Universitatea „Dunărea de Jos” din Galaţi
Școala doctorală de inginerie

TEZĂ DE DOCTORAT

CONTRIBUŢII LA ACŢIONAREA ELECTRICĂ A LINIILOR DE FABRICAȚIE FLEXIBILĂ ŞI A ROBOŢILOR INTEGRAŢI

Doctorand,

Ing. Adriana FILIPESCU

Conducător științific,

Prof. univ. Dr. Ing.Grigore FETECĂU

Seria I 3. Inginerie electrică, Nr. 4

GALAŢI

2016

Universitatea „Dunărea de Jos” din Galaţi

Școala doctorală de inginerie

TEZĂ DE DOCTORAT

CONTRIBUŢII LA ACŢIONAREA ELECTRICĂ A LINIILOR DE FABRICAȚIE FLEXIBILĂ ŞI A ROBOŢILOR INTEGRAŢI

Doctorandă,

Ing. Adriana FILIPESCU

Preşedinte,	**Prof. univ. dr. ing. Luminiţa DUMITRIU**
Conducător științific:	**Prof . univ.dr.ing. Grigore FETECĂU**
Referenți stiințifici:	**Prof. univ.dr.ing. Mihai IORDACHE** **Prof. univ.dr.ing. Emil CAZACU** **Prof . univ.dr.ing. Marian GĂICEANU**

Seria I 3. Inginerie electrică, Nr. 4

GALAŢI

2016

Seriile tezelor de doctorat sustinute public în UDJG începând cu 1 octombrie 2013 sunt:

Domeniul **ȘTIINȚE INGINEREȘTI**

Seria I 1**: Biotehnologii**
Seria I 2**: Calculatoare și tehnologia informației**
Seria I 3**. Inginerie electrică**
Seria I 4**: Inginerie industrială**
Seria I 5: **Ingineria materialelor**
Seria I 6**: Inginerie mecanică**
Seria I 7**: Ingineria produselor alimentare**
Seria I 8: **Ingineria sistemelor**

Domeniul **ȘTIINȚE ECONOMICE**

Seria E 1: **Economie**
Seria E 2: **Management**

Domeniul **ȘTIINȚE UMANISTE**

Seria U 1: **Filologie- Engleză**
Seria U 2: **Filologie- Română**
Seria U 3**: Istorie**

Cuprins……..5

Contents……8

Prefață

Această lucrare este rezultatul cercetărilor efectuate în perioada octombrie 2012 – 30-septembrie 2016 în cadrul Universității "Dunărea de Jos" din Galați, Facultatea de Calculatoare, Automatică, Inginerie Electrică și Electronică, Departamentul de Automatică și Inginerie Electrică.

Cercetarea și teza de doctorat s-au facut cu suportul logistic și financiar din cadrul proiectului UEFISCDI-PARTENERIATE, cod proiect PN-II-PT-PCCA-2013-4-0686, **Prototipuri de sisteme robotice autonome destinate asistenței medico-sociale şi deservirii unor procese de fabricaţie din metalurgie, ceramică, sticlă și industria de automobile (ProRobSis)**. Proiectul se derulează în perioada 1.iulie.2014-30.septembrie.2017, eu facând parte din echipa de cercetare ca student doctorand.

În primul rând, îi mulțumesc domnului profesor Grigore FETECĂU, coordonatorul Tezei, pentru susținerea de a urma studiile doctorale în domeniul Inginerie electrică. Înțelegerea, ajutorul, sfaturile și cunoștințele sale au reprezentat un real suport fără de care această teză nu ar fi putut fi elaborată.

Mulțumesc domnilor profesori Mihai IORDACHE și Emil CAZACU, de la Universitatea Politehnică București, pentru că au acceptat calitatea de referenți oficiali, din postura în care au analizat cu profesionalism teza și au facut sugestii și observații pertinente și la obiect.

Îi mulțumesc domnului profesor Marian GĂICEANU, în calitate de referent oficial și domnului conferențiar VONCILĂ, acesta din urmă directorul Departamentul de Automatică și Inginerie Electrică, ambii fiind membri în comisia de îndrumare, pentru ajutorul acordat în perioada stagiului cât și pentru atenția și minuțiozitatea cu care au analizat rapoartele științifice și teza.

Nu în ultimul rând, mulțumesc tatălui meu pentru înțelegerea, dragostea și sprijinul moral acordat pe perioada cercetărilor, oferindu-mi motivația și condițiile necesare pentru realizarea și finalizarea tezei de doctorat.

Un gând ales se îndreaptă și către mama care, deși a părăsit această lume, mi-a insuflat ambiția și forța de a merge mai departe, de a nu mă da bătută și de a lupta cu greutățile vieții până la capăt, convinsă fiind că voi învinge.

Mulțumesc tuturor celor care m-au încurajat îndrumat și ajutat la elaborarea tezei de doctorat.

Ing. Adriana FILIPESCU,
Galați, Noiembrie 2016

Notații și abrevieri

A/D-assembly/disassembly (asamblare/dezasamblare);
A/DML-assembly/disassembly mechatronics line (linie de mecatronica de asamblare/dezasamblare);
ALB-assembly line balancing (echilibrare linie de asamblare);
A/DLB-assembly/disassembly line balancing (echilibrare linie de asamblare/dezasamblare);
API-application programming interface (interfață pentru programarea de aplicații);
ARIA-advanced robotic interface for applications (interfață robotică avansată pentru aplicații);
CAM-computer aided manufacturing (fabricație asistată de calculator);
CAD-computer aided design (proiectare asistată de calculator);
CAE-computer aided engineering (inginerie asistată de calculator),
CAQ-computer aided quality (calitate asistată de calculator);
CAP-computer aided planning (planificare asistată de calculator);
CAS-computer aided service (întreținere asistată de calculator);
CNC-computer numerical control (control numeric pe calculator);
CT-cycle time (timp de ciclu);
DES-discrete events system (sistem cu evenimente discrete);
DLB-disassembly line balancing (echilibrare linie de dezasamblare);
DOF-degree of freedom (grad de libertate);
DV-decision variable (variabilă de decizie);
DW-driving wheel (roată motoare);
ED-electric drive (acționare electrică);
FML-flexible manufacturing line (linie de fabricație flexibilă);
FMML-flexible manufacturing mechatronics line (linie de mecatronică de fabricaţie flexibilă);
FMC-flexible manufacturing cell (celulă de fabricație flexibilă);
FW-free wheel (roată liberă);
GUI-graphic user interface (interfață grafică utilizator);
HPN-hybrid Petri net (reţea Petri hibridă);
I/O-input/output (intrare/ieșire);
LB-line balancing (echilibrare linie);
ML-mechatronics line (linie de mecatronica);
MPI-message passing interface (interfață pentru mesaje);
MPS-mechatronics processing system (sistem mecatronic de procesare);
NR-net revenue (valoare netă obținută din vânzare sau reutilizare);
OOP-object oriented programming (programare orientată pe obiecte);
PC-Personal computer;
PLC-programmable logic controller (automat programabil);
PN-Petri net (rețea Petri);
P/R-processing/reprocesing (prelucrare/reprelucrare);
P/RML-processing/reprocesing mechatronics line (linie de mecatronică de prelucrare/reprelucrare);

Profibus DP-professionl field bus descentralized periphery (magistrală profesională pentru mediu industrial și periferie distribuită);
Profinet-professional network (protocol de comunicație în rețea pe magistrala profibus DP);
RM-robotic manipulator (manipulator robotic);
SHPN-sinchronised hybrid Petri net (retea Petri hibridă sincronizată);
SIMATIC-SIEMENS family controllers for automation (familie de controller SIEMENS pentru automatic);
SIMATIC STEP 7-software package for SIEMENS controllers (pachet software pentru programarea PLC-urilor SIEMENS);
SP-strategic planning (planificare strategică);
SIPs-server information packets (pachete de informare server);
SMC-sliding-mode control (conducere în mod alunecător);
SM-TT-sliding mode-trajectory tracking (conducere cu mod alunecător-urmărirea traiectoriei);
TC-task cycle (task al ciclului);
TPN-timed Petri net (reţea Petri temporizată);
THPN-timed hybrid Petri net (reţea Petri hibrida temporizată);
TP-task planning (planificare taskuri);
TT-trajectory tracking (urmărirea traiectoriei);
WMR-wheeled mobile robot (robot mobil cu roți).

Listă figuri și tabele

Introducere

Linii de fabricație flexibilă deservite de sisteme robotice mobile

Până acum industria bazată pe tehnologia de producție era deservită de roboți care, în general, aveau poziții fixe și îndeplineau sarcini de manipulare. În ultimul timp, această industrie a intrat într-un proces de evoluție datorat progresului tehnologic. În prezent, evoluția în cauză a intrat într-o nouă etapă, caracterizată de sisteme de fabricație flexibilă, complet automatizate și conduse în regim distribuit și supervizat. Acest nou context a necesitat dezvoltarea de noi sisteme robotice, scule si mașini de procesare care să permită transportul şi manipularea pieselor într-o manieră cât mai eficientă, [2], [4], [5], [7], [8]. Abordarea care se propune în teză răspunde noilor concepte de planificare și conducere a proceselor de fabricație flexibilă, de asamblare/dezasamblare (A/D), [6], [28], [30], [71], [73], și de prelucrare/reprelucrare (P/R), [29], [30], [63], [67], [68], pe sisteme de laborator, linii de mecatronică deservite de platforme robotice mobile echipate cu manipulatoare. Aceste structuri de laborator au corespondent în industria reală, mai ales, procesele de asamblare şi prelucrare din industria de automobile, la asamblarea caroseriei, a cutiei de viteze și a blocului motor. La procesele existente, manipulatoarele robotice au poziții fixe. Prin cercetarea din teză, s-a dorit îmbunătățirea gradului de automatizare și eficientizarea acestor linii de producție, prin utilizarea roboților mobili echipați cu manipulatoare. Astfel, liniile de asamblare devin reversibile, fiind capabile să facă și dezasamblare, dezasamblare care permite recuperarea și reutilizarea componentelor și subansamblelor, în eventualitatea că produsul final nu corespunde standardelor de calitate. Liniile de prelucrare devin capabile să reproceseze componente care nu corespund calitativ. Liniile de mecatronică, de asamblare sau prelucrare sunt sisteme de fluxuri de producție, unde unitățile de producție execută operații pe stații de lucru, care pot fi configurate serial, paralel, circular, în forma de U, celular sau paralel. Piesa aflată în lucru parcurge stațiile succesiv, deplasându-se de-a lugul liniei prin intermediul unui sistem de transport, de exemplu un sistem de benzi transportoare, [8], [34], [70], [83]. Operațiile de dezasamblare implică separarea componentelor reutilizabile din produsul final, considerat rebut. Reprelucrarea implică întoarcerea produsului final pe linia de producţie, pentru a-i fi efectuate încă odată, în parte sau integral, operațiile de prelucrare cu scopul de a satisfice cerințele de calitate, [93]. Astfel, sistemele de producție devin recuperative, prin oparații de refabricație sau vânzare către furnizori, [50], [51], [75]. Liniile de mecatronică, de asamblare/dezasamblare (A/DML) și de prelucrare/reprelucrare (P/RML), sunt sisteme complexe, conduse în timp-real, care execută sarcini aflate sub diverse condiționări.

Stadiul actual. Abordarea hibridă.

În ultimul timp, sistemele hibride au avut parte de o atenție considerabilă. A/DML și P/RML deservite de roboți mobili au caracteristici hibride, având două dinamici, una continuă și cealaltă discretă, bazată pe evenimente. HPNs sunt instrumente de modelare a unor astfel de sisteme [1], [15], [31], [41], [62], [67], [72], [73]. Planurile de A/D sunt făcute ca

subansamblele să poată fi împreunate, [16], [93]. Planurile de P/R sunt operații care se execută secvențial (găurire, alezare, extrudere, îndoire, etc), serial sau paralel astfel ca produsul final să îndeplinească cerințele de calitate. Particularizând, subiectele de cercetare relevante sunt reprezentările de A/D și P/R, poziționarea stațiilor și celulelor de lucru, planificarea planurilor de operații și a sarcinilor, etc. Planificarea off-line a taskurilor reprezintă un domeniu mai vast, ce cuprinde un set de metodologii de planificare a operațiilor, poziționarea și interogarea senzorilor, planificarea operațiilor de manipulare, planificare traiectoriilor aferente roboților mobili, [39], planificare poziționări grosiere și de precizie, [22]. Planificarea on-line reprezintă execuția și reacția de felul cum se fac on-line planurile de producție, cum se execută și monitorizează planurile off-line și cum se poate reacționa la situații variate în decursul execuției planurilor, [35], [42]. Toate aceste aspecte legate de planificarea on-line pot fi clasificate în: monitorizare, programare reactivă, și intervenție bazată pe comportament. Planificarea operaţiilor de A/D și P/R necesită cerințe complexe cum ar fi: interfațare geometrică, masurători de precizie, evaluare, programarea resurselor, conducere bazată pe model și planificare de sistem. Această planificare este o sarcină complexă pe liniile de A/D și P/R într-un mediu de fabricație concurențial şi flexibil. Combinând toți aceşti factori, rezultă că planificarea de A/D și P/R este mult mai dificilă și necesită din partea proiectantului și inginerului de producție experiență și cunoștințe vaste. Până acum au fost propuse numeroase tehnici de generare, reprezentare și planificarea taskurilor, ca: matrici binare, grafuri directe, stabilirea condiționărilor, relații de precedenţă sau grafuri AND/OR, [5]. Trebuie căutate planurile de asamblare sau producție, pentru a proiecta inteligent și eficient A/D sau P/R, unde operatorii autonomi, robotici sau umani, execută o sarcină dată, bazată pe informaţii sigure de proiectare, stocare și măsurare. Oricum, sistemele robotice mobile echipate cu manipulatoare și strategiile orientate pe caracteristicile de sistem sunt mult mai eficiente decât tehnicile bazate pe metode independente de domeniu. Reprezentarea convențională a modelelor de sisteme, fără constrângeri, poate conduce la un spațiu uriaș de planuri fezabile. Utilizând aceste modele, planificatorii de taskuri pot determina secvența de componente care trebuie îndepărtate sau secvența de operații de reprocesare necesare pentru a obține o secvență specifică de taskuri. Dacă obiectivul constă în a dezasambla anumite componente sau a efectua o anumită prelucrare, atunci planificatorul de taskuri poate furniza cea mai bună secvenţă pentru atingerea ei, [57], [66], [68], [73]. Dacă produsul final asamblat sau prelucrat, în întregime, nu trece testul de calitate, planificatorul de taskuri furnizează cea mai bună secvenţă pentru dezasamblarea sau reprelucrarea completă. Abordarea planificării taskurilor aferente dezasamblării sau reprelucrării, reclamă o cunoaștere cât mai bună a constrângerilor tehnice şi economice, [15], [41], [42], [43], [44]. Dezvoltarea cunoașterii bazată pe model HPN, integrat cu o secvenţă generatoare de algoritm, a fost cu succes aplicată la modelarea şi planificarea proceselor de dezasamblare flexibilă ca sistem de nivel înalt. Oricum tipologia roboților mobili, metodă de planificare a dezasamblării și nivelul de planificare a taskurilor, îmbunatățesc eficiența întregului proces și contribuie la reducerea costurilor pe produs dezasamblat. Definirea și planificare taskurilor, la nivelul de jos, constă în schimbarea modelelor sau a secvențelor de operații [61], [65], [66], [71], [83].

Linii de mecatronică, de fabricație flexibilă, deservite de sisteme robotice mobile

Se precizează, încă de la bun început, că versiunea în limba română, din secțiunea ”Introducere”, utilizează abrevierile din limba engleză. Litera “s” pusă ca sufix al abrevierii, semnifică pluralul substantivelor. Totuși, acordurile se fac respectând gramatica limbii

române. Cercetarea din teză conține și obținerea de modele generale SHPN asociate la două linii de mecatronică, A/DML și P/RML, deservite de WMRs echipate cu MRs. Modelele SHPN s-au particularizat pe linia de mecatronică, de asamblare, HERA&HORSTMANN, care asamblează un produs din cinci componente şi pe linia de mecatronică, de prelucrare, FESTO MPS-200, formată din 4 stații de lucru. La A/DML s-au considerat două cazuri: primul, când linia de mecatronică este deservită de un singur WMR echipat cu RM; al doilea, când A/DML este deservită de doi WMRs, lucrând în paralel, unul este echipat cu RM, utilizat pentru manipulare, iar al doilea, fără RM, este utilizat pentru transport. Utilizând platforme LabView, [54], sau Vizual C++, [98], s-a implementat conducerea în timp real a A/DML deservită de unul sau doi WMRs, conducere bazată pe modelele SHPN. Aceste modele furnizează și o descriere la nivel înalt a produsului supus dezasamblării. Obiectivul major a fost sa se atribuie taskuri asociate operațiilor de dezasamblare efectuate pe stațiile de lucru pentru a se maximiza valoarea totală a componentelor recuperate. Operațiile de dezasamblare sunt efectuate pe aceiași linie de asamblare formată din stații de lucru dispuse liniar. Prima stație de lucru preia produsul care trebuie dezasamblat, iar componentele sunt dezasamblate pe staţii diferite. Un ciclu se consideră terminat când toate componentele sunt dezasamblate. În această teză, taskurile asociate A/D sunt prinse în modelele SHPN, modele care surprind un dublu aspect: aspectul discret asociat operațiilor de A/D și aspectul continuu asociat deplasării și operațiilor de manipulare executate de sistemele robotice. De asemenea, în teză este prezentat un model SHPN și pentru P/RML deservită de un WMR echipat cu RM. Problema devine critică în ceea ce privește minimizarea utilizării resurselor (timp şi bani) în reprocesare. Obiectivul este de a atribui taskuri corespunzatoare stațiilor de lucru pentru a reprocesa piesele care nu trec testul de calitate. Operațiile de reprocesare sunt executate pe aceiasi linie de fabricație. Bazată pe modelul SHPN și prin utilizarea unei platforme MATLAB,[96], este implementată conducerea în timp real a P/RML deservită de un WMR echipat cu RM. Dinamica evoluției în spațiul stărilor a A/DML și P/RML este determinată de evenimente generate de secvențele de comandă ale automatului de sistem (PLC) şi de interacțiunea cu WMRs, acesta din urmă reprezintă componenta cu timp continuu a sistemului. Sistemul considerat este de tip hibrid şi necesită instrumente specializate pentru modelare. Modelele hibride sunt elaborate utilizând instrumente dedicate, HPN, descrise în [1], [15], [29], [30], [31], [62], [63], [67], [68]. Un model SHPN rezultă din combinația dintre modelul DES al sistemului analizat și modelul cu timp continuu al WMR echipat cu RM. Modelele SHPN propuse în teză au fost analizate şi verificate prin simulare în pachetul Sirphyco, [97].

