2021. doi: 10.2777/308407.
- [21] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi, “Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” *Mechatronics*, vol. 55, pp. 248–266, 2018, doi: 10.1016/j.mechatronics.2018.02.009.
- [22] International Organization for Standardization, “Robots and robotic devices — Collaborative robots (ISO/TS 15066).” ISO, Geneva, Switzerland, 2016. Accessed: Jul. 19, 2026. [Online]. Available: https://www.diag.uniroma1.it/deluca/pHRI_elective/ISO_TS_15066_2016_en.pdf
- [23] D. Hartmann, K. Hamříková, A. Vysocký, V. Laciok, and A. Bernatík, “Evolution of safety requirements in industrial robotics: Comparative analysis of ISO 10218-1/2 (2011 vs. 2025) and integration of ISO/TS 15066,” *Results in Engineering*, vol. 30, p. 110486, Jun. 2026, doi: 10.1016/j.rineng.2026.110486.
- [24] S. Matzka, “Explainable Artificial Intelligence for Predictive Maintenance Applications,” in *2020 Third International Conference on Artificial Intelligence for Industries*, 2020, pp. 15–20. doi: 10.1109/AI4I49448.2020.00023.
- [25] M. A. Rahman, M. F. Shahrior, K. Iqbal, and A. A. Abushaiba, “Enabling Intelligent Industrial Automation: A Review of Machine Learning Applications with Digital Twin and Edge AI Integration,” *Automation*, vol. 6, no. 3, p. 37, Aug. 2025, doi: 10.3390/automation6030037.

- [26] T. Stock, M. Obenaus, S. Kunz, and H. Kohl, "Industry 4.0 as Enabler for a Sustainable Development: A Qualitative Assessment of Its Ecological and Social Potential," *Process Safety and Environmental Protection*, vol. 118, pp. 254–267, 2018, doi: 10.1016/j.psep.2018.06.026.
- [27] A. Giret, D. Trentesaux, and V. Prabhu, "Sustainability in Manufacturing Operations Scheduling: A State of the Art Review," *Journal of Manufacturing Systems*, vol. 37, pp. 126–140, 2015, doi: 10.1016/j.jmsy.2015.08.002.
- [28] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004, doi: 10.1109/TDSC.2004.2.
- [29] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann, 2007.
- [30] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Berlin, Heidelberg: Springer, 2009. doi: 10.1007/978-3-540-68899-0.
- [31] Edward M. Marszal and Jim McGlone, "Overview of the ISA/IEC 62443 Series," in *Security PHA Review for Consequence-Based Cybersecurity*, John Wiley & Sons, Ltd, 2026, pp. 19–24. doi: 10.1002/9781394442447.ch2.
- [32] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST AI 100–1, Jan. 2023. doi: 10.6028/NIST.AI.100-1.
- [33] M. G. Hudedmani, R. M. Umayal, S. K. Kabberalli, and R. Hittalamani, "Programmable Logic Controller (PLC) in Automation," *Advanced Journal of Graduate Research*, vol. 2, no. 1, pp. 37–45, 2017, doi: 10.21467/ajgr.2.1.37-45.
- [34] International Electrotechnical Commission, "Programmable Controllers — Part 3: Programming Languages (IEC 61131-3)," no. IEC 61131-3:2025. IEC, Geneva, Switzerland, 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://webstore.iec.ch/en/publication/68533>
- [35] OPC Foundation, "OPC UA FX: Extending OPC UA to the field level," OPC Foundation, 2023. Accessed: Jul. 19, 2026. [Online]. Available: <https://opcfoundation.org/wp-content/uploads/2023/11/OPCF-FLC-Technical-Paper-C2C-EN.pdf>
- [36] X. Li, Z. Fan, S. Wang, A. Qiu, and J. Mao, "A Distributed Fault Diagnosis and Cooperative Fault-Tolerant Control Design Framework for Distributed Interconnected Systems," *Sensors*, vol. 22, no. 7, p. 2480, 2022, doi: 10.3390/s22072480.
- [37] A. M. Kugelman, *Practical Process Control Design with Industrial Applications*. John Wiley & Sons, 2024. doi: 10.1002/9781394246526.
- [38] Swapan Basu, *Plant Intelligent Automation and Digital Transformation*. 2022. Accessed: Jul. 26, 2026. [Online]. Available: <https://shop.elsevier.com/books/plant-intelligent-automation-and-digital-transformation/basu/978-0-323-90246-5>
- [39] Gnana Swathika O.V., Aayush Karthikeyan, K. Karthikeyan, P. Sanjeevikumar, Sajju Karapparambil Thomas, and Amin Babu, "Critical review of SCADA and PLC in smart buildings and energy sector," *Energy Reports*, vol. 11, pp. 4259–4272, 2024, doi: 10.1016/j.egy.2024.07.041.

- [40] S. A. Boyer, *SCADA: Supervisory Control and Data Acquisition*, 4th ed. Research Triangle Park, NC: International Society of Automation, 2010.
- [41] Eric D. Knapp and Joel Thomas Langill, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*, 2nd ed. Syngress, 2015. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.sciencedirect.com/book/monograph/9780124201149/industrial-network-security>
- [42] G. Yadav and K. Paul, “Architecture and Security of SCADA Systems: A Review,” 2020, Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2001.02925>
- [43] T. J. Williams, “The Purdue Enterprise Reference Architecture,” *Computers in Industry*, vol. 24, no. 2–3, pp. 141–158, 1994, doi: 10.1016/0166-3615(94)90017-5.
- [44] D. A. Wollman, M. A. Weiss, Y.-S. Li-Baboud, E. R. Griffor, and M. J. Burns, “Framework for Cyber-Physical Systems: Volume 3, Timing Annex,” National Institute of Standards and Technology, NIST Special Publication 1500–203, 2017. doi: 10.6028/NIST.SP.1500-203.
- [45] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003, doi: 10.1109/TNET.2002.808417.
- [46] P. Levis *et al.*, “TinyOS: An Operating System for Sensor Networks,” in *Ambient Intelligence*, W. Weber, J. M. Rabaey, and E. Aarts, Eds., Berlin, Heidelberg: Springer, 2005, pp. 115–148. doi: 10.1007/3-540-27139-2_7.
- [47] FreeRTOS, “RTOS Basics.” Accessed: Jul. 19, 2026. [Online]. Available: <https://www.freertos.org/fr-content-src/RTOS-book.html>
- [48] Zephyr, “Kernel Services — Zephyr Project Documentation.” Accessed: Jul. 19, 2026. [Online]. Available: <https://docs.zephyrproject.org/latest/kernel/services/index.html?>
- [49] Wind River, “VxWorks RTOS for Aerospace and Defense.” 2021. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.windriver.com/>
- [50] F. Sygulla and others, “An EtherCAT-Based Real-Time Control System Architecture for Humanoid Robots,” in *2018 IEEE International Conference on Automation Science and Engineering*, 2018, pp. 483–490. doi: 10.1109/COASE.2018.8560532.
- [51] The Linux Kernel Organization, “Real-time preemption — The Linux Kernel documentation.” Accessed: Jul. 19, 2026. [Online]. Available: <https://www.kernel.org/doc/html/latest/core-api/real-time/index.html?>
- [52] Real-Time Linux Project, “Getting Started with PREEMPT_RT Guide.” Accessed: Jul. 19, 2026. [Online]. Available: https://realtime-linux.org/getting-started-with-preempt_rt-guide/
- [53] International Electrotechnical Commission, “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems (IEC 61508, Parts 1–7).” IEC, Geneva, Switzerland, 2010.
- [54] QNX, “The QNX Neutrino Microkernel.” Accessed: Jul. 19, 2026. [Online]. Available: https://qnx.com/developers/docs/7.1/com.qnx.doc.neutrino.sys_arch/topic/kernel.html?
- [55] M. O. Farooq and T. Kunz, “Operating Systems for Wireless Sensor Networks: A Survey,” *Sensors*, vol. 11, no. 6, pp. 5900–5930, Jun. 2011, doi: 10.3390/s110605900.