Obiectivele tezei

- Modelarea, acționarea și conducerea roboților mobili (WMRs) echipați cu manipulatoare robotice (RMs);
- Planificarea taskurilor, echilibrarea (optimizarea), modelarea hibridă, simularea, acţionarea și conducerea liniei de mecatronică de asamblare/dezasamblare (A/DML) deservită de un robot mobil echipat cu manipulator. Cazul general. Particularizare la A/DML HERA&HORSTMANN, deservită de un WMR, Pioneer 3-DX, echipat cu un RM, Pioneer 5-DOF Arm;
- Testarea și validarea în laborator a tehnologiei hibide de fabricație flexibilă pe A/DML, HERA&HORSTMANN, deservită de un WMR echipat cu RM;
- Planificarea taskurilor, echilibrarea (optimizarea), modelarea hibridă, simularea, acţionarea și conducerea liniei de mecatronică de asamblare/dezasamblare (A/DML) deservită de doi roboți mobili echipați cu manipulatoare. Cazul general.

Particularizare la A/DML, HERA&HORSTMANN, deservită de doi roboți mobili colaborativi, lucrând în paralel, unul dintre ei, Pioneer 3-DX, echipat cu un RM, Pioneer 5-DOF Arm, utilizat pentru manipulare, și un al doilea, PatrolBot, utilizat pentru transpot. Testare în laborator;

- Planificarea taskurilor, modelarea hibridă, simularea, acţionarea și conducerea liniei de mecatronică, de procesare/reprocesare (P/RML), FESTO MPS-200, deservită de un WMR, Pioneer 3-DX, echipat cu un RM, Pioneer 6-DOF Arm;
- Testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe P/RML, FESTO-MPS 200, deservită de un WMR echipat cu RM;

Conținutul capitolelor

În Capitolul 1, sunt prezentate modelarea cinematică, acționarea electrică și conducerea roboților mobili cu două roți motoare și una sau două roţi libere(2DW/1FW, 2DW/2FW). De asemenea, sunt prezentate modelarea, acționarea electrică și conducerea roboților mobili 2DW/1FW Pioneer 3-DX și 2DW/2FW Patrol Bot;

Structura, funcționalitatea, acționarea electrică și conducerea liniilor de fabricație flexibilă deservite de sisteme robotice cu particularizare la A/DML, HERA&HORSTMANN, şi P/RML, FESTO MPS-200, sunt prezentate în Capitolul 2;

În Capitolul 3, se fac ipoteze preliminare utile la planificarea taskurilor şi echilibrarea FMMLs A/DML și P/RML cu roboți integrați. Mai întâi, pentru A/DML, deservită de una sau două platforme mobile, se face o generalizare la *N* stații de A/D. Urmează, particularizarea la A/DML HERA&HORSTMANN cu 5 stații de A/D. Tot la A/DML, este propus un criteriu de optimizare care poate fi asimilat și ca un criteriu de echilibrare a liniei pentru secvenţa de operații de dezasamblare. Minimizarea criteriului este echivalentă cu minimizarea costurilor în secvența de dezasamblare odată cu reutilizarea componentelor în procesul de fabricație.

În Capitolul 4 se prezintă modelului SHPN și testarea prin simulare, în Sirphyco pentru A/DML deservită de un WMR echipat cu RM, în forma generală și particularizată. De asemenea, se prezintă modelul SHPN și simularea în Sirphyco a două sisteme robotice mobile integrate în A/DML. Tot în Capitolul 4, sunt prezentate modelul SHPN și simularea în Sirphyco pentru P/RML deservită de un WMR echipat cu RM, unde sincronizarea se face şi prin intermediul unui sistem servoing vizual cu poziţie fixă sau mobilă;

Utilizând, după caz, platforme LabView, Visual C++ şi MATLAB, în Capitolul 5, se prezintă conducerea în timp real a roboților mobili integrați în A/DML și P/RML, conducere bazată pe modelele SHPN.

Concluzii finale, contribuții, direcții de cercetare viitoare şi diseminare rezultate sunt prezentate în Capitolul 6.

Introduction

Flexible manufacturing lines served by mobile robotic systems

Lately, the industry has faced with new global evolution, driven by the technological progress. This improvement extends to all industrial domains and triggers the evolution of new generations of advanced flexible production systems and new methods of centralized management distributed or supervised. In addition, this involves the evolution of new types of robots and processing machine tools and the need of efficient transport and manipulation systems, [2], [4], [5], [7], [8]. The approach proposed in this thesis responds to the concepts of planning and control of the manufacturing of assembly/disassembly (A/D), [6], [28], [30], [71], [73] and of processing/reprocessing (A/P), [29], [30], [63], [67], [68] on laboratory systems, mechatronics lines served by mobile platforms equipped with manipulators, with emphasis on the planning of operations. The most eloquent correspondents in the real world are assembly processes in the automotive industry, car body, gearbox and engine block assembly. In most cases, robotic manipulators that have a fixed location serve these assembly and/or processing lines. Through this study, we extended the degree of automation and efficiency of these production lines using mobile robotic systems equipped with manipulators. The assembly lines become reverse, being able to recover and reuse of components and subassemblies, in the event that the final product does not meet quality requirements. The processing lines become able to reprocess components that support this operation, in the event that the final product does not meet quality standards. Assembly and processing mechatronics lines are flow-oriented production system where the productive units perform operations on workstations, which may be configured as serial, parallel, circular, U-shaped, cellular or two-sided lines. The work pieces visit stations successively as they are moved along the line, usually by some kind of transport system, e.g, a conveyor belt [8], [34], [70], [83]. Disassembly operations involve separation of the reusable parts from the discarded products. Reprocessing involves the return piece on the production line to be subject to the same processing operations, to fit the required parameters, [93]. These parts undergo remanufacturing operations or solding to suppliers, [50], [51], [75]. Assembly/disassembly (A/D) and processing/reprocessing (P/R) manufacturing systems are real-time and complex control systems, which involve multiple operation conditions and tasks.

State of Art. Hybrid aproach.

Hybrid systems are currently the focus of considerable attention. A/D and P/R manufacturing lines served by mobile robots have hybrid characteristics, consisting

of continuous dynamic behaviours and discrete event behaviours. Hybrid Petri Nets (HPNs) are tools used to model such systems, [1], [15], [31], [41], [62], [67], [72], [73]. A/D plans are made up of parts or subassemblies that are fitted together, [16], [93]. P/R plans are operation sequences (drilling, reaming, extrusion, bending, etc.), executed in serial or parallel so that the product meets the quality requirements. Particularly relevant research topics include A/D and P/R representations, work-cell planning, sequence planning, etc. Off-line task planning is a large area encompassing a diverse set of planning methodologies capable of producing a detailed operation plan, including planning sensory action, planning manipulator action, planning the trajectory of mobile robots, [39], rough motion planning, fine motion planning and other planning, [22]. On-line planning addresses execution and reaction issues such as how to develop plans on-line, how to execute and monitor a plan developed off-line, and how to react to various situations that arise during plan execution, [35], [42]. These issues can be further classified into: plan monitoring, reactive scheduling, and behaviour-based action. A/D and P/R planning processes involve complex requirements such as geometric relationships, performance measurement, evaluation, resource scheduling, kinematics control, and system planning.This is a difficult task for complex A/D and P/R lines in a concurrent and flexible manufacturing environment. These factors combined make real A/D and P/R planning more difficult and require extensive experience and knowledge on the part of the designer and production engineer. Up to now, numerous techniques in task planning, such as use of binary matrices, directed graphs, establishment conditions, precedence relationships, AND/OR graphs, [5], have proposed for generation and representation. Have to search assembly or processing plans to design intelligent and efficient A/D or P/R, where operators (robot or human) autonomously perform a given task based on certain designated, stored or sensed information. However, mobile robotic systems with manipulators and planning strategies oriented to the characteristics of the system is often more effective than techniques derived from domain-independent methods. Conventional representation of the system models without constraints may result in a huge search space for feasible plans. Using these models, the task planners can determine the sequence of parts that must remove or the sequence of reprocessed operations to achieve specific sequences of tasks. If the target consists of disassembling a specific part or performing a specific operation, the task planner can provide the best sequence for reaching it, [57], [66], [68], [73]. If the fully assembled or processed product fails the quality test, the task planner provides the best sequence for completely disassembling or reprocessing.A comprehensive knowledge-based approach to disassembly or reprocessing task planning is required, which considers all aspects of complex interaction and domain knowledge subjected to technical and economic constraints, [15], [41], [42], [43], [44]. Development of knowledge based on a HPN model integrated with a sequence generation algorithm was successfully applied to modelling and planning of a flexible disassembly process and system at a high level. However, the typology of the autonomous mobile robot with manipulator, disassembly planning method, and task level planning, greatly improves the efficiency of the entire process and reduces the

cost of product disassembly. Task specification in low-level task planning consists in changing models or operation sequences, [61], [65], [66], [71], [83].

Flexible manufacturing mechatronics line served by mobile robotic systems

The research performed in the framework of thesis also includes getting of generalised Synchronised Hybrid Petri Nets (SHPN) models for two mechatronics lines: Assembly/Disassembly Mechatronics Line (A/DML) and Processing/Reprocessing Mechatronics Line (P/RML) served by Wheeled Mobile Robot (WMRs) equipped with a Robotic Manipulator (RMs). SHPN models have customised for an assembly mechatronics line, HERA&HORSTMANN, which assembles a 5-part product and a mechatronics line, FESTO MPS-200, of 4-workstations, respectively. To A/DML, two cases have considered: first, A/DML is served by a WMR equipped with RM; second, A/DML is served by two WMRs, parallel working, one is equipped with RM, used for manipulation and the other used for transport. Using the LabView, [54], or Visual C++ platform, [98], real-time control of A/DML served by one or two WMRs based on SHPN models is presented. These models provide a high-level description of the product to be disassembled. The aim is to assign the tasks to the disassembly line workstations so as to maximise the total value of the recovered parts. The disassembly operations are performed on the same assembly line, consisting of a number of linear configured workstations. The first workstation takes the product to be disassembled and the parts are disconnected on different workstations. A cycle terminates, i.e. the product leaves the line, whenever all its required parts are disassembled. In this thesis, the concepts of A/D tasks are caught in SHPN models, which complies with both aspects: the discrete approach for the elementary A/D operations, and the continuous approach for displacement and handlings operations executed by the robotic systems. Also, a SHPN model for P/RML, served by the WMR equipped with RM is presented. The problem is critical for minimizing the use of valuable resources (such as time and money) invested in reprocessing, and maximizing the level of automation of the process. The aim is to assign the tasks to the processing line workstations to reprocess the pieces that fail the quality test. The reprocessing operations are performed on the same line. Using the MATLAB platform,[96] real-time control of P/RML served by one WMR based on SHPN models is presented. A/DML and P/RML dynamics are triggered by events, supplied by the control sequences of the automatic system, and by interaction with the WMRs, which represent the continuous time part of the system. The considered systems are hybrid ones and requires specialised tools for modelling.The hybrid models are elaborated using the dedicated modelling tool, HPN, described in [1], [15], [29], [30], [31], [62], [63], [67], [68]. A SHPN model results from the combination of the SED model of the analysed systems with the cyclic and continuous time of the WMRs with RM. The proposed models, have been tested, analysed and verified through simulation package Sirphyco, [98].

Thesis objectives

- Modeling, acting and control of WMRs equipped with RMs:
- Task planning, balancing (optimising), hybrid modeling, simulation and control of A/DML served by a WMR equipped with RM. General case. Customization to A/DML, HERA&HORSTMANN, served by one WMR, Pioneer 3-DX, equipped with RM, Pioneer 5-DOF Arm;
- Flexible manufacturing hybrid technology fully automatic, on A/DML, HERA&HORSTMANN, served by one WMR with RM, tested in laboratory;
- Task planning, balancing, hybrid modeling, simulation and control of A/DML served by two WMRs equipped with RM. General case. Customization to A/DML, HERA&HORSTMANN, served by two collaborative WMRs one of them equipped with RM, Pioneer 3-DX, equipped with RM, Pioneer 6-DOF Arm, used for manipulation, and the other, PatrolBot, used for transport. Testing in laboratory;
- Task planning, hybrid modeling, simulation and control of P/RML, FESTO MPS-200 served by one WMR, Pioneer 3-DX, equipped with RM, Pioneer 5-DOF Arm;
- Flexible manufacturing hybrid technology, based on visual servoig systems, fully automatic, on P/RML, FESTO MPS-200, served by one WMR with RM, tested and validated in laboratory.

Chapters' content

In Chapter 1, kinematic modeling, electric drive and control of WMRs with two driving wheels and one or two free wheels (2DW/1FW, 2DW/2FW) are presented. Also, Modeling, electric drive and control of WMRs: 2DW/1FW Pioneer 3-DX and 2DW/2FW Patrol Bot are presented.

Structure, functionality, electric drive and control of flexible manufacturing lines served by robotic systems with customization to A/DML, HERA&HORSTMANN, and P/RML, FESTO MPS-200 are presented in Chapter 2.

In Chapter 3, useful preliminary assumption, hardware, task planning and balancing are laid out for WMRs integrated into A/DML and P/RML.

SHPN model and its simulation in Sirphyco of the A/DML served by one WMR with RM in generalised and customised form, is shown in Chapter 4. Also, SHPN model and its simulation in Sirphyco of two WMRs integrated into A/DML, are presented in Chapter 4. Also, in Chapter 4, it is presented SHPN model and its simulation of the P/RML served by a one WMR with RM.

Using Labview, Visual C++ and MATLAB platforms, real-time control of WMRs servicing A/DML and P/RML based on SHPN models, are presented in Chapter 5.

Final remarks, contributions, future research directions and dissemination of results can be found in Chapter 6.

Capitolul 1

Contribuții la acționarea electrică și conducerea roboților mobili și a manipulatoarelor robotice

În acest capitol se prezintă acționarea și conducerea roboților mobili (WMRs) cu două roți motoare și una sau două roți libere (2DW/1FW si 2DW/2FW), conducere bazată numai pe modelul cinematic. De precizat, că modelul cinematic este identic, atât pentru WMR 2DW/1FW, cât și pentru 2DW/2FW. Motivația utilizarii numai a modelului cinematic constă în faptul că aceşti roboți integrați în liniile de mecatronică nu necesită transportul și manipularea de sarcini mari iar conducerea în buclă închisă nu reclamă anumite proprietăți de robustețe, [23], [24], [25]. La manipulatoarele robotice care echipează platformele robotice, conducerea este una în buclă deschisă, datorită aceloraşi motive, expuse mai sus pentru roboții mobili. Modelul cinematic descrie mișcarea robotului și nu ia în calcul forțele care acționează asupra lor. Aceste modele au fost folosite pentru proiectarea a două structuri sliding-mode (Sliding-Mode Control-SMC) pentru conducerea trajectory-tracking (TT) a roboților mobili 2DW/1FW și 2DW/2FW, cu timp continuu și cu timp discret, [17], [18], [19], [20]. Testarea prin simulare și în timp real a acestor structuri de conducere s-a făcut pentru WMR 2DW/1FW, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm și 2DW/2FW, PatrolBot. Aceste sisteme robotice vor fi integrate și vor deservi A/DML și P/RML.

1.1. Determinarea modelului cinematic al WMRs

Se consideră WMR cu 2 roți motoare și una sau două libere (2DW/1FW sau 2DW/2FW) prezentat în Fig.1.1, caracterizat de variabila generalizată, $q=(q_1,q_2,...,q_n)$. Se presupune că roțile robotului se rotesc fără a aluneca, astfel încât robotul este supus numai constrângerilor nonholonomice descrise de ecuația

$$A(q)\cdot\dot{q}=0 \tag{1.1}$$

unde $A(q)$ este matricea asociată constrângerilor.

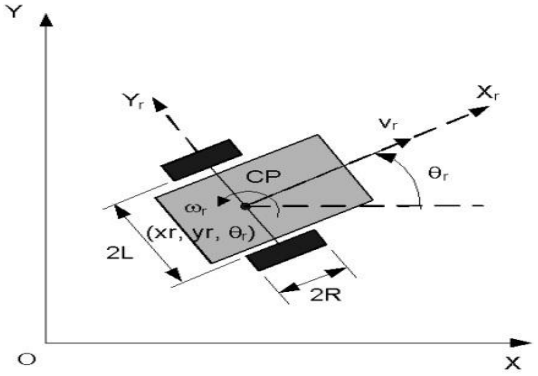


Fig. 1.1. Modelul cinematic al WMR cu 2DW/1FW sau 2DW/2FW

Se consideră modelul din [33], robotul mobil din Fig.1.1, Oxy sistemul de coordonate fix, CPX_rY_r sistemul de coordonate atașat robotului, d distanța dintre CP și centrul de greutate, cu CP aflat la mijlocul distanței dintre roțile motoare. În acest caz avem constrângerile:

$$\dot{y}_r \cdot \cos\theta_r - \dot{x}_r \cdot \sin\theta_r - d \cdot \dot{\theta} = 0 \tag{1.2}$$

$$x_r \cdot \cos\theta_r + y_r \cdot \sin\theta_r + L \cdot \dot{\theta} = R \cdot \dot{\varphi}_d \tag{1.3}$$

$$x_r \cdot \cos\theta_r + y_r \cdot \sin\theta_r - L \cdot \dot{\theta} = R \cdot \dot{\varphi}_s \tag{1.4}$$

Matricea $A(q)$ devine

$$A(q) = \begin{bmatrix} \sin\theta_r & -\cos\theta_r & d & 0 & 0 \\ \cos\theta_r & \sin\theta_r & L & -R & 0 \\ \cos\theta_r & \sin\theta_r & -L & 0 & -R \end{bmatrix} \tag{1.5}$$

Configurația robotului poate fi reprezentată utilizând cinci variabile generalizate $q = [x_r \quad y_r \quad \theta_r \quad \varphi_d \quad \varphi_s]$, unde (x_r, y_r) sunt coordonatele lui CP, $\theta\theta_r$ este orientarea, iar φ_d, φ_s sunt unghiurile roților motoare. Fie matricea $S(q)$ astfel încât

$$S^T(q) \cdot A^T(q) = 0 \tag{1.6}$$

$$S(q) = \begin{bmatrix} \frac{R}{2 \cdot L}(L \cdot \cos\theta_r - d \cdot \sin\theta_r) & \frac{R}{2 \cdot L}(L \cdot \cos\theta_r + d \cdot \sin\theta_r) \\ \frac{R}{2 \cdot L}(L \cdot \sin\theta_r + d \cdot \cos\theta_r) & \frac{R}{2 \cdot L}(L \cdot \sin\theta_r - d \cdot \cos\theta_r) \\ \frac{R}{2 \cdot L} & -\frac{R}{2 \cdot L} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.7}$$

Modelul cinematic al robotului devine

$$\dot{q} = S(q) \cdot \upsilon \tag{1.8}$$

unde $\upsilon = [\upsilon_d \quad \upsilon_s]$ reprezintă vitezele unghiulare ale roților motoare, dreapta şi stânga.
Din (1.6) și (1.8) se obține

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \\ \dot{\phi}_d \\ \dot{\phi}_s \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\theta_r & \frac{R}{2}\cos\theta_r \\ \frac{R}{2}\sin\theta_r & \frac{R}{2}\sin\theta_r \\ \frac{R}{2 \cdot L} & -\frac{R}{2 \cdot L} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \upsilon_d \\ \upsilon_s \end{bmatrix} \tag{1.9}$$

Se cunoaște relația dintre viteza liniară, viteza unghiulară a robotului și vitezele unghiulare ale roților motoare

$$\begin{bmatrix} \upsilon_d \\ \upsilon_s \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & \frac{L}{R} \\ \frac{1}{R} & -\frac{L}{R} \end{bmatrix} \cdot \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{1.10}$$

Utilizând relația (1.8) și (1.7) modelul cinematic se poate simplifica astfel

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{1.11}$$

în care:

x_r - reprezintă poziția robotului pe axa Ox;

y_r - reprezintă poziția robotului pe axa Oy;

θ_r - reprezintă orientarea robotului;

v_r - reprezintă viteza liniară a robotului;

ω_r - reprezintă viteza unghiulară a robotului.

Forma matricială (1.11), scrisă explicit devine

$$\begin{cases} \dot{x}_r = v_r \cdot \cos\theta_r \\ \dot{y}_r = v_r \cdot \sin\theta_r \\ \dot{\theta}_r = \omega_r \end{cases} \tag{1.12}$$

1.2. Conducerea SM cu timp continuu a WMRs cu două roți motoare și una sau două roți libere (WMR 2DW/1FW, 2DW/2FW)

Arhitectura conducerii sliding-mode în timp continuu a WMRs cu 2DW/1FW sau 2DW/2FW este prezentată în Fig.1.2, [38], [39], [40]. Această arhitectură de conducere permite robotului să urmărească o traiectorie dorită cu un profil de viteză impus.

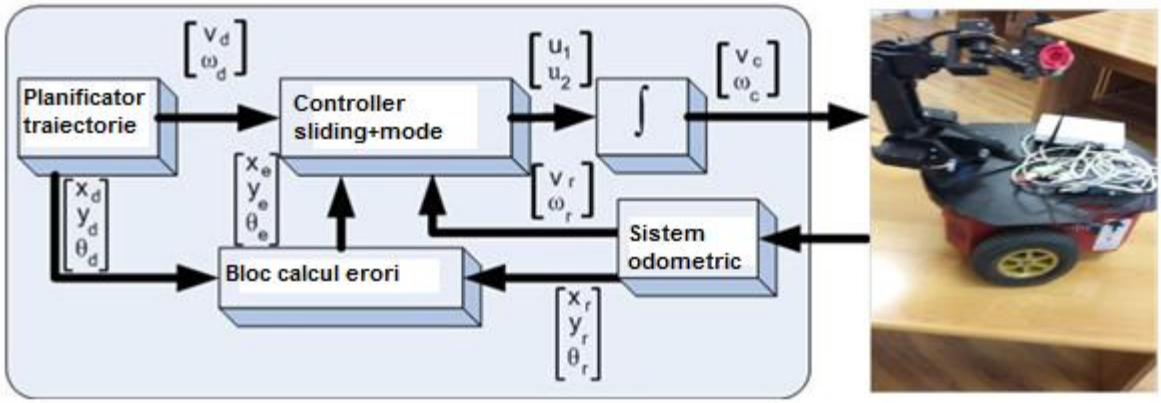


Fig.1.2. Arhitectura conducerii SM a WMR 2DW/1FW, Pioneer 3-DX

Modulul Odometry din Fig.1. 2 primește datele de la encoderele robotului și calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în softul ARIA de la Mobile Robots, [94] iar datele calculate pot fi obținute apelând în program funcțiile din ARIA.

Conducerea în regim de TT presupune că robotul urmărește un WMR virtual care se deplasează pe traiectoria dorită cu profilul de viteză impus, [76], [77], [78], [79], [80]. Se consideră că traiectoria dorită $q_d(t) = \begin{bmatrix} x_d & y_d & \theta_d \end{bmatrix}^T$ este generată de un WMR virtual, Fig.1.3.

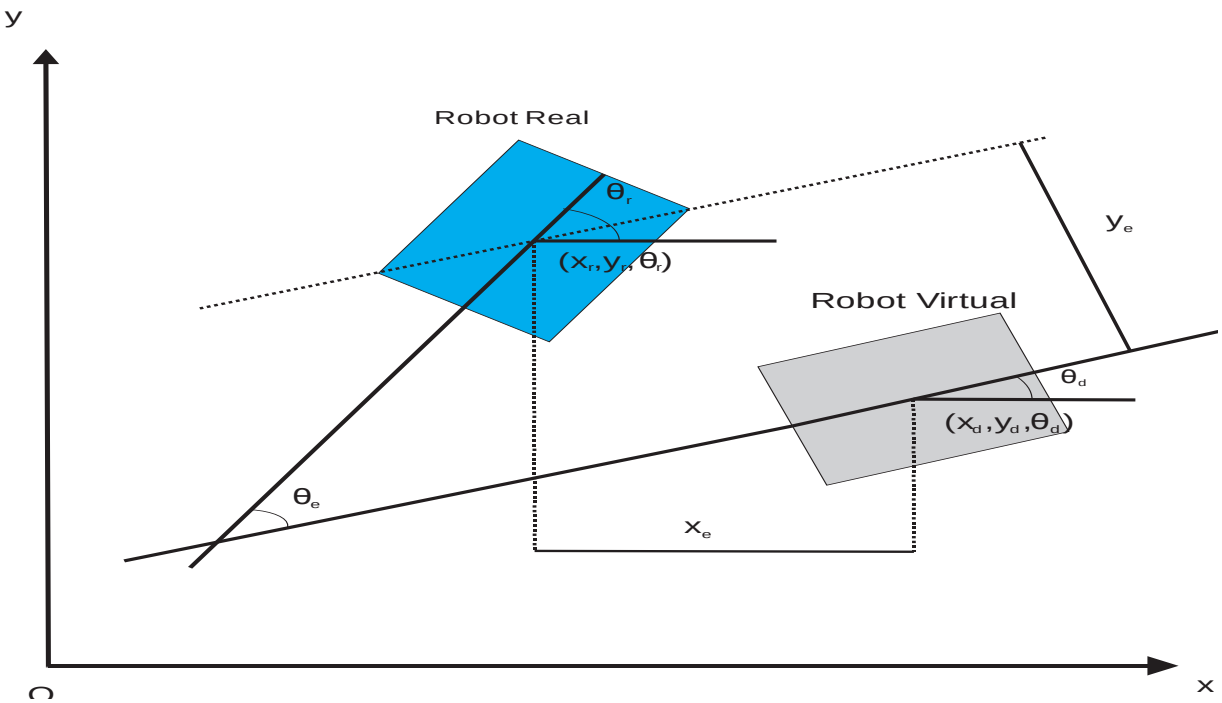


Fig.1.3. WMR virtual și WMR real, erorile de urmărire

Modelul cinematic al robotului virtual este

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \omega_d \end{cases} \tag{1.13}$$

unde (x_d, y_d) reprezintă coordonatele carteziene ale centrului geometric, v_d este viteza liniară, θ_d este orientarea și ω_d este viteza unghiulară. Când robotul este condus pentru a urmări o anumită traiectorie, se obțin erorile de urmărire, din Fig.1.3. Erorile de urmărire, pe axele Ox, Oy și de orientare sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix} \tag{1.14}$$

Dinamica erorilor de urmărire este

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \omega_d \cdot x_e \\ \dot{\theta}_e = \omega_r - \omega_d \end{cases} \tag{1.15}$$

Se presupune că $|\theta_e| < \pi/2$, astfel orientarea robotului nu este perpendiculară pe traiectoria dorită. Considerând erorile de urmărire (1.14) și derivatele lor (1.15), se definesc suprafețele de comutație:

$$\begin{cases} s_1 = \dot{x}_e + k_1 \cdot x_e \\ s_2 = \dot{y}_e + k_2 \cdot y_e + k_0 \cdot \mathrm{sgn}(y_e) \cdot \theta_e \end{cases} \tag{1.16}$$

unde $k_1, k_2, k_3 \geq 0$ sunt parametri constanți, pozitivi, iar x_e, y_e, θ_e sunt erorile de urmărire din (1.14). Dacă s_1 converge la zero, atunci x_e converge la zero. Dacă s_2 converge la 0, atunci

$\dot{y}_e = -k_2 \cdot y_e + k_0 \cdot \text{sgn}(y_e) \cdot \theta_e$. Dacă $y_e > 0$ atunci $\dot{y}_e < 0$ doar dacă $k_0 < k_2 \cdot |y_e| / |\theta_e|$. Se observă că dacă y_e și $\dot{y}_e$ converg la zero atunci θ_e converge la zero. Derivatele suprafețelor devin

$$\begin{cases} \dot{s}_1 = \ddot{x}_e + k_1 \cdot \dot{x}_e \\ \dot{s}_2 = \ddot{y}_e + k_2 \cdot \dot{y}_e + k_0 \cdot \text{sgn}(y_e) \cdot \dot{\theta}_e \end{cases} \tag{1.17}$$

Legea de conducere este folosită în forma

$$\dot{s} = -Q \cdot \text{sgn}(s) - P \cdot s \tag{1.18}$$

unde

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}, \quad P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}, \quad Q_1, Q_2, P_1, P_2 > 0,$$

$$\text{sgn}(s) = [\text{sgn}(s_1) \quad \text{sgn}(s_2)]^T, \; s = [s_1 \quad s_2]^T.$$

Din (1.14), (1.15), (1.16), (1.17) și (1.18) se obține legea de comandă SM-TT:

$$v_c = \frac{-Q_1 \cdot sign(s_1) - P_1 \cdot s_1 - k_1 \cdot \dot{x}_e - \dot{\omega}_d \cdot y_e - \omega_d \cdot \dot{y}_e + v_r \cdot \dot{\theta}_e \cdot \sin\theta_e + \dot{v}_d}{\cos\theta_e} \tag{1.19}$$

$$\omega_c = \frac{-Q_2 \cdot sign(s_2) - P_2 \cdot s_2 - k_2 \cdot \dot{y}_e - \dot{v}_r \cdot \sin\varphi_e + \dot{\omega}_d \cdot x_e + \omega_{d\cdot} \cdot \dot{x}_e}{v_r \cdot \cos\varphi_e + k_0 \cdot \text{sgn}(y_e)} + \omega_d \tag{1.20}$$

Se alege drept funcţie Lyapunov $V = \frac{1}{2} s^T s$, funcție pozitiv definită, care are derivata

$$\dot{V} = s_1 \cdot \dot{s}_1 + s_2 \cdot \dot{s}_2 = s_1 \cdot \left(-Q_1 \cdot \text{sgn}(s_1) - P_1 \cdot s_1\right) + s_2 \cdot \left(-Q_2 \cdot \text{sgn}(s_2) - P_2 \cdot s_2\right) \tag{1.21}$$

negativ semi-definită dacă se aleg parametrii $Q_i, P_i \geq 0$, [18], [19], [20].

1.3. Conducerea SM cu timp discret a WMRs cu două roți motoare și una sau două roți libere

Structura de conducere, Fig.1.4, primește de la modulul de planificare a traiectoriei, viteza liniară și viteza unghiulară dorite. De asemenea, primește erorile de poziționare de la modulul de calcul al erorilor și calculează comanda pentru viteza liniară și viteza unghiulară. Conducerea sliding-mode cu timp discret a WMR PatrolBot are structura din Fig.1.4, [17], [18], [19], [20]. Modulul Odometry din Fig.1.4 primește datele de la encodere și calculează poziția, orientarea, viteza liniară și viteza unghiulară a robotului. Acest modul este implementat în pachetul ARIA, datele calculate fiind obținute apelând, prin program, funcțiile din acest pachet.

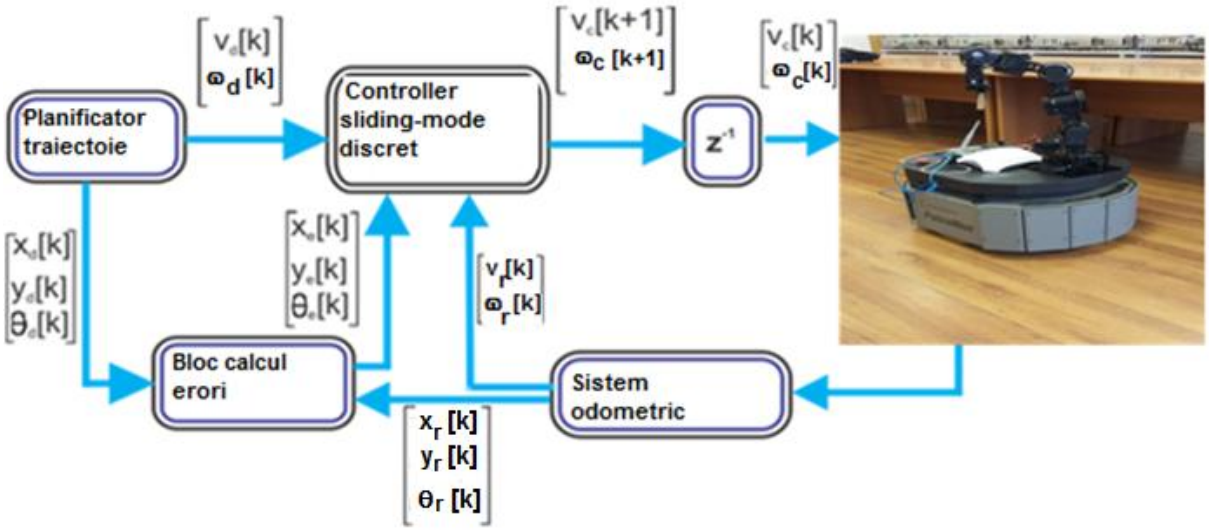


Fig.1.4. Arhitectura conducerii SM a WMR 2DW/2FW, PatrolBot

Conducerea sliding-mode cu timp discret a roboților mobili cu 2DW/2FW se realizează pornind de la ecuațiile cinematice ale centrului de greutate. Considerând perioada de eșantionare T_s și extrapolator de ordin zero, sistemul neliniar (1.12) poate fi scris în discret astfel:

$$\begin{cases} x_r[k+1] = x_r[k] + v_r[k] \cdot \cos\theta_r[k] \cdot T_s \\ y_r[k+1] = y_r[k] + v_r[k] \cdot \sin\theta_r[k] \cdot T_s \\ \theta_r[k+1] = \theta_r[k] + \omega_r[k] \cdot T_s \end{cases} \quad (1.22)$$

Problema care se pune este proiectarea unui controller capabil de urmărirea traiectoriei dorite având norma vectorului erorilor cât mai mică. Astefel, se consideră un WMR virtual având coordonatele traiectoriei dorite $q_d(t) = [x_d(t) \quad y_d(t) \quad \theta_d(t)]^T$ și modelul cinematic

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\theta}_d \end{bmatrix} = \begin{bmatrix} \cos\theta_d & 0 \\ \sin\theta_d & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_d \\ \omega_d \end{bmatrix} \quad (1.23)$$

unde:

x_d - reprezintă poziția dorită pe axa Ox;

y_d - reprezintă poziția dorită pe axa Oy;

θ_d - reprezintă orientarea dorită;

v_d - reprezintă viteza liniară dorită;

ω_d - reprezintă viteza unghiulară dorită.