- [56] M. Baniyounis and A. Mesmar, “Diagnosability of Programmable Logic Controllers,” in *2017 14th International Multi-Conference on Systems, Signals & Devices*, 2017, pp. 319–324.
- [57] S. M. Khalil, H. Sandberg, and R. H. Deng, “Threat Modeling of Cyber-Physical Systems: A Case Study of a Microgrid System,” *Computers & Security*, vol. 124, p. 102950, 2023, doi: 10.1016/j.cose.2022.102950.
- [58] M. M. Abdelhameed and H. Darabi, “Diagnosis and Debugging of Programmable Logic Controller Control Programs by Neural Networks,” in *Proceedings of the 2005 IEEE International Conference on Automation Science and Engineering*, 2005, pp. 318–323.
- [59] J. Liu, Y. Sha, W. Zhang, Y. Yan, and X. Liu, “Anomaly Detection Method for Industrial Control System Operation Data Based on Time–Frequency Fusion Feature Attention Encoding,” *Sensors*, vol. 24, no. 18, p. 6131, 2024, doi: 10.3390/s24186131.
- [60] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, in MCC ’12. New York, NY, USA: Association for Computing Machinery, Aug. 2012, pp. 13–16. doi: 10.1145/2342509.2342513.
- [61] Sundas Iftikhar et al., “AI-based fog and edge computing: A systematic review, taxonomy and future directions,” *arXiv*, 2022, Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2212.04645>
- [62] Philip Church, Harald Mueller, Caspar Ryan, Spyridon V. Gogouvitis, Andrzej Goscinski, and Zahir Tari, “Migration of a SCADA system to IaaS clouds—A case study,” *Journal of Cloud Computing*, vol. 6, no. 1, p. 12, 2017, doi: 10.1186/s13677-017-0080-5.
- [63] A. Sajid, H. Abbas, and K. Saleem, “Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges,” *IEEE Access*, vol. 4, pp. 1375–1384, 2016, doi: 10.1109/ACCESS.2016.2549047.
- [64] Margo Initiative, “Margo Initiative Accelerates Interoperable Edge Application Management with Preview Release 1.” Linux Foundation, Jan. 2026. Accessed: Jul. 24, 2026. [Online]. Available: <https://margo.org/announcements/2026/01/13/margo-initiative-accelerates-interoperable-edge-application-management-with-preview-release-1/>
- [65] OPC Foundation, “OPC Unified Architecture — Part 1: Overview and Concepts.” OPC Foundation, Oct. 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://reference.opcfoundation.org/specs/OPC-10000-1>
- [66] OASIS, “MQTT Version 5.0 | OASIS Standard.” Accessed: Jul. 19, 2026. [Online]. Available: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html?>
- [67] OPC Foundation, “OPC Unified Architecture — Part 14: PubSub.” OPC Foundation, Oct. 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://reference.opcfoundation.org/specs/OPC-10000-14>
- [68] ASHRAE, “BACnet—A Data Communication Protocol for Building Automation and Control Networks.” ASHRAE, 2024.
- [69] M. Ng, “Smart Application Using MQTT Protocol for Industrial IoT and Retail,” *Journal of Science & Technology*, vol. 5, no. 1, pp. 38–53, 2024.

- [70] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force, Request for Comments RFC 7252, Jun. 2014. doi: 10.17487/RFC7252.
- [71] G. Selander, J. P. Mattsson, F. Palombini, and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)," Internet Engineering Task Force, Request for Comments RFC 8613, Jul. 2019. doi: 10.17487/RFC8613.
- [72] Z. Shelby, A. P. Castellani, and C. Bormann, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 02, pp. 62–67, Mar. 2012, doi: 10.1109/MIC.2012.29.
- [73] Open Mobile Alliance, "Lightweight Machine to Machine Technical Specification: Core," no. OMA-TS-LightweightM2M_Core-V1_2_2-20240613-A. Open Mobile Alliance, Jun. 2024. Accessed: Jul. 19, 2026. [Online]. Available: https://www.openmobilealliance.org/release/LightweightM2M/V1_2_2-20240613-A/HTML-Version/OMA-TS-LightweightM2M_Core-V1_2_2-20240613-A.html
- [74] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002, doi: 10.1016/S1389-1286(01)00302-4.
- [75] V. Bonilla, B. Campoverde, and S. G. Yoo, "A Systematic Literature Review of LoRaWAN: Sensors and Applications," *Sensors*, vol. 23, no. 20, p. 8440, Oct. 2023, doi: 10.3390/s23208440.
- [76] LoRa Alliance, "LoRaWAN L2 1.0.4 Specification," no. TS001-1.0.4. LoRa Alliance, Oct. 2020. Accessed: Jul. 19, 2026. [Online]. Available: https://lora-alliance.org/resource_hub/ts001-1-0-4-lorawan-l2-1-0-4-specification/
- [77] K. Banti, I. Karampelias, T. Dimakis, A.-A. A. Boulogeorgos, T. Kyriakidis, and M. Louta, "LoRaWAN Communication Protocols: A Comprehensive Survey under an Energy Efficiency Perspective," *Telecom*, vol. 3, no. 2, pp. 322–357, 2022, doi: 10.3390/telecom3020018.
- [78] B. S. Chaudhari, M. Zennaro, and S. Borkar, "LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations," *Future Internet*, vol. 12, no. 3, p. 46, 2020, doi: 10.3390/fi12030046.
- [79] Connectivity Standards Alliance, "Zigbee Core Specification," no. Zigbee Core R23.2. Connectivity Standards Alliance. Accessed: Jul. 19, 2026. [Online]. Available: <https://csa-iot.org/developer-resource/specifications-download-request/>
- [80] Bluetooth SIG, "Bluetooth Core Specification 6.3." Bluetooth Special Interest Group, May 2026. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.bluetooth.com/specifications/specs/core-specification-6-3/>
- [81] 3rd Generation Partnership Project, "System Architecture for the 5G System (5GS)." 3rd Generation Partnership Project, Dec. 2024. Accessed: Jul. 19, 2026. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.501/23501-i80.zip
- [82] R. Bostani, E. R. Koleibi, G. Gagnon-Turcotte, R. Fontaine, B. Sharmistha, and B. Gosselin, "Readout Techniques and Offset Compensation Strategies for Biomedical Resistive MEMS Sensors: A Comprehensive Review," *IEEE Rev Biomed Eng*, vol. PP, Dec. 2025, doi: 10.1109/RBME.2025.3639404.

- [83] W. Choi *et al.*, “Recent Advances in Energy-Efficient and Temperature-Resilient Sensor Interfaces,” *IEEE Open Journal of the Solid-State Circuits Society*, vol. 5, pp. 456–469, 2025, doi: 10.1109/OJSSCS.2025.3634970.
- [84] S. Lakshminarayana, H. Park, and S. Jung, “Emerging Technologies in CMOS Integrated Sensing System-On-Chip: A Review,” *IEEE Sensors Reviews*, vol. 2, pp. 397–404, 2025, doi: 10.1109/SR.2025.3592654.
- [85] N. Almarri, J. Chang, W. Song, D. Jiang, and A. Demosthenous, “Piezoelectric energy harvesting and ultra-low-power management circuits for medical devices,” *Nano Energy*, vol. 131, p. 110196, Dec. 2024, doi: 10.1016/j.nanoen.2024.110196.
- [86] K. Zhang, S. Liu, S. An, Y. Zhou, and X. Pu, “Towards high-efficiency self-powered wireless sensor networks: A systematic review of co-design of energy and signal,” *Nano Energy*, vol. 144, p. 111337, Nov. 2025, doi: 10.1016/j.nanoen.2025.111337.
- [87] A. Nasrallah *et al.*, “Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019, doi: 10.1109/COMST.2018.2869350.
- [88] EtherCAT Technology Group, “EtherCAT Compendium.” EtherCAT Technology Group, May 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.ethercat.org/en/compendium.htm>
- [89] EtherCAT Technology Group, “Safety over EtherCAT (FSoE) Specification.” EtherCAT Technology Group. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.ethercat.org/en/safety.html>
- [90] International Electrotechnical Commission and Institute of Electrical and Electronics Engineers, “Time-Sensitive Networking Profile for Industrial Automation (IEC/IEEE 60802).” IEC/IEEE, Geneva, Switzerland, 2026.
- [91] IEEE, “IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications.” Institute of Electrical and Electronics Engineers, 2020. doi: 10.1109/IEEESTD.2020.9121845.
- [92] IEEE, “IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic.” Institute of Electrical and Electronics Engineers, 2016. doi: 10.1109/IEEESTD.2016.7572855.
- [93] IEEE, “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.” Institute of Electrical and Electronics Engineers, Jun. 2020. doi: 10.1109/IEEESTD.2020.9120376.
- [94] IEEE, “IEEE Standard for Local and Metropolitan Area Networks—Frame Replication and Elimination for Reliability.” Institute of Electrical and Electronics Engineers, 2017. doi: 10.1109/IEEESTD.2017.8091139.
- [95] J.-D. Decotignie, “Ethernet-Based Real-Time and Industrial Communications,” *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005, doi: 10.1109/JPROC.2005.849721.
- [96] International Electrotechnical Commission, “Programmable controllers - Part 9: Single-drop digital communication interface for small sensors and actuators (SDCI / IO-Link),” no. IEC 61131-9:2022. IEC, Geneva, Switzerland, 2022. Accessed: Jul. 19, 2026. [Online]. Available: <https://webstore.iec.ch/en/publication/68534>

- [97] M. Felser, “Real-Time Ethernet - Industry Prospective,” *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1118–1129, 2005, doi: 10.1109/JPROC.2005.849720.
- [98] Ignacio Rodriguez et al., “An Experimental Framework for 5G Wireless System Integration into Industry 4.0 Applications,” *Energies*, vol. 14, no. 15, p. 4444, 2021, doi: 10.3390/en14154444.
- [99] Modbus Organization, “MODBUS Application Protocol Specification.” Modbus Organization, Apr. 2012. Accessed: Jul. 19, 2026. [Online]. Available: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
- [100] Modbus Organization, “MODBUS over Serial Line Specification and Implementation Guide.” Modbus Organization, Dec. 2006. Accessed: Jul. 19, 2026. [Online]. Available: https://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf
- [101] Modbus Organization, “MODBUS Messaging on TCP/IP Implementation Guide.” Modbus Organization, Oct. 2006. Accessed: Jul. 19, 2026. [Online]. Available: https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf
- [102] Mo Chen, Zhe Wei, Li Li, and Kai Zhang, “Edge computing-based proactive control method for industrial product manufacturing quality prediction,” *Scientific Reports*, vol. 14, no. 1, p. 1288, 2024, doi: 10.1038/s41598-024-51974-z.
- [103] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA: O’Reilly Media, 2019. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.oreilly.com/library/view/tinyml/9781492052036/>
- [104] P. Kairouz *et al.*, “Advances and Open Problems in Federated Learning,” *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021, doi: 10.1561/22000000083.
- [105] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, PMLR, Apr. 2017, pp. 1273–1282. Accessed: Jul. 19, 2026. [Online]. Available: <https://proceedings.mlr.press/v54/mcmahan17a.html>
- [106] F. Cappa, R. Oriani, E. Peruffo, and I. McCarthy, “Big Data for Creating and Capturing Value in the Digitalized Environment,” *Journal of Product Innovation Management*, vol. 38, no. 1, pp. 49–67, 2021, doi: 10.1111/jpim.12545.
- [107] International Organization for Standardization and International Electrotechnical Commission, “Software Engineering—Software Product Quality Requirements and Evaluation (SQuaRE)—Data Quality Model (ISO/IEC 25012).” ISO/IEC, Geneva, Switzerland, Dec. 2008. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/35736.html>
- [108] W. L. Chang, D. Boyd, and O. Levin, “NIST Big Data Interoperability Framework: Volume 6, Reference Architecture,” National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST Special Publication 1500-6r2, Oct. 2019. doi: 10.6028/NIST.SP.1500-6r2.
- [109] R. Kimball and J. Caserta, *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*, 1st ed. Indianapolis, IN, USA: John Wiley & Sons, 2004.
- [110] Databricks, “ETL vs. ELT: Two Data Processing Approaches.” Accessed: Jul. 19, 2026. [Online]. Available: <https://www.databricks.com/discover/etl/vs-elt>