Sistemul (1.20) poate fi scris în forma

$$\begin{cases} \dot{x}_d = v_d \cdot \cos\theta_d \\ \dot{y}_d = v_d \cdot \sin\theta_d \\ \dot{\theta}_d = \omega_d \end{cases} \quad (1.24)$$

Reprezentarea discretă, cu extrapolator de ordin zero a sistemului (1.24), este:

$$\begin{cases} x_d[k+1] = x_d[k] + v_d[k] \cdot \cos\theta_d[k] \cdot T_s \\ y_d[k+1] = y_d[k] + v_d[k] \cdot \sin\theta_d[k] \cdot T_s \\ \theta_d[k+1] = \theta_d[k] + \omega_d[k] \cdot T_s \end{cases} \quad (1.25)$$

Erorile de urmărire din Fig.1.4 sunt

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix} \quad (1.26)$$

Detaliat, aceste erori au expresiile

$$\begin{cases} x_e[k] = x_r[k] - x_d[k] \cdot \cos\theta_e[k] + y_r[k] - y_d[k] \cdot \sin\theta_e[k] \\ y_e[k] = -x_r[k] - x_d[k] \cdot \sin\theta_e[k] + y_r[k] - y_d[k] \cdot \cos_e \theta[k] \\ \theta_e[k] = \theta_r[k] - \theta_d[k] \end{cases} \quad (1.27)$$

Dinamica cu timp continuu a erorilor de urmărire poate fi exprimată astfel

$$\begin{cases} \dot{x}_e = -v_d + v_r \cdot \cos\theta_e + \omega_d \cdot y_e \\ \dot{y}_e = v_r \cdot \sin\theta_e - \omega_d \cdot x_e \\ \dot{\theta}_e = \dot{\theta}_r - \dot{\theta}_d \end{cases} \quad (1.28)$$

Dinamica cu timp discret a erorilor de urmărire se exprimă astfel

$$\begin{cases} x_e[k+1] = x_e[k] + (v_r[k] \cdot \cos\theta_e[k] - v_d[k] + y_e[k] \cdot \omega_d[k]) \cdot T_s \\ y_e[k+1] = y_e[k] + (v_r[k] \cdot \sin\theta_e[k] - x_e[k] \cdot \omega_d[k]) \cdot T_s \\ \theta_e[k+1] = \theta_e[k] + (\omega_r[k] - \omega_d[k]) \cdot T_s \end{cases} \quad (1.29)$$

Structura de conducere SM-TT, cu timp discret, achiziționează date de la encodere privind deplasarea robotului, calculează și eliberează comanda la momentele de discretizare T_s. O proprietate importantă a conducerii sliding-mode în timp discret este comanda discontinuă, considerată constantă la valoarea din stânga a intervalului de discretizare.

În [36] s-a propus următoarea condiție pentru existență a regimului alunecător, sliding-mode, care practic reprezintă echivalentul condiției de atractivitate a hipersuprafeței de comutație

$$s[k] \cdot (s[k+1] - s[k]) < 0 \quad (1.30)$$

Inegalitatea (1.30) este satisfăcută dacă:

$$s[k+1] = (1 - q \cdot T_s) \cdot s[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s[k]) \quad (1.31)$$

$$0 < 1 - q \cdot T < 1 \quad (1.32)$$

$$0 < \varepsilon \cdot T_s < 1 \quad (1.33)$$

unde:

$T_s > 0$ este perioada de eșantionare;

$\varepsilon > 0$ este viteza de aducere;

$q > 0$ este convergenţa exponențială.

Traiectoria sliding-mode evoluează în cvasi-banda sliding-mode

$$|s[k]| < \frac{\varepsilon \cdot T_s}{1 - q \cdot T_s} \quad (1.34)$$

Se definesc suprafețele de alunecare

$$\begin{cases} s_1[k] = x_e[k+1] + k_1 \cdot x_e[k] \\ s_2[k] = y_e[k+1] + k_2 \cdot y_e[k] + k_0 \cdot sign(y_e[k]) \cdot \theta_e[k] \end{cases} \quad (1.35)$$

unde: k_0, k_1, k_2 sunt constante pozitive x_e, y_e și θ_e sunt erorile de urmărire.

Din (1.31) și (1.35) se obțin expresiile suprafețelor de alunecare cu timp discret:

$$s_1[k+1] = x_e[k+2] + k_1 \cdot x_e[k+1] = (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \text{sgn}(s_1[k]) \quad (1.36)$$

$$s_2[k+1] = y_e[k+2] + k_2 \cdot y_e[k+1] + k_0 \cdot \text{sgn}(y_e[k]) \cdot \theta_e[k] = (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \text{sgn}(s_1[k]) \quad (1.37)$$

Din (1.36) și (1.37) se obțin comenzile, expresiile pentru viteza liniară și viteza unghiulară, [17], [18], [20]:

$$v[k+1] = \frac{1}{\cos\theta_e[k] \cdot T_s} \cdot [-(1 - q_1 \cdot T_s) \cdot s_1[k] + \varepsilon_1 \cdot T_s \cdot \text{sgn}(s_1[k] - x_e[k+1] \cdot (1 + k_1) - (v_d[k+1] - v_r[k] \cdot \theta_e[k+1] \cdot \sin\theta_e[k] - w_d[k+1] \cdot y_e[k] - w_d[k] \cdot y_e[k+1]) \cdot T_s] \quad (1.38)$$

$$\omega[k] = \frac{1}{v_r[k] \cdot \cos\theta_e + k_0 \cdot \text{sgn} y_e[k+1]} \cdot [-(1 - q_2 \cdot T_s) \cdot s_2[k] + \varepsilon_2 \cdot T_s \cdot \text{sgn}(s_2[k]) - \theta_e[k] - y_e[k+1] \cdot (k_2 + 1) - (v_r[k+1] \cdot \sin\theta_e[k] + w_d[k+1] \cdot x_e[k] - w_d[k] \cdot x_e[k+1]) \cdot T_s)] + w_d[k] \quad (1.39)$$

1.4. Comunicația și conducerea WMRs 2DW/1FW, Pioneer 3-DX și 2DW/2FW, PatrolBot, utilizând pachete software dedicate

Comunicarea cu simulatorul se realizează cu ajutorul funcțiilor ARIA, [95], produs al firmei MobileRobots este un mediu de dezvoltare de tip open-source implementat în limbajul C++, care asigură o interfaţă client, robustă, către o mare varietate de sisteme robotice inteligente. ARIA are la bază programarea orientată pe obiecte şi poate fi utilizată atât sub UNIX cât şi Microsoft Windows, conţinând de asemenea un API cu clase şi metode orientate spre obiecte proiectat şi scris în întregime pentru a fi flexibil şi uşor de înţeles.

Softul Aria realizează conectarea automată la simulatorul MobileSim în cazul în care nu este detectat nici un robot conectat la portul COM1. Simulatorul are implementate modelele cinematice ale WMRs, funcții pentru simularea sonarelor și laserelor, care sunt folosite pentru a simula comportamentul unui WMR real. Programul scris în C++ apelează funcțiile ARIA în cazul în care se dorește trimiterea unor comenzi către simulator sau citirea datelor simulate. Protocolul de comunicaţie dintre robotul mobil şi un controller exterior se desfaşoară pe o linie serială, într-o structura *client-server,* în care WMR este *server* şi controllerul este *client*. Pentru a elimina dezavantajul principal al folosirii cablului serial putem folosi o interfaţă wireless folosind două modem-uri radio.

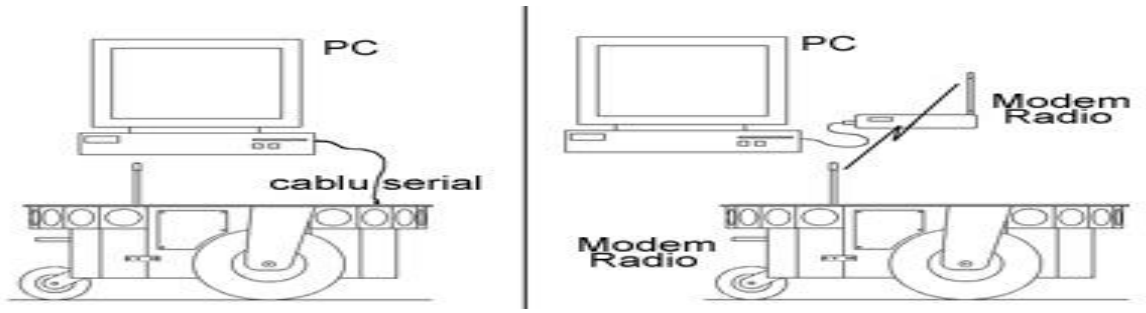


Fig.1.5. Tipuri de conexiune pe linie serială folosind un computer tip desktop

Pentru a elimina dezavantajele folosirii unui computer tip desktop în cele două cazuri din Fig.1.5 putem folosi un computer tip notebook sau varianta cea mai bună din punct de vedere al autonomiei, un computer la bordul robotului (onboard computer) aşa cum se poate observa în Fig.1.6.

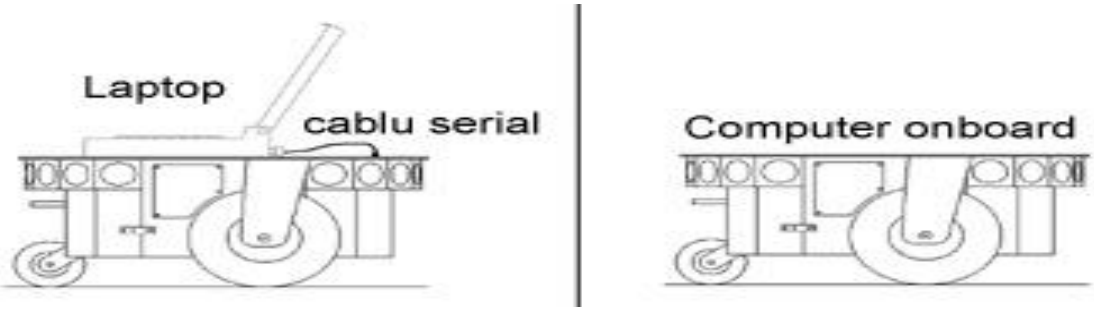


Fig.1.6. Conexiune cu computer notebook (stânga). Robot cu computer on board (dreapta)

După terminarea secvenţei iniţiale de conectare, care constă în transmisia a trei pachete de la client către server (SYNC0, SYNC1 şi SYNC2) robotul ia în considerare pachetele de comandă emise de client şi trimite permanent pachete cu informaţii de stare, la intervale de 100ms, [25], [82]. Pachetele de comandă transmise către robot au următoarea structură generală:

Tab. 1.1. Structura pachetelor de comandă

$FA	$FB	Count	Opcode	Type	Param	Param	CRCH	CRCL

Octeţii $FA-$FB - secvenţă de sincronizare cu care începe fiecare pachet;
Count – numărul total de octeţi ai pachetului minus 3 (nu se iau în considerare cei doi octeţi de sincronizare şi nici octetul count);
Opcode – indică tipul comenzii;

Tab. 1.2. Principalele opcode-uri valide

Opcode	**Denumire**	**Semnificaţie**
0x04	ENABLE	Activează/dezactivează motoarele în funcţie de parametrul specificat (0/1).
0x06	SETV	Setează viteza de translație maximă.
0x07	SET0	Defineşte originea sistemului odometric în punctul curent.

0x08	MOVE	Mişcare de translaţie înainte sau înapoi pe distanţa specificată în parametru (în mm).
0x0B	VEL	Translaţie cu viteza specificată de parametru,mm/s. Înainte dacă parametrul este pozitiv, înapoi dacă este negativ.
0x0C	HEAD	Rotire la unghiul absolut specificat de parametru.
0x1D	STOP	Oprire. Motoarele rămân în starea *enabled.*
0x20	VEL2	Specifică viteza pentru fiecare roată în parte

Datorită faptului că arhitectura sistemului este una deschisă, dezvoltatorii care vor să creeze propriile rutine pentru controlul low level a WMRs nu mai au munca îngreunată cu controlul hardware sau comunicarea la nivel de cod maşină. Comunicaţia cu robotul se bazează pe o relaţie de tip client-server folosind fie o conexiune serială pentru robot (tip RS232), fie una TCP/IP (pentru a comunica cu un simulator robotic).

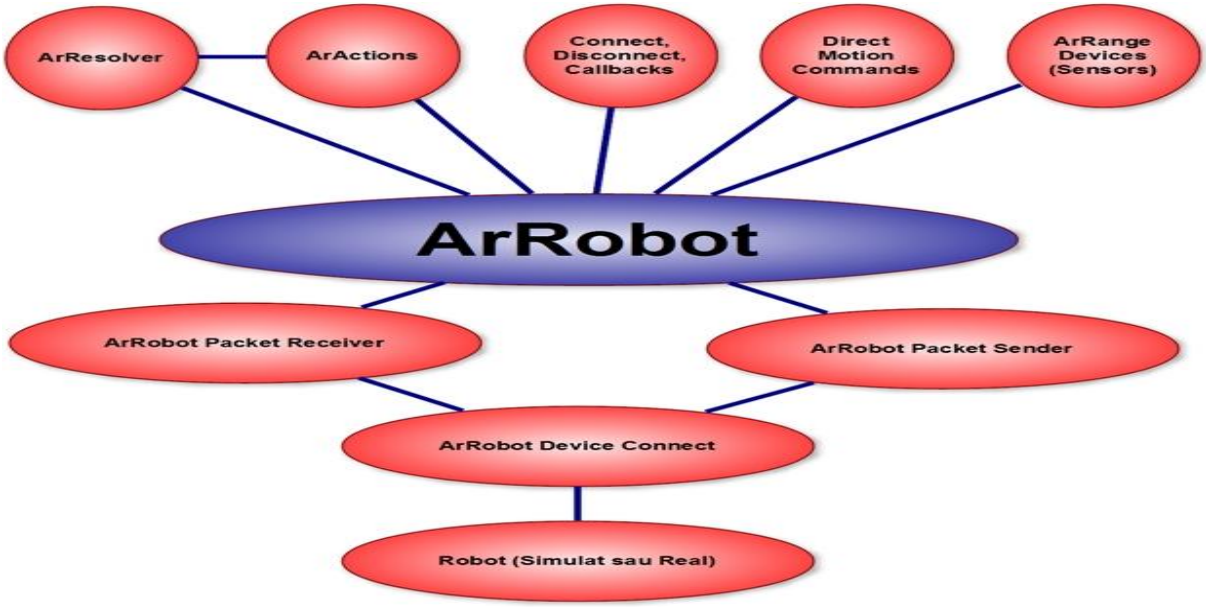


Fig.1.7. Clase ale API-ul ARIA;

“Inima” API-ului ARIA este clasa *ArRobot*. Această clasă are rolul de a controla ciclul de comunicaţie cu firmware-ul robotului**,** de a primi şi a permite accesul la datele despre starea curentă a platformei robot, de a porni sarcini şi de a determina comenzile ce vor fi trimise înapoi către robot. Prin infrastructura clasei *ArActions* ARIA pune la dispoziţie un puternic mecanism pentru combinarea comportamentelor independente pentru a obţine un control coordonat al mişcării şi ghidare inteligentă. Cu ajutorul clasei *ArActions* se pot implementa cu uşurinţă aspectele ce ţin de mişcare folosite în aplicaţii de genul: operare de la distanţă, navigare autonomă, recunoaştere vizuală, etc. Celelalte clase ARIA ne pun la dispoziţie interfeţe pentru a accesa şi controla senzorii şi alte dispozitive cu care ar putea fi dotat robotul, cum ar fi controlul braţului (în 3 sau 5 puncte), a sonarelor şi a laserului.