- [111] T. Akidau *et al.*, “The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing,” *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1792–1803, Aug. 2015, doi: 10.14778/2824032.2824076.
- [112] Google, “Protocol Buffers Language Guide.” Google, 2026. Accessed: Jul. 19, 2026. [Online]. Available: <https://protobuf.dev/programming-guides/>
- [113] C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR),” Internet Engineering Task Force, Request for Comments RFC 8949, Dec. 2020. doi: 10.17487/RFC8949.
- [114] Apache Software Foundation, “Apache Avro 1.12.0 Specification.” Apache Software Foundation, 2026. Accessed: Jul. 19, 2026. [Online]. Available: <https://avro.apache.org/docs/1.12.0/specification/>
- [115] A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.
- [116] I.-M. Rus and others, “Extracting Appliance Behavior from Heterogeneous Data,” in *2023 IEEE International Conference on Intelligent Computer Communication and Processing*, 2023, pp. 87–92. doi: 10.1109/ICCP60212.2023.10398620.
- [117] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: John Wiley & Sons, 2015.
- [118] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1999.
- [119] A. K. S. Jardine, D. Lin, and D. Banjevic, “A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance,” *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006, doi: 10.1016/j.ymssp.2005.09.012.
- [120] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018, doi: 10.1016/j.ymssp.2017.11.016.
- [121] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009, doi: 10.1145/1541880.1541882.
- [122] S. Katipamula and M. R. Brambley, “Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems—A Review,” *HVAC&R Research*, vol. 11, no. 1, pp. 3–25, 2005, doi: 10.1080/10789669.2005.10391123.
- [123] Tiago Zonta *et al.*, “Predictive Maintenance in the Industry 4.0: A Systematic Literature Review,” *Computers & Industrial Engineering*, vol. 150, p. 106889, 2020, doi: 10.1016/j.cie.2020.106889.
- [124] H. He and E. A. Garcia, “Learning from Imbalanced Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.
- [125] International Organization for Standardization, “Condition Monitoring and Diagnostics of Machine Systems—Prognostics—Part 1: General Guidelines and Requirements (ISO 13381-1).” ISO, Geneva, Switzerland, Sep. 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/88029.html>

- [126] J. P. Klein and M. L. Moeschberger, *Survival Analysis: Techniques for Censored and Truncated Data*, 2nd ed. New York, NY, USA: Springer, 2003. doi: 10.1007/b97377.
- [127] A. Saxena *et al.*, “Metrics for Evaluating Performance of Prognostic Techniques,” in *2008 International Conference on Prognostics and Health Management*, IEEE, 2008, pp. 1–17. doi: 10.1109/PHM.2008.4711436.
- [128] A. Rajkomar, J. Dean, and I. Kohane, “Machine Learning in Medicine,” *The New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, 2019, doi: 10.1056/NEJMra1814259.
- [129] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014, doi: 10.1145/2523813.
- [130] C. R. Farrar and K. Worden, *Structural Health Monitoring: A Machine Learning Perspective*. Chichester, UK: John Wiley & Sons, 2012. doi: 10.1002/9781118443118.
- [131] S. Kumar and H. Snooks, “Remote Patient Monitoring: Technology, Benefits and Challenges,” *Journal of Medical Systems*, vol. 44, no. 9, p. 159, 2020, doi: 10.1007/s10916-020-01621-4.
- [132] A. Pantelopoulos and N. G. Bourbakis, “A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 1, pp. 1–12, 2010, doi: 10.1109/TSMCC.2009.2032660.
- [133] A. Vegesna, M. Tran, M. Angelaccio, and S. Arcona, “Remote Patient Monitoring via Non-Invasive Digital Technologies: A Systematic Review,” *Telemedicine and e-Health*, vol. 23, no. 1, pp. 3–17, 2017, doi: 10.1089/tmj.2016.0051.
- [134] Health Level Seven International, “FHIR Release 5.” Health Level Seven International, Mar. 2023. Accessed: Jul. 19, 2026. [Online]. Available: <https://hl7.org/fhir/R5/>
- [135] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Springer, 2022.
- [136] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems*, 2015.
- [137] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
- [138] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.
- [139] A. Dosovitskiy *et al.*, “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale,” in *International Conference on Learning Representations*, 2021. Accessed: Jul. 19, 2026. [Online]. Available: <https://openreview.net/forum?id=YicbFdNTTy>
- [140] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, “Towards Total Recall in Industrial Anomaly Detection,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14298–14308. doi: 10.1109/CVPR52688.2022.01392.
- [141] V. Zavrtanik, M. Kristan, and D. Skočaj, “DRAEM—A Discriminatively Trained Reconstruction Embedding for Surface Anomaly Detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8330–8339. doi: 10.1109/ICCV48922.2021.00822.

- [142] Taichi Fukuda, Kotaro Hasegawa, Shinya Ishizaki, Shohei Nobuhara, and Ko Nishino, “BlindSpotNet: Seeing where we cannot see.” 2022. Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2207.03870>
- [143] J. Jeong, Y. Zou, T. Kim, D. Zhang, A. Ravichandran, and O. Dabeer, “WinCLIP: Zero-/Few-Shot Anomaly Classification and Segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19606–19616. doi: 10.1109/CVPR52729.2023.01878.
- [144] Yuqi Cheng *et al.*, “A comprehensive survey for real-world industrial defect detection: Challenges, approaches, and prospects.” 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2507.13378>
- [145] P. Bergmann, X. Jin, D. Sattlegger, and C. Steger, “The MVTec 3D-AD Dataset for Unsupervised 3D Anomaly Detection and Localization,” in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Volume 5: VISAPP*, 2022, pp. 202–213. doi: 10.5220/0010865000003124.
- [146] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD — A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9592–9600. doi: 10.1109/CVPR.2019.00982.
- [147] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, “SPot-the-Difference Self-Supervised Pre-training for Anomaly Detection and Segmentation,” in *Computer Vision – ECCV 2022*, in Lecture Notes in Computer Science. Springer, 2022, pp. 392–408. doi: 10.1007/978-3-031-19821-2_23.
- [148] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.
- [149] T. Bohlin, *Practical Grey-Box Process Identification: Theory and Applications*. London, UK: Springer, 2006. doi: 10.1007/1-84628-403-0.
- [150] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 6th ed. Hoboken, NJ, USA: John Wiley & Sons, 2021.
- [151] F. E. Cellier and E. Kofman, *Continuous System Simulation*. New York, NY, USA: Springer, 2006. doi: 10.1007/0-387-30260-3.
- [152] S. J. Qin and T. A. Badgwell, “A Survey of Industrial Model Predictive Control Technology,” *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003, doi: 10.1016/S0967-0661(02)00186-7.
- [153] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, “Constrained Model Predictive Control: Stability and Optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, doi: 10.1016/S0005-1098(99)00214-9.
- [154] A. Afram and F. Janabi-Sharifi, “Theory and Applications of HVAC Control Systems—A Review of Model Predictive Control (MPC),” *Building and Environment*, vol. 72, pp. 343–355, 2014, doi: 10.1016/j.buildenv.2013.11.016.
- [155] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-Based Model Predictive Control: Toward Safe Learning in Control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020, doi: 10.1146/annurev-control-090419-075625.

- [156] M. Grieves and J. Vickers, “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems,” *Transdisciplinary Perspectives on Complex Systems*, pp. 85–113, 2017, doi: 10.1007/978-3-319-38756-7_4.
- [157] International Organization for Standardization, “Automation systems and integration — Digital twin framework for manufacturing — Part 1: Overview and general principles (ISO 23247-1).” ISO, Geneva, Switzerland, 2021. Accessed: Jul. 19, 2026. [Online]. Available: https://www.iso.org/obp/ui/en/?__cf_chl_f_tk=dRixhowTMiSTOK1C5ErKIM1d2lg8Rw_ByZrXUROouuk-1783290429-1.0.1.1-IZjIbAZE1SpzFz.YKZbLRiA6Iu2r0vLYBJ5bOumToMY#iso:std:iso:23247:-1:ed-1:v1:en
- [158] Jia Hu, Yongwei Feng, Shuoyuan Li, Haoran Wang, Jaehyun So, and Junnian Zheng, “Mirroring the parking target: An optimal-control-based parking motion planner with strengthened parking reliability and faster parking completion.” 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2405.07538>
- [159] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, “Digital Twin in Manufacturing: A Categorical Literature Review and Classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, doi: 10.1016/j.ifacol.2018.08.474.
- [160] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui, “Digital twin-driven product design, manufacturing and service with big data,” *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9–12, pp. 3563–3576, 2018, doi: 10.1007/s00170-017-0233-1.
- [161] E. Negri, L. Fumagalli, and M. Macchi, “A Review of the Roles of Digital Twin in CPS-Based Production Systems,” *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017, doi: 10.1016/j.promfg.2017.07.198.
- [162] A. Humayed, J. Lin, F. Li, and B. Luo, “Cyber-Physical Systems Security—A Survey,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017, doi: 10.1109/JIOT.2017.2703172.
- [163] International Organization for Standardization, “Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design,” no. ISO 13849-1:2023. ISO, Geneva, Switzerland, 2023. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/obp/ui/fr/#iso:std:iso:13849:-1:ed-4:v1:en>
- [164] European Parliament and Council of the European Union, “Regulation (EU) 2023/1230 of the European Parliament and of the Council of 14 June 2023 on machinery and repealing Directive 2006/42/EC of the European Parliament and of the Council and Council Directive 73/36/EEC,” *Official Journal of the European Union*, vol. 165. in L, vol. 165. pp. 1–102, 2023. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2023/1230/oj>
- [165] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*. Reading, MA, USA: Addison-Wesley, 1989.
- [166] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, “A Survey of Cyber Security Management in Industrial Control Systems,” *International Journal of Critical Infrastructure Protection*, vol. 9, pp. 52–80, 2015, doi: 10.1016/j.ijcip.2015.02.002.

- [167] International Electrotechnical Commission, “Industrial Communication Networks—Network and System Security—Part 3-3: System Security Requirements and Security Levels (IEC 62443-3-3).” IEC, Geneva, Switzerland, 2013.
- [168] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero Trust Architecture,” National Institute of Standards and Technology, NIST Special Publication 800–207, Aug. 2020. doi: 10.6028/NIST.SP.800-207.
- [169] A. A. Cárdenas, S. Amin, and S. Sastry, “Secure Control: Towards Survivable Cyber-Physical Systems,” *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops*, pp. 495–500, 2008, doi: 10.1109/ICDCS.Workshops.2008.40.
- [170] International Electrotechnical Commission, “Security for Industrial Automation and Control Systems—Part 2-1: Security Program Requirements for IACS Asset Owners (IEC 62443-2-1).” IEC, Geneva, Switzerland, 2024.
- [171] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, “Computer Security Incident Handling Guide,” National Institute of Standards and Technology, NIST Special Publication 800–61 Revision 2, Aug. 2012. doi: 10.6028/NIST.SP.800-61r2.
- [172] European Parliament and Council of the European Union, “Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence,” *Official Journal of the European Union*, vol. L, no. 2024/1689. pp. 1–144, Jun. 13, 2024. Accessed: Aug. 06, 2026. [Online]. Available: <http://data.europa.eu/eli/reg/2024/1689/oj>
- [173] European Parliament and Council of the European Union, “Regulation (EU) 2026/1744 Amending Regulations (EU) 2024/1689, (EU) 2018/1139 and (EU) 2023/1230 as Regards the Simplification of the Implementation of Harmonised Rules on Artificial Intelligence,” *Official Journal of the European Union*, vol. L, no. 2026/1744. Jul. 08, 2026. Accessed: Aug. 06, 2026. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202601744
- [174] European Parliament and Council of the European Union, “Regulation (EU) 2024/2847 on Horizontal Cybersecurity Requirements for Products with Digital Elements and Amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828,” *Official Journal of the European Union*, vol. L, no. 2024/2847. pp. 1–71, Oct. 23, 2024. Accessed: Aug. 06, 2026. [Online]. Available: <http://data.europa.eu/eli/reg/2024/2847/oj>
- [175] M. R. Endsley, “Toward a Theory of Situation Awareness in Dynamic Systems,” *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995, doi: 10.1518/001872095779049543.
- [176] M. R. Endsley and D. G. Jones, *Designing for Situation Awareness: An Approach to User-Centered Design*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2012.
- [177] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, “A Model for Types and Levels of Human Interaction with Automation,” *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 30, no. 3, pp. 286–297, 2000, doi: 10.1109/3468.844354.
- [178] International Society of Automation, “Human Machine Interfaces for Process Automation Systems.” International Society of Automation, 2015.
- [179] B. R. Hollifield, D. Oliver, I. Nimmo, and E. Habibi, *The High Performance HMI Handbook*. Houston, TX, USA: Plant Automation Services, 2008.
- [180] C. D. Wickens, J. D. Lee, Y. Liu, and S. E. Gordon-Becker, *An Introduction to Human Factors Engineering*, 2nd ed. Pearson, 2015.