1.4.1. Comunicația cu robotul

Una dintre cele mai importante funcţii ale API-ului ARIA şi unul dintre primele lucruri pe care programul trebuie să-l facă este stabilirea unei conexiuni între instanţa obiectului *ArRobot* şi sistemul de operare al platformei robot (firmware). În plus, faţă de WMR, unele accesorii cum ar fi sonarele, braţul mobil, camerele PTZ, şi altele sunt conectate intern la microcontroller-ul

robotului sau la liniile digitale de I/O şi folosesc, de asemenea, conexiunea robotului, astfel interfeţele claselor pentru aceste obiecte au nevoie de o referinţă către un obiect *ArRobot* care trebuie să fie conectat pentru ca aceste dispozitive să funcţioneze.

1.4.2. Pachetele SIP

Pachetele de informaţii server (SIPs Server Information Packets) sunt pachetele trimise de WMR server conţinând informaţii actualizate despre robot şi accesoriile sale. SIP-ul standard este trimis de WMR către un client conectat, la fiecare 100ms (intervalul de timp poate fi configurat cu ajutorul parametrilor firmware-ului). Aceste pachete conţin: poziţia curentă a WMR, translaţia curentă şi vitezele de rotaţie, citiri actualizate ale sonarelor, voltajul bateriei, stările I/O analogice şi digitale.

1.4.3. Pachete de comandă

Pentru a conduce platforma robot, un program client trebuie să trimită pachete de comandă folosind conexiunea către robot. Această acţiune se poate face folosind funcţiile de comandă ale mişcării din clasa *ArRobot*, folosind clasa *ArAction* sau la nivel de bază folosind comenzi directe. Folosirea oricărei dintre metodele enumerate anterior conduce la trimiterea de pachete de comandă către robot.

1.4.4. Ciclul de sincronizare a robotului

SIP-urile standard sunt trimise într-un ciclu constant, primirea unui SIP declanşând o nouă iteraţie a ciclului de procesare a unei sarcini al clasei *ArRobot*. Acest ciclu constă într-o serie de sarcini sincronizate, incluzând procesarea pachetelor SIP, sarcini de interpretare a senzorilor, etc. Având în vedere faptul că ciclul sarcinilor este (în mod normal) declanşat de primirea fiecărui SIP (asta în afară de cazul în care platforma robot eşuează în trimiterea SIP-urilor sau ciclul sarcinilor este explicit disociat de conexiunea robotului), fiecare sarcină va fi procesată într-o ordine stabilită, având date actualizate pe baza cărora se pot decide acţiuni. Nici o sarcină nu va pierde oportunitatea de a folosi un SIP, atât timp cât execută sarcini. În Fig.1.18 este prezentată o simulare cu doi WMRs, comunicația și conducerea este relizată cu pachetul ARIA, [81], [82].

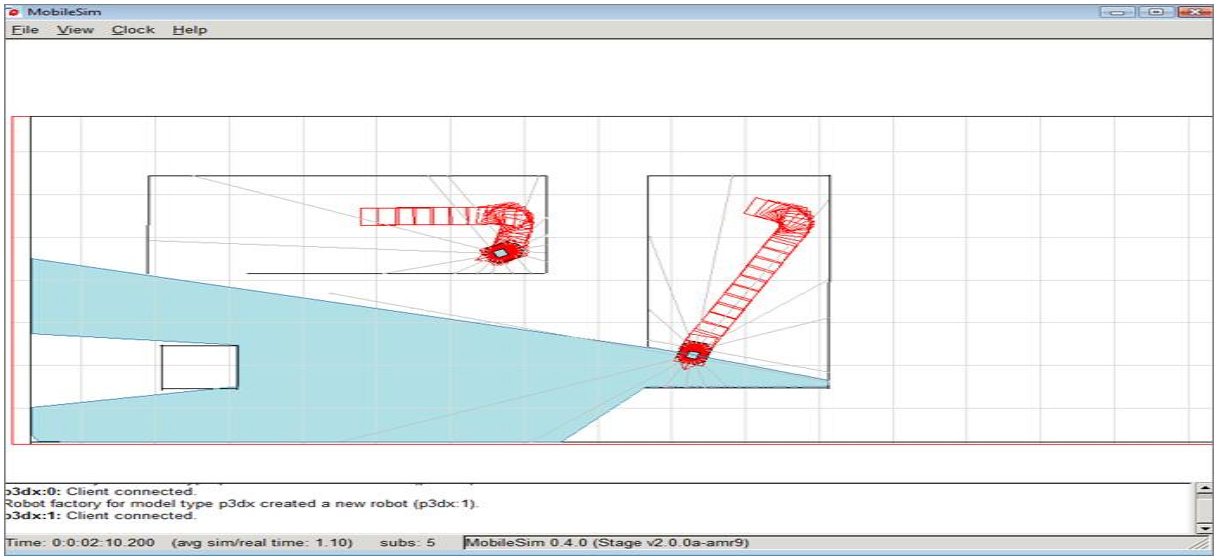


Fig.1.8. Simulare cu 2 WMRs, Pioneer 3-DX, în ARIA/MobileSim

1.5. Testarea metodelor de conducere a WMRs, 2DW/1FW Pioneer 3-DX și 2DW/2FW PatrolBot, utilizând pachete software dedicate

Roboții mobili 2DW/1FW Pioneer 3-DX şi 2DW/2FW PatrolBot sunt roboți conduși diferențial și echipați cu sonare. Robotul mobil Pioneer 3-DXeste echipat cu un braț articulat cu 5 grade de libertate și un gripper. Brațul robotului nu este prevăzut cu senzori, iar comanda brațului poate fi realizată doar în buclă deschisă. Fiecare platformă robotică vine echipată cu motoare și controlere, toate controlate de un microcontroller încorporat care acționează ca un server și software client pentru roboții mobili. Dezvoltarea de software include ARIA, [95] și ArNetworking, [94]. Roboții sunt prevăzuți cu o conexiune serială RS232 pentru comunicația cu exteriorul, iar conectarea la calculator se realizează utilizând o conexiune wireless, utilizând un access point wireless și un echipament server universal pentru conversia de la protocolul RS232 la protocolul Ethernet. Programul este scris în Visual C++ și rulat pe un PC cu o frecvență de eșantionare de 100 ms. În continuare sunt prezentate simulări și implementări în timp real pentru a valida metodele propuse. Pentru a trimite comenzile calculate de algoritmii de conducere s-au folosit funcțiile din ARIA: setVel(Velocity), setVel2(leftVelocity,RightVelocity), setRotVel (Angular Velocity). S-au citit datele de la robot cu ajutorul funcțiilor ARIA: getX(), getY(), getTH(), getRotVel(), getVel(). Structurile sliding-mode, propuse pentru conducerea celor doi WMRs au fost testate în MobileSim [94]. Programul este scris în C++ (Anexa 1) și compilat în Visual Studio, [98].

1.6. Simularea în bucla închisă, conducerea SM-TT a WMR 2DW/2FW, PatrolBot

Urmărirea unei traiectorii liniare urmată de o traiectorie în formă de ”S” a fost testată pentru WMR PatrolBot utilizând conducerea sliding-mode cu timp discret, (1.38), (1.39), [78], [79], [80]. Parametrii constanți folosiți în acest experiment au fost: $q_1 = 0.9$, $q_2 = 0.9$, $\varepsilon_1 = 0.01$, $\varepsilon_2 = 0.75$, $k_0 = 9.5$, $k_1 = 0.75$, $k_2 = 15$. Acești parametri au fost obținuți prin identificare în urma unor simulări succesive utilizând diferite valori ale parametrilor. Arhitectura de conducere folosită este prezentată în Fig.1.9. Modulul Trajectory Planner constă, în acest caz, în impunerea vitezei liniare de 0.4m/s și vitezei unghiulare de 0 rad/s timp de 10s, apoi impunerea vitezei unghiulare de 0.2 rad/s timp de 15s și în final impunerea vitezei unghiulare de -0.2 rad/s timp de 15s. Modulul Odometry este implementat de softul ARIA și viteza liniară, viteza unghiulară, poziția și orientarea WMR sunt obținute utilizând funcțiile ARIA corespunzătoare. Pentru implementarea conducerii sliding-mode cu timp discret am elaborat şi executat programul prezentat în Anexa 1.

În Fig.1.10 este prezentată simularea în MobileSim a traiectoriei WMR-ului PatrolBot utilizând conducerea sliding-mode trajectory tracking (SM-TT) cu timp discret. În Fig.1.11, este prezentată traiectoria simulată în MobileSim a conducerii sliding-mode în timp discret a robotului Patrolbot și cu o linie continuă roșie și traiectoria dorită cu o linie întreruptă albastră. În Fig.1.12, Fig.1.13 și Fig.1.14, sunt prezentate eroarea pe axa Ox, eroarea pe axa Oy, respectiv eroarea de orientare în urmărirea traiectoriei impuse. În Fig.1.15 și Fig.1.16, sunt prezentate suprafețele de comutație s_1, respectiv s_2 în urmărirea traiectoriei impuse.

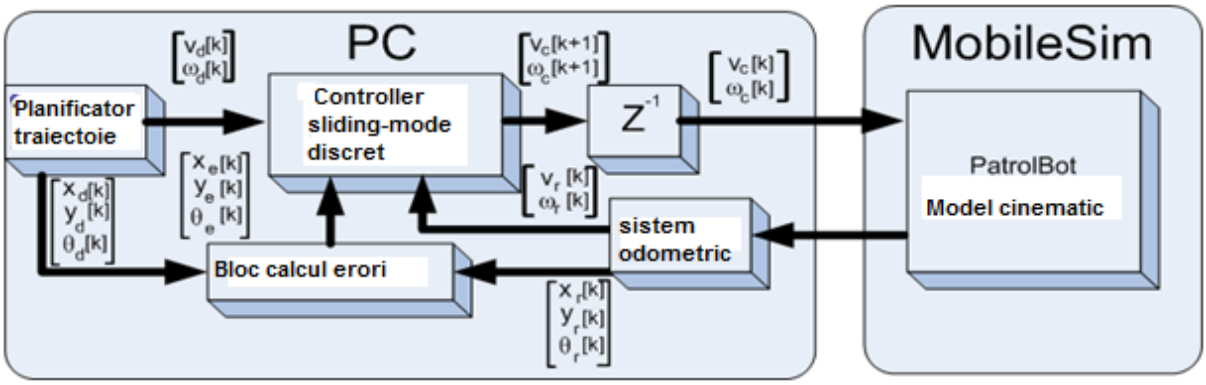


Fig.1.9. Arhitectura de conducere a WMR 2DW/2FW, PatrolBot, în MobileSim

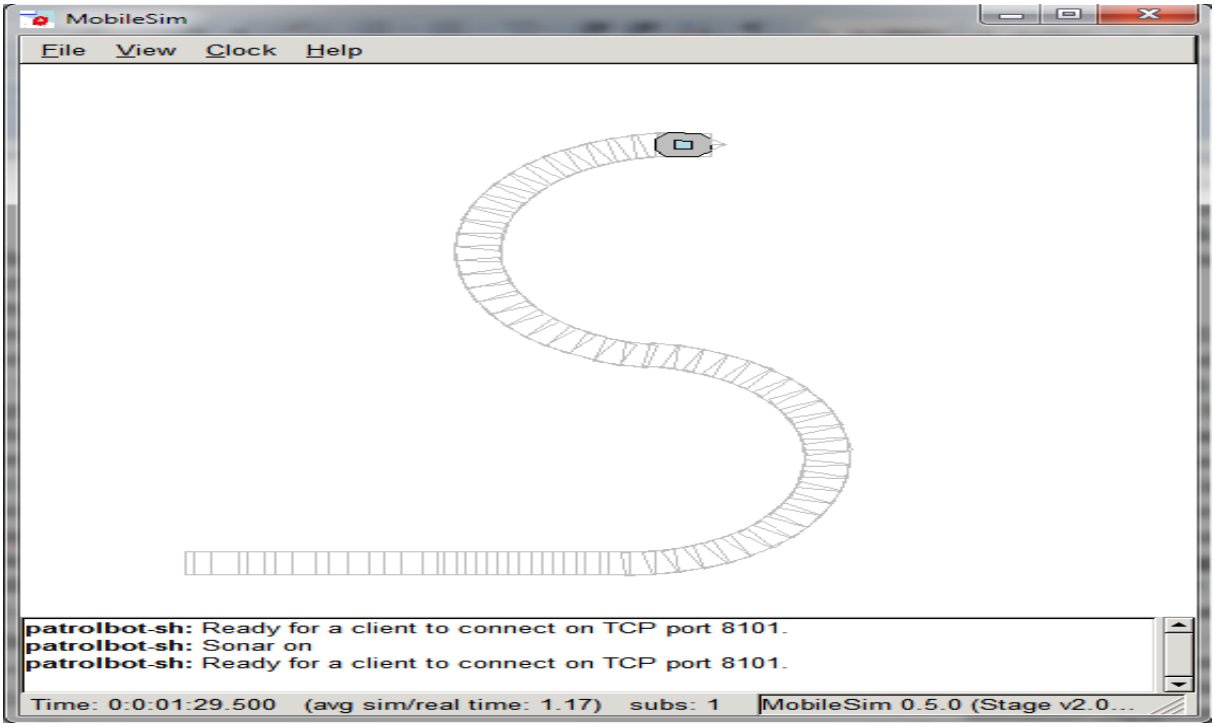


Fig.1.10. Traiectoria în MobileSim la conducerea SM-TT, cu timp discret a WMR PatrolBot

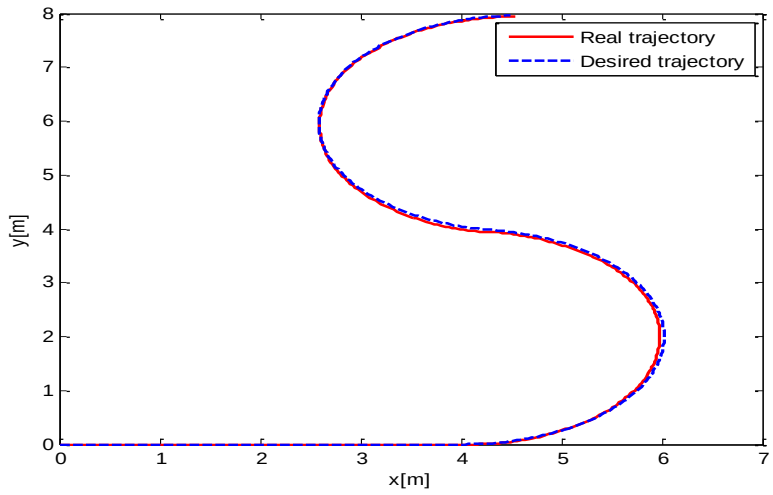


Fig.1.11. Traiectoria în MobileSim și traiectoria impusă la conducerea SM-TT, cu timp discret a WMR PatrolBot

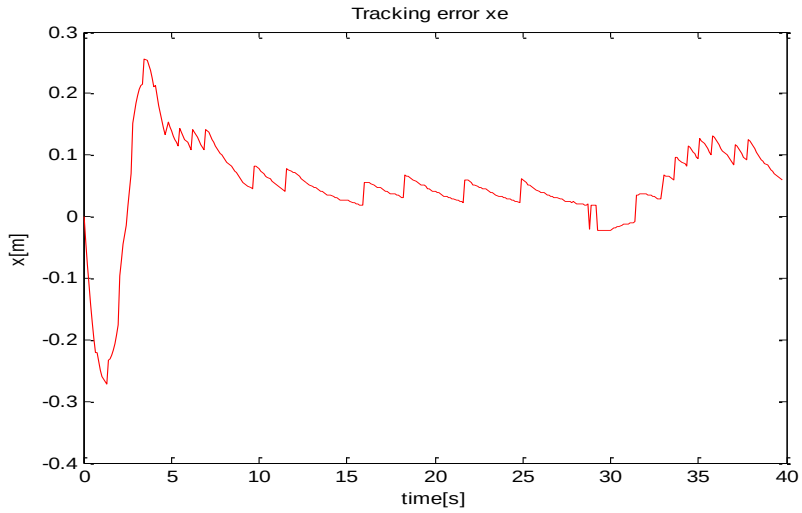


Fig.1.12. Eroarea de urmărire pe axa X în MobileSim la conducerea SM-TT, cu timp discret a WMR 2DW/2FW PatrolBot

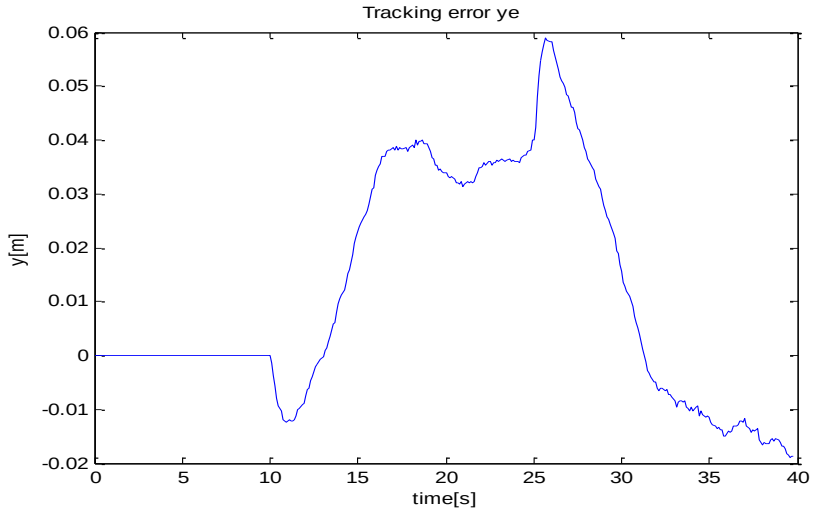


Fig.1.13. Eroarea de urmărire pe axa Y în MobileSim la conducerea SM-TT, cu timp discret a WMR PatrolBot