- [181] International Society of Automation, “Management of Alarm Systems for the Process Industries.” International Society of Automation, 2016.
- [182] International Electrotechnical Commission, “Management of Alarm Systems for the Process Industries (IEC 62682).” IEC, Geneva, Switzerland, 2022.
- [183] I. Lee and Y. J. Shin, “Machine Learning for Enterprises: Applications, Algorithm Selection, and Challenges,” *Business Horizons*, vol. 63, no. 2, pp. 157–170, 2020, doi: 10.1016/j.bushor.2019.10.005.
- [184] International Federation of Robotics (IFR), “World Robotics 2024 – Industrial Robots,” IFR Statistical Department, VDMA Services GmbH, Frankfurt, Germany, 2024. Accessed: Jul. 19, 2026. [Online]. Available: https://ifr.org/img/worldrobotics/Executive_Summary_WR_2024_Industrial_Robots.pdf
- [185] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Springer, 2016. doi: 10.1007/978-3-319-32552-1.
- [186] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, UK: Cambridge University Press, 2017. doi: 10.1017/9781107156302.
- [187] D. Mostaghniyazdi and S. E. Nodehi, “Resistive Sensing in Soft Robotic Grippers: A Comprehensive Review of Strain, Tactile, and Ionic Sensors,” *Electronics*, vol. 14, no. 23, p. 4290, 2025, doi: 10.3390/electronics14234290.
- [188] J. Hernandez et al., “Current Designs of Robotic Arm Grippers: A Comprehensive Systematic Review,” *Robotics*, vol. 12, no. 1, p. 5, 2023, doi: 10.3390/robotics12010005.
- [189] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994.
- [190] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo, *Robotics: Modelling, planning and control*. Springer, 2009. Accessed: Jul. 19, 2026. [Online]. Available: <https://link.springer.com/book/10.1007/978-1-84628-642-1>
- [191] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
- [192] T. Bhuiyan and others, “Deep-Reinforcement-Learning-Based Path Planning for Industrial Robots Using Distance Sensors as Observation.” 2023. Accessed: Jul. 19, 2026. [Online]. Available: <https://arxiv.org/abs/2301.05980>
- [193] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement Learning in Robotics: A Survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013, doi: 10.1177/0278364913495721.
- [194] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World,” *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 23–30, 2017, doi: 10.1109/IROS.2017.8202133.
- [195] S. Gupta, U. Mamodiya, and A. J. A. Al-Gburi, “Speech Recognition-Based Wireless Control System for Mobile Robotics: Design, Implementation, and Analysis,” *Automation*, vol. 6, no. 3, p. 25, Sep. 2025, doi: 10.3390/automation6030025.

- [196] International Organization for Standardization, “Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots (ISO 10218-1).” ISO, Geneva, Switzerland, 2011. Accessed: Jul. 19, 2026. [Online]. Available: <https://cdn.standards.iteh.ai/samples/51330/c008a0a974584a5098400991b63eaae9/ISO-10218-1-2011.pdf>
- [197] International Organization for Standardization, “Robotics—Safety Requirements—Part 1: Industrial Robots (ISO 10218-1).” ISO, Geneva, Switzerland, 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/73933.html>
- [198] International Organization for Standardization, “Robotics—Safety Requirements—Part 2: Industrial Robot Applications and Robot Cells (ISO 10218-2).” ISO, Geneva, Switzerland, 2025.
- [199] R. R. Murphy, B. M. Gandudi, and J. Adams, “Applications of Robots for COVID-19 Response,” *arXiv*, 2020.
- [200] SAE International, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.” SAE International, 2021.
- [201] İ. Avcı and M. Koca, “Intelligent Transportation System Technologies, Challenges and Security,” *Applied Sciences*, vol. 14, no. 11, p. 4646, 2024, doi: 10.3390/app14114646.
- [202] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [203] M. Zorkany, A. Yasser, and A. I. A. Galal, “Vehicle-to-Vehicle Communication: Scope, Importance, Challenges, Research Directions and Future,” *The Open Transportation Journal*, vol. 14, pp. 86–98, 2020, doi: 10.2174/1874447802014010086.
- [204] Q. Chen, S. Tang, Q. Yang, and S. Fu, “Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds,” *2019 IEEE 39th International Conference on Distributed Computing Systems*, pp. 514–524, 2019, doi: 10.1109/ICDCS.2019.00058.
- [205] Mubarak Badamasi Aremu, Gamil Ahmed, Sami Elferik, and Abdul-Wahid A. Saif, “Autonomous mobile robot path planning techniques—A review: Metaheuristic and cognitive techniques,” *Robotics*, vol. 15, no. 1, p. 23, 2026, doi: 10.3390/robotics15010023.
- [206] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, *A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles*. IEEE, 2016. doi: 10.1109/TIV.2016.2578706.
- [207] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [208] Mingyang Zhu and Gangfeng Tan, “Research on cooperative adaptive cruise control (CACC) based on fuzzy PID algorithm,” *SAE Technical Paper*, no. 2023-01-0682, pp. 2023-01-0682, 2023, doi: 10.4271/2023-01-0682.
- [209] K. Abboud, H. A. Omar, and W. Zhuang, “Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9457–9470, 2016, doi: 10.1109/TVT.2016.2591558.
- [210] J. B. Kenney, “Dedicated Short-Range Communications (DSRC) Standards in the United States,” *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011, doi: 10.1109/JPROC.2011.2132790.