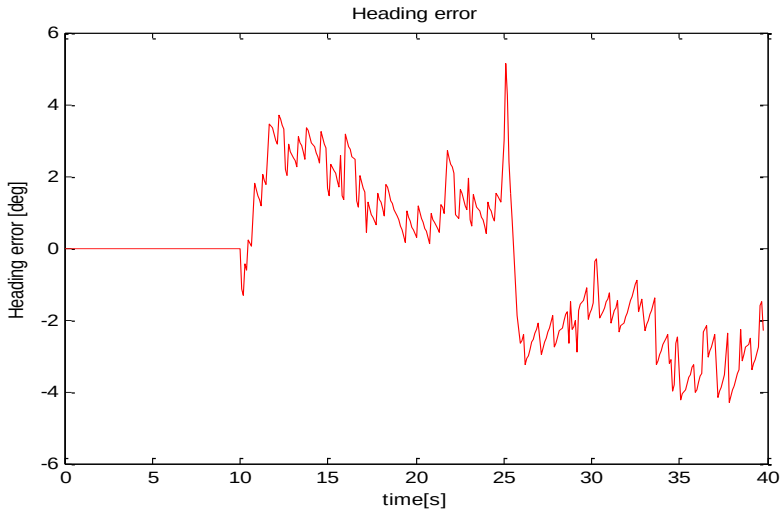


Fig.1.14. Eroarea de orientare în MobileSim la conducerea SM-TT, cu timp discret, a WMR PatrolBot

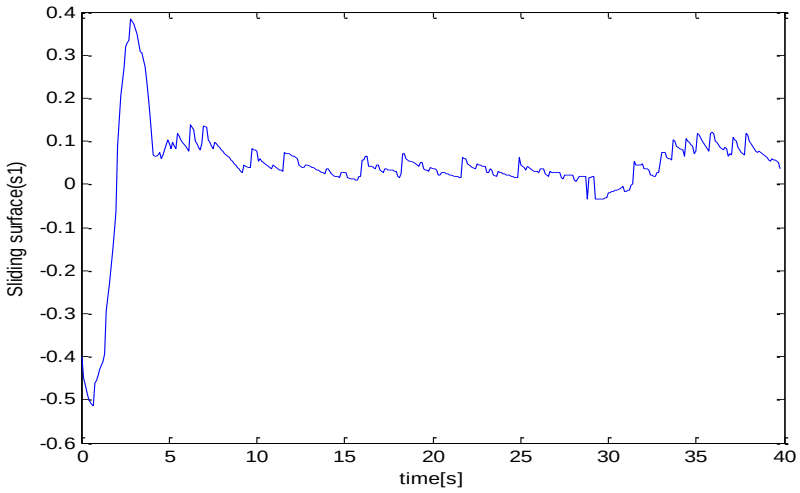


Fig.1.15. Suprafața de comutație, s_1, în MobileSim la conducerea SM-TT, cu timp discret, a WMR PatrolBot

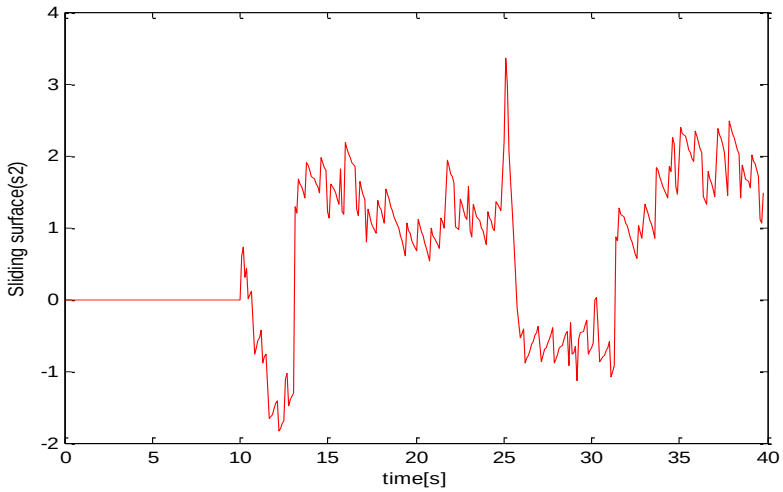


Fig.1.16. Suprafața de comutație, s_2, în MobileSim la conducerea SM-TT, cu timp discret, a WMR PatrolBot

Din figurile prezentate se observă că robotul urmărește traiectoria dorită cu erori mici.

1.7. Rezultate de simulare în buclă închisă la conducerea SM-TT cu timp continuu a WMR 2DW/1FW, Pioneer 3-DX

În această parte sunt reprezentate grafice rezultatele simulării, în buclă închisă, privind conducerea sliding mode cu timp continuu din (1.19), (1.20) a WMR 2DW/1FW Pioneer 3-DX. Schema bloc în Simulink a structurii de conducere este în Fig.1.17.

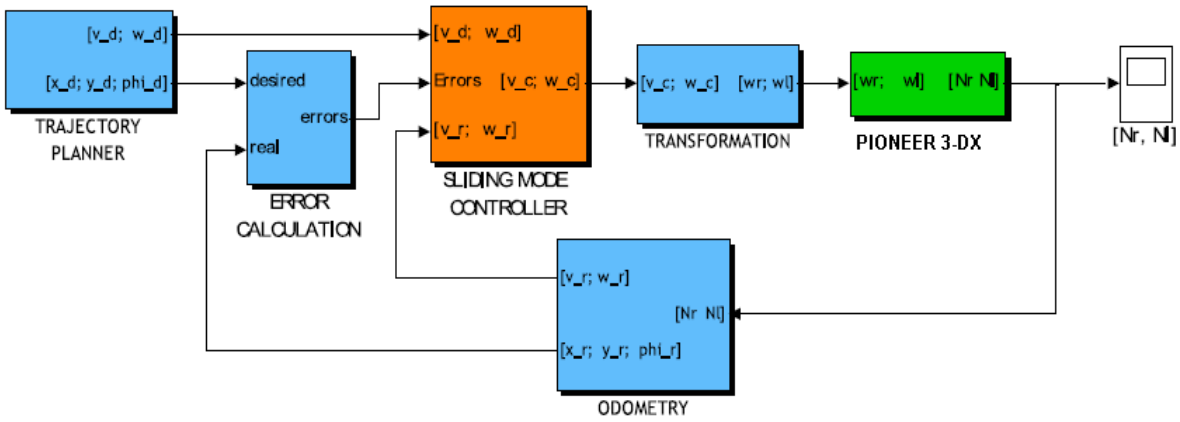


Fig.1.17. Schema bloc de simulare în buclă închisă la conducerea SM cu timp continuu a WMR 2DW/1FW Pioneer 3-DX

Scopul a fost să se obţină erorile de urmărire a treiectoriei cât mai mici. S-a testat pe mai multe tipuri de traiectorie: pătrat, cerc și curbă deschisă. S-au utilizat funcțiile din ARIA, iar programul de conducere, atât în regim de simulare, cât și în timp real, este scris în Visual C++, Anexa 1. Întâi, s-a testat conducerea SM în buclă închisă utilizând modelul robotului Pioneer 3-DX în MobileSim. În cele ce urmează este prezentat răspunsul simulat al WMR la conducerea SM-TT, uitilizând trei tipuri de traiectorie, liniară inchisă (pătrat), curbilinie închisă (cerc) și curbilinie deschisă. Secvența de cod din programul sursă, scris în limbajul C++, utilizează legea de comandă, în care:

temp1 = -Q1*satur(s_1/0.5) - gama_x*x_e_der - (aw*y_e) - w*y_e_der + av;

temp2 = -Q2*satur(s_2/0.5) - gama_y*y_e_der + (aw*x_e) + w*x_e_der;

Pentru început, s-a urmărit cum se comportă WMR la urmărirea unei traiectorii liniare închisă. In Fig.1.18, este captată imaginea din MobileSim în momentul în care robotul reușește să ajungă în punctul din care a plecat la începutul simulării. În Fig.1.19, cu ajutorul pachetului Matlab, [96], s-au reprezentat grafic erorile laterale x_e şi y_e, eroarea unghiulară, θ_e, vitezele liniară și unghiulară.

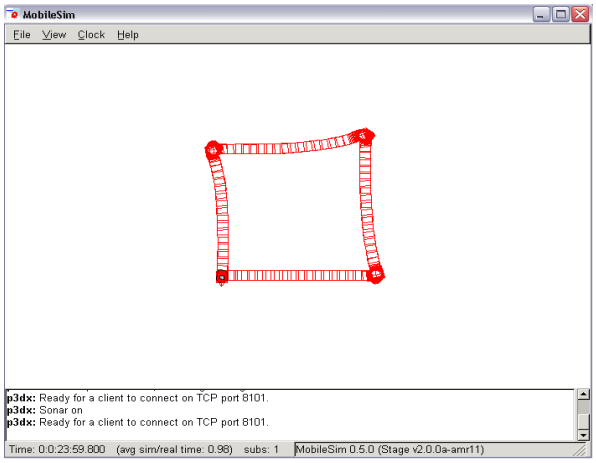


Fig.1.18. Traiectorie pătrat în MobileSim

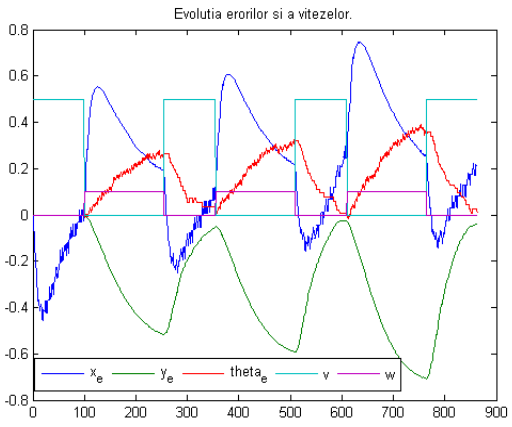


Fig.1.19. Erori și viteze

Se observă o curbare a laturilor pătratului, dar robotul ajunge în punctul de plecare. În Fig.1.20 si Fig.1.21 s-au reprezentat sub formă de histograme erorile x_e, y_e, θ_e, viteza liniară, v, viteza unghiulară, w, dar și derivatele: $\dot{x}_e$, $\dot{y}_e$ si $\dot{\theta}_e$.

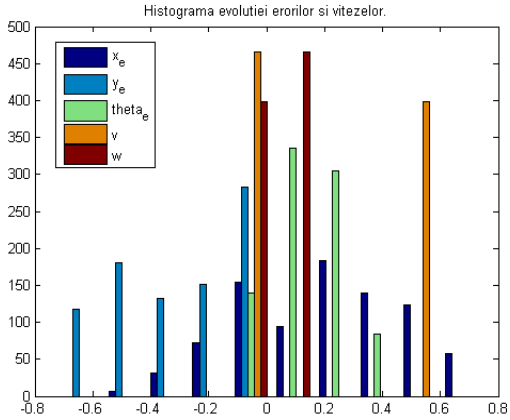


Fig.1.20. Histograme erori și viteze

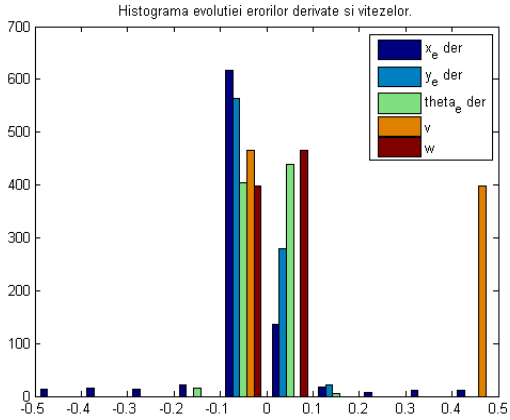


Fig.1.21. Derivate erori și viteze

S-a testat, apoi, comportarea WMR în conducerea SM-TT, pe traiectorie circulară.

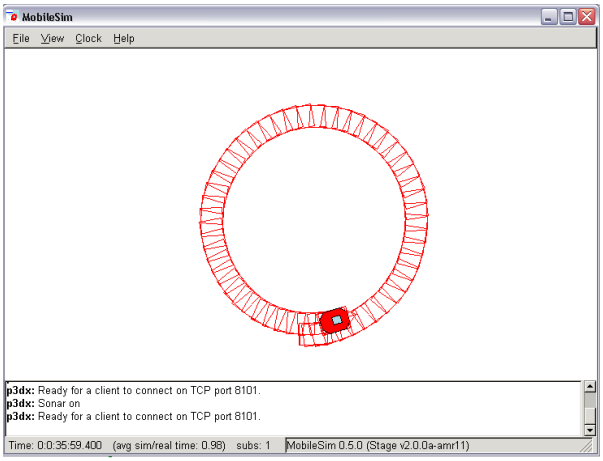


Fig.1.22. Traiectorie circulară

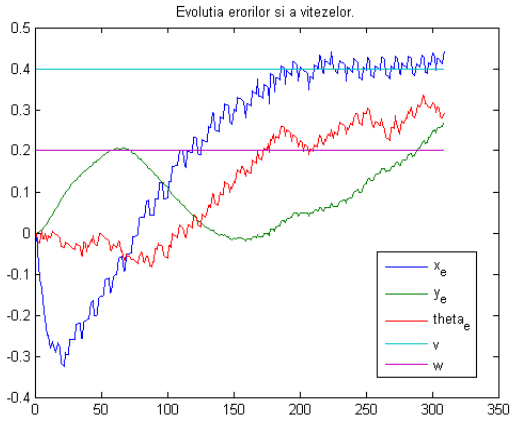


Fig.1. 23. Erori și viteze

Se observă în Fig.1.22, de mai sus, că punctul de plecare nu coincide cu punctul de sosire, ș constatându-se o creştere a erorilor x_e , y_e și θ_e, Fig.1.23.

În Fig.1.24 şi 1.25 s-au reprezentat sub forma de histograme erorile x_e , y_e , θ_e, viteza liniară, v , viteza unghiulară, w , dar și derivatele: $\dot{x}_e$, $\dot{y}_e$, $\dot{\theta}_e$.

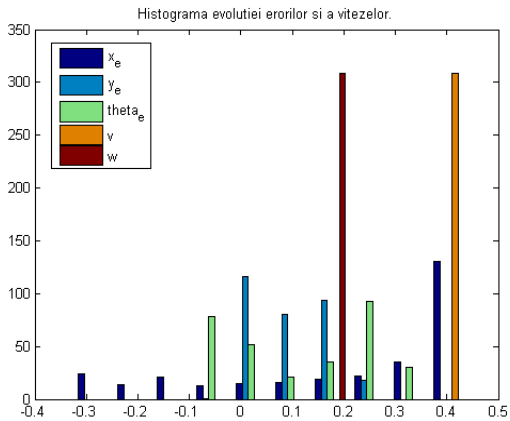


Fig.1. 24. Histograme erori și viteze

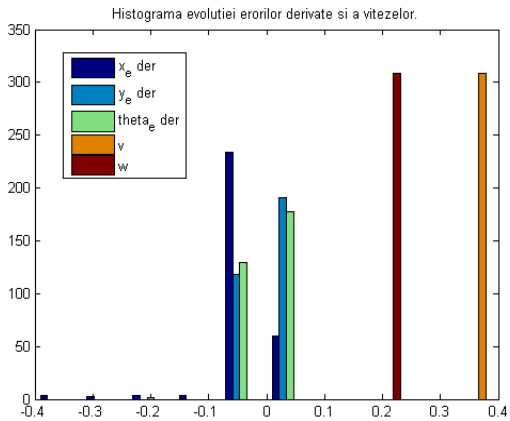


Fig.1. 25. Derivate erori și viteze

O altă traiectorie utilizată în conducerea SM-TT a WMR Pioneer 3-DX robotului a fost curbă deschisă. Aceasta traiectorie a fost obţinută prin cuplarea a două semicercuri. La programul scris în C++, robotului îi este specificat să urmeze o traiectorie în formă de semicerc, timp de 155 de iterații, având viteza liniară egală cu 0.4 m/s iar viteza unghiulară de 0.2 rad/s iar după durata a 155 iterații, îi este impusă o viteza unghiulară de -0.2 rad/s. În Fig.1.26 este captată imaginea din MobileSim, în Fig.1.27 sunt reprezentate erorile și vitezele, iar în Fig.1.28 și 1.29 sunt reprezentate histogramele erorilor, vitezelor și derivatele erorilor.