- [211] M. Papageorgiou, *Applications of Automatic Control Concepts to Traffic Flow Modeling and Control*. Berlin, Germany: Springer, 2003.
- [212] International Organization for Standardization, “Industrial Trucks—Safety Requirements and Verification—Part 4: Driverless Industrial Trucks and Their Systems (ISO 3691-4).” ISO, Geneva, Switzerland, Jun. 2023. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/83545.html>
- [213] International Society of Automation, “Enterprise-Control System Integration—Part 1: Models and Terminology (NSI/ISA-95.00.01-2025 / IEC 62264-1 Modified).” ISA, 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.isa.org/standards-and-publications/isa-standards/isa-95-standard>
- [214] E. Monk and B. Wagner, *Concepts in Enterprise Resource Planning*, 4th ed. Boston, MA, USA: Cengage Learning, 2013.
- [215] T. H. Davenport, “Putting the Enterprise into the Enterprise System,” *Harvard Business Review*, vol. 76, no. 4, pp. 121–131, 1998.
- [216] J. Kletti, Ed., *Manufacturing Execution Systems—MES*. Berlin, Germany: Springer, 2007. doi: 10.1007/978-3-540-49744-8.
- [217] B. Saenz de Ugarte, A. Artiba, and R. Pellerin, “Manufacturing Execution System—A Literature Review,” *Production Planning & Control*, vol. 20, no. 6, pp. 525–539, 2009, doi: 10.1080/09537280902938613.
- [218] World Wide Web Consortium, “PROV-O: The PROV Ontology.” World Wide Web Consortium, Apr. 2013. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.w3.org/TR/prov-o/>
- [219] S. Nakajima, *Introduction to TPM: Total Productive Maintenance*. Cambridge, MA, USA: Productivity Press, 1988.
- [220] D. J. Power, *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books, 2002.
- [221] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.
- [222] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007, doi: 10.1109/TEVC.2007.892759.
- [223] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the Strength Pareto Evolutionary Algorithm,” Computer Engineering and Networks Laboratory, ETH Zurich, Zurich, Switzerland, TIK Report 103, May 2001. doi: 10.3929/ethz-a-004284029.
- [224] M. Emmerich, N. Beume, and B. Naujoks, “An EMO Algorithm Using the Hypervolume Measure as Selection Criterion,” in *Evolutionary Multi-Criterion Optimization*, in Lecture Notes in Computer Science, vol. 3410. Springer, 2005, pp. 62–76. doi: 10.1007/978-3-540-31880-4_5.
- [225] A. Rahman, A. Hussain, S. Nazir, S. Khan, and H. U. Khan, “Intelligent Decision Support Systems — An Analysis of Machine Learning and Multicriteria Decision-Making Methods,” *Applied Sciences*, vol. 13, no. 22, p. 12426, 2023, doi: 10.3390/app132212426.

- [226] Juan Moreno-Malo et al., “Improving traffic light systems using Deep Q-networks,” *Expert Systems with Applications*, vol. 252, no. B, p. 124178, 2024, doi: 10.1016/j.eswa.2024.124178.
- [227] M. L. Markus, S. Axline, D. Petrie, and C. Tanis, “Learning from Adopters’ Experiences with ERP: Problems Encountered and Success Achieved,” *Journal of Information Technology*, vol. 15, no. 4, pp. 245–265, 2000, doi: 10.1080/02683960010008944.
- [228] S. Shang and P. B. Seddon, “Assessing and Managing the Benefits of Enterprise Systems: The Business Manager’s Perspective,” *Information Systems Journal*, vol. 12, no. 4, pp. 271–299, 2002, doi: 10.1046/j.1365-2575.2002.00132.x.
- [229] N. Melville, K. Kraemer, and V. Gurbaxani, “Information Technology and Organizational Performance: An Integrative Model of IT Business Value,” *MIS Quarterly*, vol. 28, no. 2, pp. 283–322, 2004, doi: 10.2307/25148636.
- [230] A. Schumacher, S. Erol, and W. Sihn, “A Maturity Model for Assessing Industry 4.0 Readiness and Maturity of Manufacturing Enterprises,” *Procedia CIRP*, vol. 52, pp. 161–166, 2016, doi: 10.1016/j.procir.2016.07.040.
- [231] G. Schuh, R. Anderl, R. Dumitrescu, A. Krüger, and M. ten Hompel, *Industrie 4.0 Maturity Index: Managing the Digital Transformation of Companies*, Update 2020. Munich, Germany: acatech – National Academy of Science and Engineering, 2020.
- [232] International Organization for Standardization, International Electrotechnical Commission, and Institute of Electrical and Electronics Engineers, “Software, Systems and Enterprise—Architecture Description (ISO/IEC/IEEE 42010).” ISO/IEC/IEEE, Geneva, Switzerland, Nov. 2022. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/74393.html>
- [233] Industry IoT Consortium, “The Industrial Internet Reference Architecture,” Industry IoT Consortium, Nov. 2022. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iiconsortium.org/IIRA/>
- [234] International Electrotechnical Commission, “Smart Manufacturing—Reference Architecture Model Industry 4.0 (RAMI4.0) - IEC PAS 63088.” IEC, Geneva, Switzerland, Mar. 2017. Accessed: Jul. 19, 2026. [Online]. Available: <https://webstore.iec.ch/en/publication/30082>
- [235] International Organization for Standardization and International Electrotechnical Commission, “Internet of Things (IoT)—Reference Architecture (ISO/IEC 30141).” ISO/IEC, Geneva, Switzerland, Aug. 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/88800.html>
- [236] A. Ghaffarianhoseini and others, “What Is an Intelligent Building? Analysis of Recent Interpretations from an International Perspective,” *Architectural Science Review*, vol. 59, no. 5, pp. 338–357, 2016, doi: 10.1080/00038628.2015.1079163.
- [237] International Electrotechnical Commission, “Asset Administration Shell for Industrial Applications—Part 1: Asset Administration Shell Structure (IEC 63278-1).” IEC, Geneva, Switzerland, 2023.
- [238] European Commission, “Commission Staff Working Document on Common European Data Spaces,” European Commission, Brussels, SWD(2024) 21 final, Jan. 2024. Accessed: Jul. 24, 2026. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/library/second-staff-working-document-data-spaces>

- [239] A. Nizamis, P. Gkonis, D. Ioannidis, A. Ntafalias, D. Tzovaras, and P. Trakadas, "Manufacturing Data Spaces Applications in Europe: A Survey," *Data in Brief*, vol. 63, p. 112149, 2025, doi: 10.1016/j.dib.2025.112149.
- [240] International Data Spaces Association, "IDS Reference Architecture Model," International Data Spaces Association, Dortmund, Germany, 2022. Accessed: Jul. 24, 2026. [Online]. Available: <https://docs.internationaldataspaces.org/ids-knowledgebase/ids-ram-4>
- [241] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big Data in Smart Farming—A Review," *Agricultural Systems*, vol. 153, pp. 69–80, 2017, doi: 10.1016/j.agsy.2017.01.023.
- [242] N. Zhang, M. Wang, and N. Wang, "Precision Agriculture—A Worldwide Overview," *Computers and Electronics in Agriculture*, vol. 36, no. 2–3, pp. 113–132, 2002, doi: 10.1016/S0168-1699(02)00096-0.
- [243] X. Xu, "From Cloud Computing to Cloud Manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 28, pp. 75–86, 2012, doi: 10.1016/j.rcim.2011.07.002.
- [244] C. Liu, Z. Su, X. Xu, and Y. Lu, "Service-Oriented Industrial Internet of Things Gateway for Cloud Manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102217, 2022, doi: 10.1016/j.rcim.2021.102217.
- [245] D. Wu, M. J. Greer, D. W. Rosen, and D. Schaefer, "Cloud manufacturing: Strategic vision and state-of-the-art," *Journal of Manufacturing Systems*, vol. 32, no. 4, pp. 564–579, Oct. 2013, doi: 10.1016/j.jmsy.2013.04.008.
- [246] K. Bunse, M. Vodicka, P. Schönsleben, M. Brühlhart, and F. O. Ernst, "Integrating Energy Efficiency Performance in Production Management—Gap Analysis between Industrial Needs and Scientific Literature," *Journal of Cleaner Production*, vol. 19, no. 6–7, pp. 667–679, 2011, doi: 10.1016/j.jclepro.2010.11.011.
- [247] A. González-Vidal, V. Moreno-Cano, F. Terroso-Sáenz, and A. F. Skarmeta, "Towards Energy Efficiency Smart Buildings Models Based on Intelligent Data Analytics," *Procedia Computer Science*, vol. 83, pp. 994–999, 2016, doi: 10.1016/j.procs.2016.04.213.
- [248] International Organization for Standardization, "Energy Audits—Part 1: General Requirements with Guidance for Use (ISO 50002-1)." ISO, Geneva, Switzerland, Jun. 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/83645.html>
- [249] International Organization for Standardization, "Energy Management Systems—Evaluating Energy Performance Using Energy Performance Indicators and Energy Baselines (ISO 50006)." ISO, Geneva, Switzerland, 2023.
- [250] International Organization for Standardization, "Energy Management Systems—Measurement and Verification of Energy Performance of Organizations—General Principles and Guidance (ISO 50015)." ISO, Geneva, Switzerland, Dec. 2014. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/60043.html>
- [251] International Organization for Standardization, "Circular Economy—Vocabulary, Principles and Guidance for Implementation (ISO 59004)." ISO, Geneva, Switzerland, 2024.
- [252] International Organization for Standardization, "Circular Economy—Measuring and Assessing Circularity Performance (ISO 59020)." ISO, Geneva, Switzerland, May 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/80650.html>

- [253] International Organization for Standardization, “Environmental Management—Life Cycle Assessment—Principles and Framework (ISO 14040).” ISO, Geneva, Switzerland, Jul. 2006. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/37456.html>
- [254] International Organization for Standardization, “Environmental Management—Life Cycle Assessment—Requirements and Guidelines (ISO 14044).” ISO, Geneva, Switzerland, Jul. 2006. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/38498.html>
- [255] International Electrotechnical Commission, “Function blocks - Part 1: Architecture (IEC 61499-1),” no. IEC 61499-1:2012. IEC, Geneva, Switzerland, 2012. Accessed: Jul. 19, 2026. [Online]. Available: <https://webstore.iec.ch/en/publication/5506>
- [256] V. Vyatkin, “IEC 61499 as an Enabler of Distributed and Intelligent Automation: State-of-the-Art Review,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 768–781, 2011, doi: 10.1109/TII.2011.2166785.
- [257] L. Ren *et al.*, “Industrial Foundation Model,” *IEEE Transactions on Cybernetics*, vol. 55, no. 5, pp. 2286–2301, 2025, doi: 10.1109/TCYB.2025.3527632.
- [258] S. Zhao *et al.*, “Industrial Foundation Models (IFMs) for Intelligent Manufacturing: A Systematic Review,” *Journal of Manufacturing Systems*, vol. 82, pp. 420–448, 2025, doi: 10.1016/j.jmsys.2025.06.011.
- [259] International Organization for Standardization, “Mechanical vibration — Measurement and evaluation of machine vibration — Part 3: Industrial machinery with a power rating above 15 kW and operating speeds between 120 r/min and 30 000 r/min (ISO 20816-3),” ISO. Accessed: Aug. 14, 2026. [Online]. Available: <https://www.iso.org/standard/78311.html>
- [260] International Organization for Standardization, “Building Automation and Control Systems (BACS)—Part 1: Project Specification and Implementation (ISO 16484-1).” ISO, Geneva, Switzerland, Jan. 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/84890.html>
- [261] International Organization for Standardization, “Building Automation and Control Systems (BACS)—Part 2: Hardware (ISO 16484-2).” ISO, Geneva, Switzerland, Jan. 2025. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/86354.html>
- [262] International Organization for Standardization, “Building Automation and Control Systems (BACS)—Part 5: Data Communication Protocol (ISO 16484-5).” ISO, Geneva, Switzerland, 2022. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/80638.html>
- [263] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. New York, NY, USA: Springer, 2012. doi: 10.1007/978-1-4614-1433-9.
- [264] 3rd Generation Partnership Project, “Service Requirements for V2X Services.” 3rd Generation Partnership Project, 2026. Accessed: Jul. 19, 2026. [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2989>
- [265] International Organization for Standardization, “Tractors and Machinery for Agriculture and Forestry—Serial Control and Communications Data Network—Part 1: General Standard for Mobile Data Communication (ISO 11783-1).” ISO, Geneva, Switzerland, Dec. 2017. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/57556.html>

- [266] International Organization for Standardization, “Tractors and Machinery for Agriculture and Forestry—Serial Control and Communications Data Network—Part 3: Application Layer, Transport Layer and Network Layer (ISO 11783-3).” ISO, Geneva, Switzerland, Mar. 2026. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/89949.html>
- [267] International Organization for Standardization, “Agricultural Machinery and Tractors—Safety of Partially Automated, Semi-Autonomous and Autonomous Machinery—Part 2: Design Principles for Obstacle Protection Systems (ISO 18497-2).” ISO, Geneva, Switzerland, Jul. 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/82686.html>
- [268] International Organization for Standardization, “Agricultural Machinery and Tractors—Safety of Partially Automated, Semi-Autonomous and Autonomous Machinery—Part 3: Autonomous Operating Zones (ISO 18497-3).” ISO, Geneva, Switzerland, Jul. 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/82687.html>
- [269] International Organization for Standardization, “Agricultural Machinery and Tractors—Safety of Partially Automated, Semi-Autonomous and Autonomous Machinery—Part 4: Verification Methods and Validation Principles (ISO 18497-4).” ISO, Geneva, Switzerland, Jul. 2024. Accessed: Jul. 19, 2026. [Online]. Available: <https://www.iso.org/standard/82688.html>