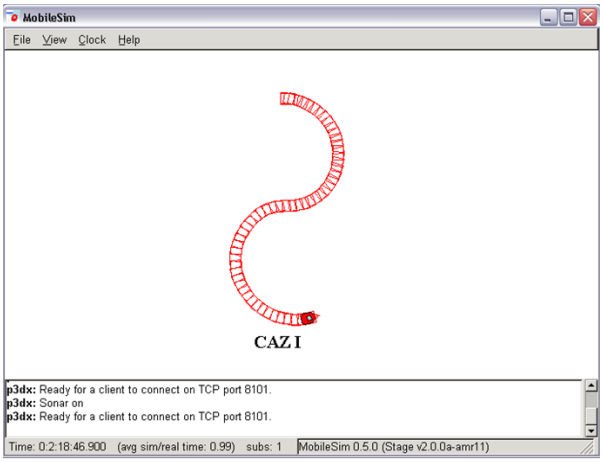


Fig.1. 26. Traiectorie curbă deschisă

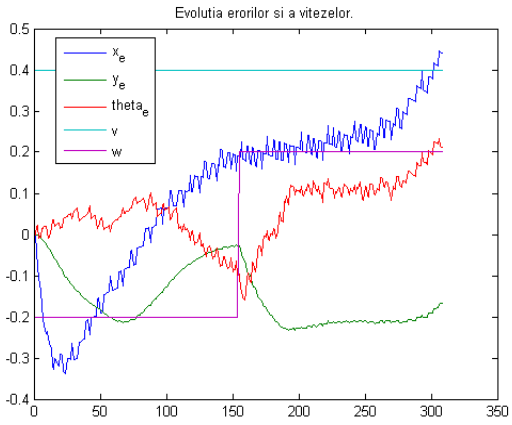


Fig.1. 27. Erori și viteze

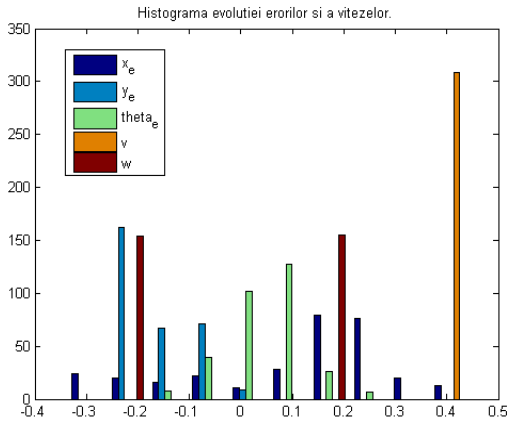


Fig.1. 28. Histograme erori și viteze

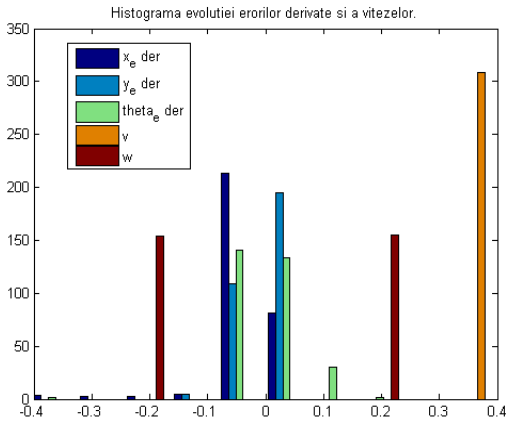


Fig.1. 29. Derivate erori și viteze

1.8. Conducerea SM-TT, în timp real, a WMR 2DW/1FW, Pioneer 3-DX

În acest subcapitol, sunt reprezentate grafic rezultatele implementării metodei de conducere SM, cu timp discret, în timp real, a robotului mobil Pioneer 3-DX. În Fig.1.30 este prezentată schema bloc de conducere în buclă inchisă, iar în Fig.1.31 este prezentat modelul Simulink de acţionare a roţilor motoare. La conducerea SM în timp real a WMR Pioneer 3-DX, s-au comparat erorile obținute cu cele simulate. În cazul traiectoriei în formă de pătrat s-au reprezentat grafic erorile în Fig.1.32. La conducerea SM pe linie curbă deschisă s-au obținut erorile reprezentate grafic în Fig.1.33.

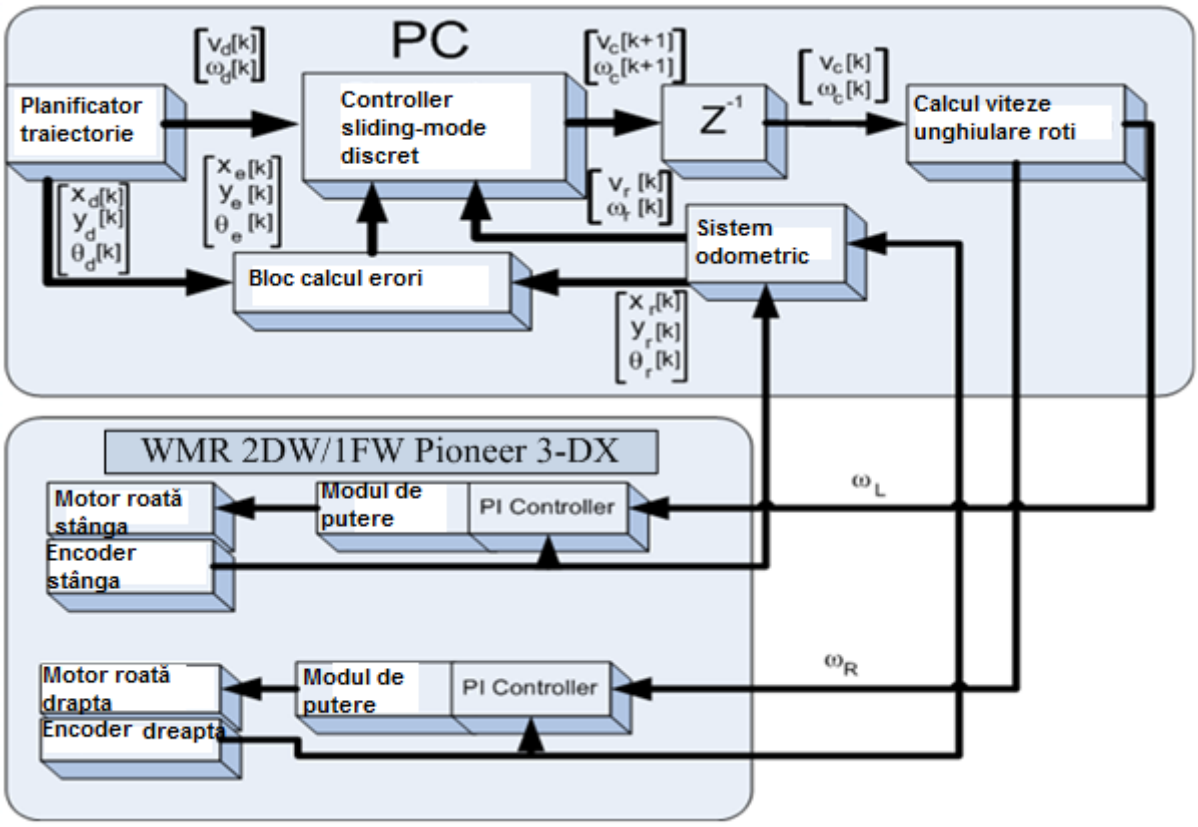


Fig.1.30. Structura de conducere SM în timp real

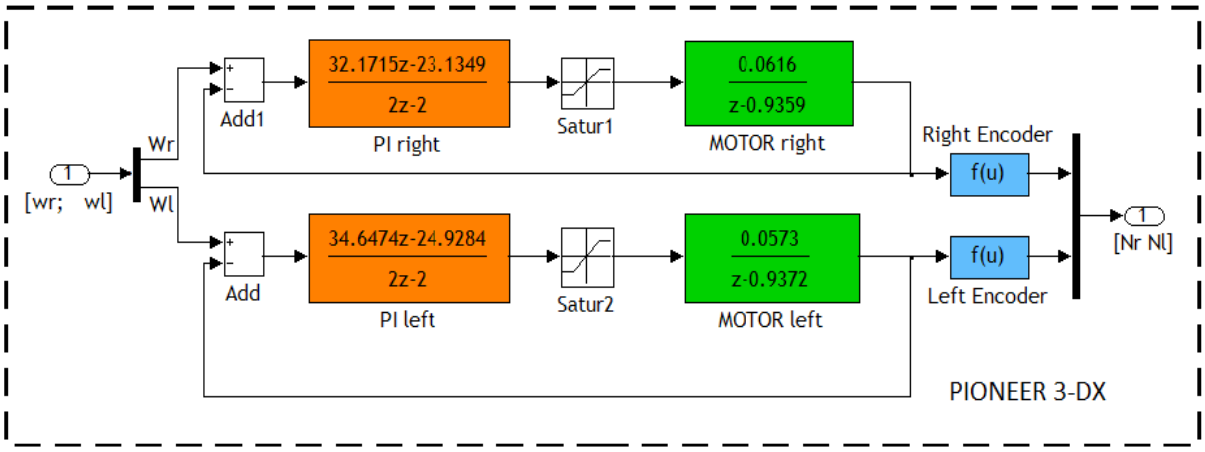


Fig.1.31. Modelul Simulink corespunzător acționării roților motoare ale WMR, 2DW/1FW Pioneer 3-DX

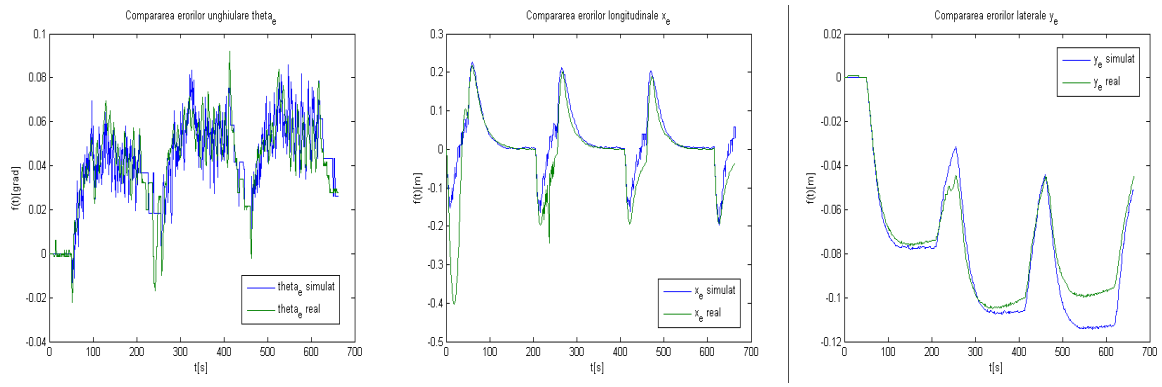


Fig.1.32. Erorile de poziție și de orientare, în timp real și MobileSim, traiectorie pătrat

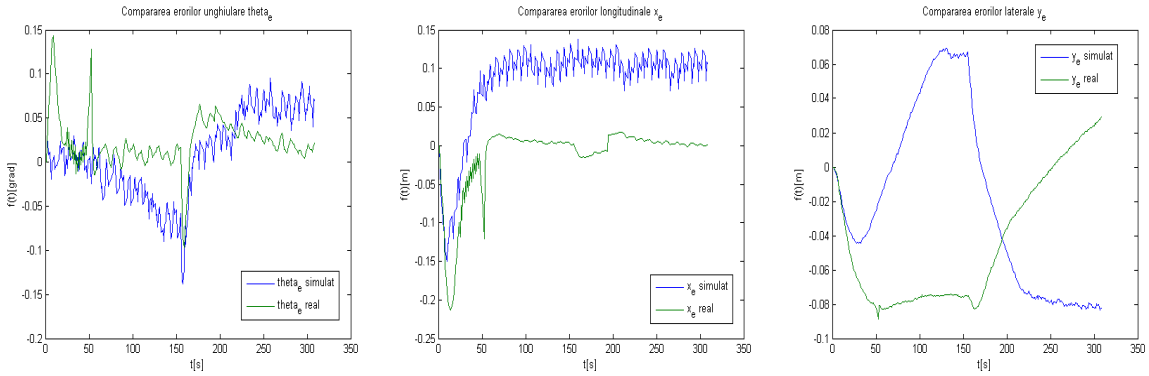


Fig.1.33. Erorile de poziție și de orientare, în timp real și MobileSim, traiectorie curbă deschisă

1.9. Conducerea în buclă deschisă a RM Pioneer 5-DOF Arm

Manipulatorul robotic Pioneer 5-DOF Arm, Fig.1.34, are 6 articulații comandabile prin motoare pas cu pas. Plasat pe un WMR, capătă o cinematică redundantă, cu 6 grade de libertate (6-DOF). S-a utilizat pe WMR 2DW/1F, Pioneer 3-DX, deservind liniile de mecatronică A/DML și P/RML, atât în manipulare, cât și la transport. Nu are prevăzute encodere în articulații. Prin urmare, poate fi comandat numai în buclă deschisă. Datorită acestui fapt, nu poate executa operații de manipulare cu precizie, fiind nevoie de sistem servoing vizual, fix sau mobil, acesta din urmă fiind plasat pe ultimul braț al manipulatorului. Se utilizează funcția ARIA arm.moveTo(X,Y,Z) cu trei argumente. Primul argument este numărul articulației. Al doilea este unghiul de mișcare în grade sexagesimale, care poate fi negativ sau pozitiv. Al treilea este viteza de mișcare. Pentru parcarea manipulatorului se utilizează funcția arm.park. S-au mai utilizat funcțiile: arm.getJoint, arm.moveToTicks, arm.requestStatus.

1.10. Conducerea în buclă închisă a RM 7-DOF, Cyton 1500

RM Cyton 1500, Fig.1.35, are 7-DOF și poate manipula sarcini de pâna la 1,5 kg, cu o anvergură a braţului de 68 cm. Gradul de libertate suplimentar îi permite o mișcare fluidă, o poziţionare precisă, cu ocolire de obstacole. Are encodere în articulaţii ceea ce îi permite comanda în buclă închisă, suportând drept comenzi, momente active în articulaţii de valori crescute. Astfel, poate apuca şi manipula sarcini mari în condiţii de sigurantă şi fiabilitate. Este prevăzut cu o interfaţă grafică utilizator, care permite poziţionarea 3D precisă prin simpla manevrabilitate cu mouse sau joystick. Poate memora o traiectorie dată prin puncte, a cărei repetabilitate este foarte precisă. S-a utilizat integrat în A/DML, HERA&HORSTMANN, unde are de manipulat sarcini de greutăţi variable în intervalul de până la 1,5 kg.

1.11. Concluzii

În acest capitol, s-a tratat, mai întâi, modelul cinematic al WMRs 2DW/2FW, PatrolBot şi 2DW/1FW, Pioneer 3-DX cât și acționarea și comanda RMs, Pioneer 5-DOF Arm și 7-DOF Cyton 1500. Pentru determinarea modelului cinematic al WMR am considerat variabilele generalizate ale sistemului, rotirea roților robotului fără alunecare și aplicat constrângerile nonholomice specifice acestui caz. Modelul rezultat astfel are cinci variabile: două variabile reprezintă centrul geometric al WMR, o variabilă reprezentând unghiul de direcție al robotului și alte două variabile

reprezentând unghiurile fiecărei roți. Aceste variabile depind de vitezele de rotire ale roților. S-a simplificat acest model pentru că am dorit doar calculul coordonatelor carteziene ale centrului geometric și unghiul de direcție, înlocuind vitezele celor două roți cu viteza liniară și viteza unghiulară a robotului.

Fig.1.34. RM Pioneer 5-DOF Arm

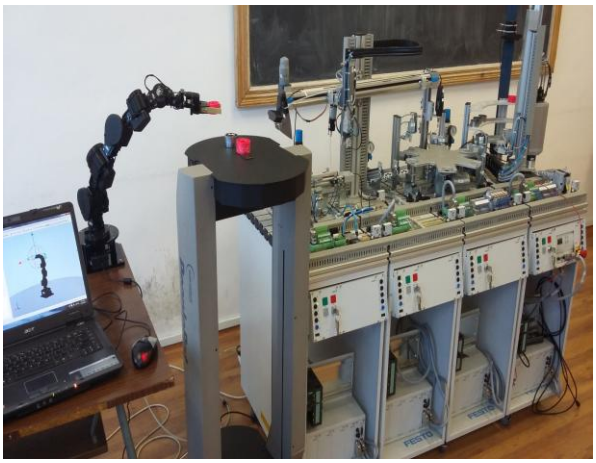


Fig.1.35. RM 7-DOF Cyton 1500

S-au prezentat și analizat două metode pentru conducerea WMRs: conducerea sliding-mode cu timp continuu și conducerea sliding-mode cu timp discret. Folosind modelul cinematic al WMRs cu 2DW/2FW și 2DW/1FW, modelul erorilor de urmărire și dinamica erorilor de urmărire s-au calculat comenzile pentru viteza liniară și viteza unghiulară a WMR.

S-a făcut prezentarea sistemului de comunicație de la distanță și de comandă a WMR cu 2DW/2FW și 2DW/1FW în protocolul client server. Pornind de la modelul cinematic al WMR, 2DW/2FW, PatrolBbot, erorile de urmărire, dinamica erorilor de urmărire, suprafețele de comutație și legea de conducere, toate în timp discret, s-au calculat comenzile pentru viteza liniară și viteza unghiulară a WMR 2DW/2FW, PatrolBot. Utilizând MobileSim s-a testat conducrea în TT a unei traiectorii în formă de curba deschisă (“S”).

Pornind de la modelul cinematic al WMR, 2DW/1FW, Pioneer 3-DX, erorile de urmărire, dinamica erorilor de urmărire, suprafețele de comutație și legea de conducere, toate în timp continuu, s-au calculat comenzile pentru viteza liniară și viteza unghiulară. S-a prezentat schema bloc de acționare a roților și de conducere în buclă închisă cu structura de conducere SM, cu timp continuu. Utilizând MobileSim s-a testat comparativ conducerea TT în regim simulat (care în fapt este o conducere în timp real, aplicată unui model simulat) pentru trei traiectorii, cerc, pătrat și curbă deschisă a WMR 2DW/1FW, Pioneer 3-DX. S-au realizat clipuri video privind conducerea TT, în timp real a celor două platforme robotice mobile, 2DW/2FW, PatrolBot și 2DW/1FW, Pioneer 3-DX împreună cu acționarea și comanda MR, Prioneer 5-DOF.

Rezultatele testelor de simulare și de timp real au validat metodele de acționare și conducere propuse cât și posibilitatea integrării sistemelor robotice în cauză în sistemele de fabricație flexibilă, A/DML și P/RML.

Capitolul 2

Acționarea şi conducerea FMLs, de A/D și de P/R. Particularizare la FMMLs, A/DML și P/RML

În acest capitol, se prezintă structura, organizarea ierarhică, funcționalitățile, fluxurile de date și de informații în liniile de fabricație flexibilă, deservite de sisteme robotice mobile. Sunt făcute referințe bibligrafice la [39], [42], [43], [44], [45]. Tot aici, se prezintă structura, funcționalitatea, acționarea electrică și conducerea liniilor de mecatronică, de fabricație flexibilă deservite de sisteme robotice cu particularizare la A/DML, HERA&HORSTMANN, şi P/RML, FESTO MPS-200.

Detalierea lor se realizează în cadrul a trei secțiuni principale. În prima secțiune se analizează o structură clasică de sistem flexibil de fabricație în conformitate cu literatura de specialitate, urmărind ca, în secțiunea următoare să se particularizeze structura clasică pe procese de asamblare, de dezasamblare și de prelucrare, procese care pot fi deservite de WMR echipat cu RM. Ultima secțiune este dedicată concluziilor prin care se pun în evidență contribuțiile privind analiza unei structuri flexibile de asamblare și deservirea acesteia în cadrul procesului de dezasamblare cu ajutorul un robot mobil echipat cu manipulator.

2.1. Structura unei FML cu roboți integrați

O linie de fabricație flexibilă reprezintă totalitatea stațiilor și celulelor de lucru, echipamentelor de măsură și achizitii date, WMRs, RMs, sisteme de transport, de depozitare, de monitorizare și comandă, capabilă să execute sarcini pentru asamblarea de componente sau operații prelucrare, într-o maniera reconfigurabilă care să confere reversibilitate, repetabilitate și, nu în ultimul rând, flexibilitate, [4], [6], [8]. Conceptul de FML a fost proiectat şi dezvoltat pentru fabricarea de produse diferite, în loturi mici sau medii. În Fig.2.1 este prezentată schema bloc a unei FML, capabilă să execute A/D și P/R deservită de roboți, aceasta fiind alcătuită din următoarele blocuri funcționale, [71], [72], [73]:

- RMs industriale necesare operațiilor de manipulare (necesită precizie, sistem de control al traiectoriei și sistem de senzori și traductoare);
- WMRs pentru transport (necesită sistem de control al traiectoriei sau sistem de ghidare, sistem de senzori de poziție și navigație);
- echipamente de A/D și P/R care echipează RMs industriale, stațiile și celulele de fabricație;
- magazii de depozitare componente și/sau subansamble necesare asigurării unui flux continuu de A/D și P/R. Sunt incluse și magaziile de depozitare componente;
- sistem de transport (benzi transportoare) necesare transportului componentelor sau subansamblelor de la o celulă flexibilă la alta;
- stații și celule de lucru, reconfigurabile, echipate cu utilaje necesare operațiilor de A/D sau P/R;

- senzori și traductoare plasate în rețea distribuită pe FML, pe WMRs și RMs;
- echipamente de monitorizare și comandă în structura distribuită și structura centralizată;
- echipamente de compatibilizare între FML, sisteme robotice și sisteme de calcul;
- echipamente de achiziții date și comunicație.

Pornind de la obiectivele impuse şi de analiză a FML privind realizările teoretice şi experimentale din domeniul industrial se prezintă în acest capitol FMMLs de A/D HERA&HORSTMANN și de P/R FESTO MPS-200 deservite de un WMR 2DW/1FW, Pioneer P3-DX, echipat cu RM Pioneer 5-DOF Arm, sau de doi WMRs.

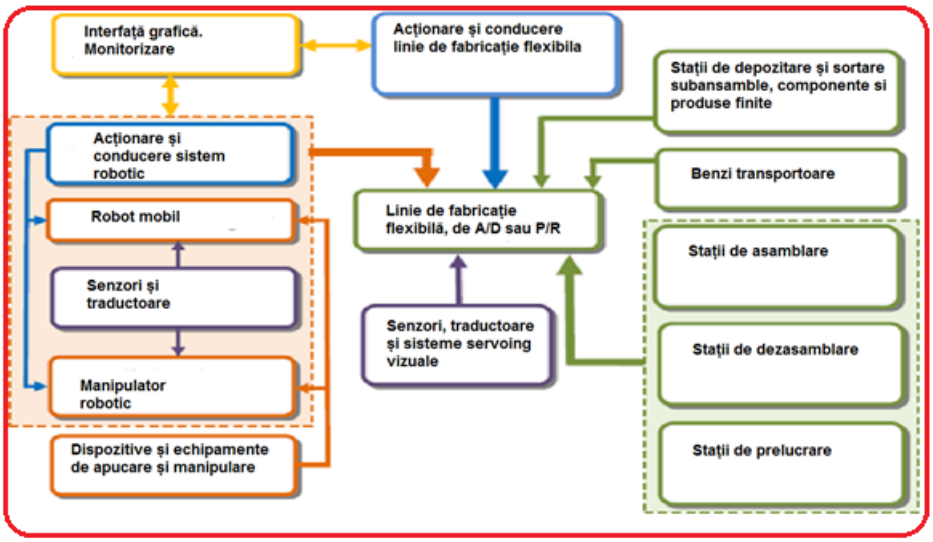


Fig.2.1. Structura unei FML de A/D și de P/R cu roboți integrați

2.2. Organizarea ierarhică a unei FML

Liniile de fabricație flexibilă (FMLs) sunt organizate pe nivele ierarhice, FML-urile inferioare sunt destinate operațiilor primare de A/D sau P/R, pe când FML-urile superioare sunt capabile de A/D și P/R complexe, de mare precizie și eficiență. Până în prezent nu apare un acord unanim în această privință, în continuare fiind prezentată în Fig.2.2 o structură de FML organizată pe patru niveluri, [73]:

- FML1, la acest nivel se găsește cea mai mică unitate autonomă, cu funcții de fabricație de A/D și P/R, aceasta fiind o mașină-unealtă flexibilă, multifuncțională, respectiv un echipament care concentrează un număr mare de operații cu prelucrări diferite, caracterizată de: dotarea cu comandă numerică; posibilități de prelucrare multiple; prezența unui dispozitiv de depozitare scule, dispozitiv care nu influențează procesul tehnologic și la care depozitarea temporară a sculelor este efectuată codificat; schimbarea și transferul automat al sculelor;
- FML2 unde se regăsește *celula de fabricație flexibilă* (FMC) în care sunt concentrate mai multe mașini-unelte cu comandă numerică, deservite de RMs industriali, tot ansamblul fiind conectat la un calculator sau PLC care asigură întreaga funcționare a celulei. Celula flexibilă poate asigura prelucrarea automată integrală a unor produse sau/și piese diferite (componente ale unei clase stabilite în prealabil) având un grad de flexibilitate ridicat;
- FML3 la al cărui nivel se găsesc sistemele flexibile, compuse de regulă din mai multe celule de fabricație flexibilă legate prin dispozitive de transport și manipulare. În cadrul acestor sisteme de la acest nivel se pot deosebi mai multe tipuri de subniveluri (sau

subsisteme). Astfel, pentru sistemul de transport pot fi prevăzute transportoare cu deplasarea semifabricatelor într-o singură direcție în cadrul sistemului, acestea fiind comandate din calculatorul central și, ca urmare, semifabricatele sau piesele sunt paletizate, urmând a fi deplasate la orice stație de prelucrare și în orice ordine, ceea ce deschide largi posibilități de optimizare a funcționării întregului ansamblu. Transportul pieselor poate fi efectuat și prin intermediul cărucioarelor autopropulsate, în multe cazuri acestea sunt comandate prin diverse metode (cabluri pilot amplasate în pardoseală, etc). Preluarea semifabricatelor și pieselor de pe dispozitivele de transport, alimentarea mașinilor-unelte și readucerea pieselor (după prelucrare) pe dispozitivele respective, sunt de regulă realizate de WMRs, RMs industriali și de mecanisme de paletizare (așezare și fixare pe paleți) sau de depaletizare (desprinderea de pe paleți). În cazul sistemelor de la nivelul 3, este prevăzută posibilitatea unor activități suplimentare (în raport cu cele executate la nivelele 1 și 2: manipulare, transport, alimentare, prelucrare, evacuare piese prelucrate, schimarea sculelor, supravegherea instalației etc.) cum sunt cele de testări automate de pregatirea fabricației sau de comandă a aprovizionării cu materiale;

- FML4, unde se pot regăsi totalitatea mijloacelor tehnice și persoanele necesare pentru realizarea aprovizionării, depozitării, planificării de lungă durată, proiectării constructive și tehnologice a produselor și fabricației propriu-zise.

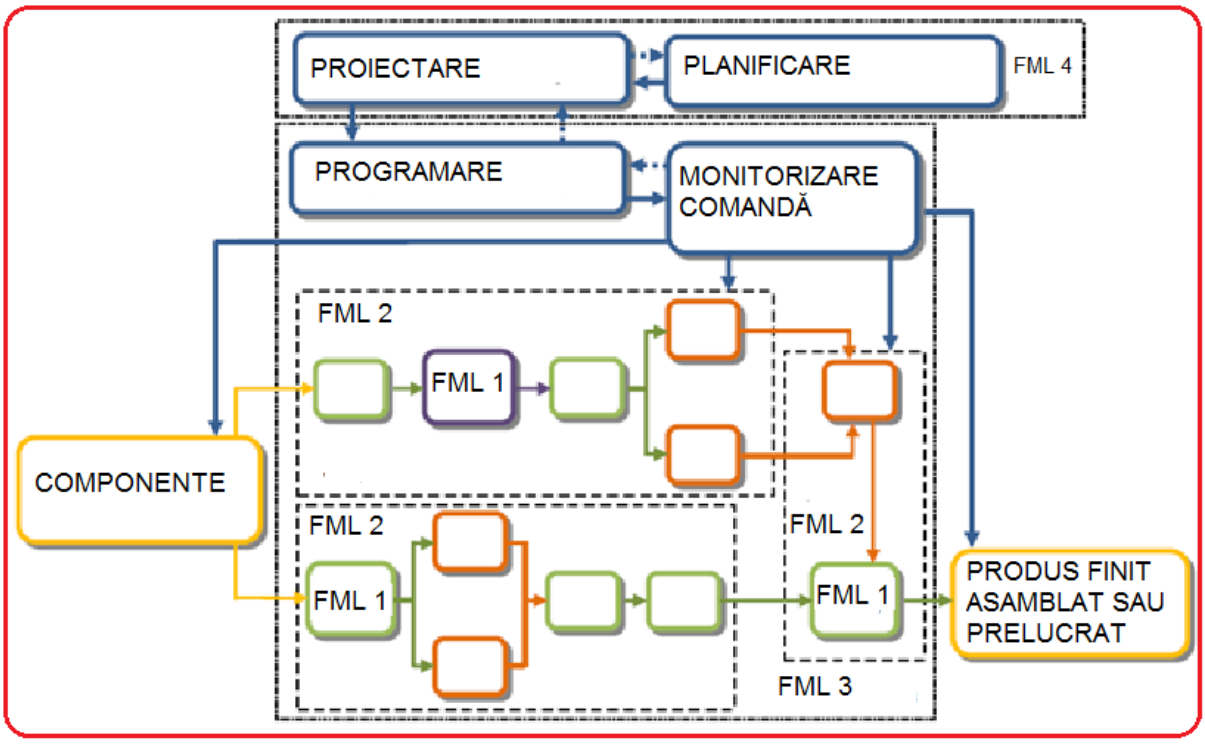


Fig.2.2. Organizarea ierarhică a unei FML

Prezentarea structurilor și nivelelor de organizare are în vedere fluxurile de obiecte (semifabricate, piese, scule) și utilajele necesare prelucrării, manipulării și transportului, în cadrul acestor structuri intervenind și depozitele, dispozitivele de control, de întreținere și reparare etc., care pot fi realizate cu funcționare automată.

2.3. Funcțiile unei FML

Structura generală a unei FML, prezentată în Fig.2.1, permite evidențierea funcțiilor generale ale sistemului:

- funcția de prelucrare automată a pieselor sau subansamblelor;
- funcția de depozitare, transport și manipulare automată;
- funcția de comandă automată a tuturor componentelor sistemului și de supraveghere, control și diagnostic automat, funcție care se realizează cu ajutorul unuia sau mai multor PLC-uri în diverse configurații, centralizat sau distribuit, sau calculatoare de proces ce lucrează în timp real sau unități locale de comandă (echipamente CNC, PLC la sistemele de manipulare și transport, microcalculatoare pentru comanda depozitelor automate, etc). Programele de calculator, furnizează întregului sistem informațiile necesare pentru comanda procesului de prelucrare și pentru comanda producției (comanda depozitelor de piese și scule, comanda sistemului de transport, etc.). Informațiile pentru realizarea acestor subfuncții sunt obținute din sistem cu ajutorul unor traductoare, senzori, aparate de măsură etc. și se transmit în sens invers, către calculatorul de proces, AP, PLC sau microcalculatorului local;
- funcția de prelucrare automată se realizează în cadrul subsistemului tehnologic al FML, având în componență stațiile (celule) de lucru, mijloacele de manipulare a pieselor și sculelor. Realizarea acestei funcții presupune alimentarea automată cu piese și scule a mașinii-unelte, prelucrarea propriu-zisă în comandă numerică și eventual optimizarea procesului de comandă pe mașina-unealtă. Pot fi incluse aici și dispozitivele de asamblare/dezasamblare, unele dintre acestea având funcții speciale;
- funcția de depozitare, transport și manipulare automată se referă la fluxul automat al sculelor, pieselor, componentelor și subansamblelor necesare FML și care includ mai multe funcții parțiale: înmagazinarea automată a pieselor, sculelor, dispozitivelor și materialelor auxiliare; identificarea și livrarea în sistem a piesei sau subansamblelor în mod automat; transportul automat al pieselor, sculelor, dispozitivelor și materialelor auxiliare între depozite și stațiile de lucru. Condiția principală în funcționarea subsistemului de depozitare și transport este ca transferul materialelor să se efectueze totdeauna la locul și momentul potrivit; manipularea pieselor, subansamblelor, sculelor și dispozitivelor în depozite și între stațiile de lucru;
- funcția de comandă, monitorizare, control și diagnostic dintr-un FML este realizată de subsistemul informațional prin fluxul informațional care se transmite în 2 sensuri: sensul direct, al informațiilor de comandă și sensul invers, al informațiilor de monitorizare, control și diagnostic.

2.4. Conducerea unei FML

Structurile și nivelele de organizare sunt puternic corelate cu nivelele de comandă și control, acestea se realizează sub forma unei rețele de echipamente (centralizate sau distribuite) de conducere care permit legarea într-un singur sistem a tuturor echipametelor (AP, PLC, calculatoare etc.) care comandă mașinile-unelte, roboți industriali, sistemul de manipulare, transport și depozitare a pieselor, subansamblelor etc. Structura generală a subsistemului de comandă al FML se distribuie pe nivele ierarhice, numărul acestora depinzând de mărimea SFF, de domeniul de aplicare și de numărul funcțiilor de fabricație flexibilă integrate sistem, Fig.2.3, [44], [72], [73]. Corespunzător FML, la care toate funcțiile ilustrate în Fig.2.1 sunt automatizate, structura generală a subsistemului de comandă se prezintă ca o structură distribuită pe patru nivele din Fig.2.3. La partea inferioară a structurii de comandă (Nivelul 1) se află echipamentele industriale de comandă a mașinilor-unelte, roboților industriali, precum și echipamentele de comandă locală a depozitelor și sistemelor de transport. La nivelul ierarhic 2 se află PLC-ul sau calculatorul de conducere locală a fabricației care realizează conducerea echipamentelor din nivelul inferior și transmiterea informațiilor către nivelul

superior. La acest nivel se realizează diagnosticarea instalațiilor și echipamentelor de lucru dar în unele cazuri și planificarea producției la nivel de celulă de fabricație.

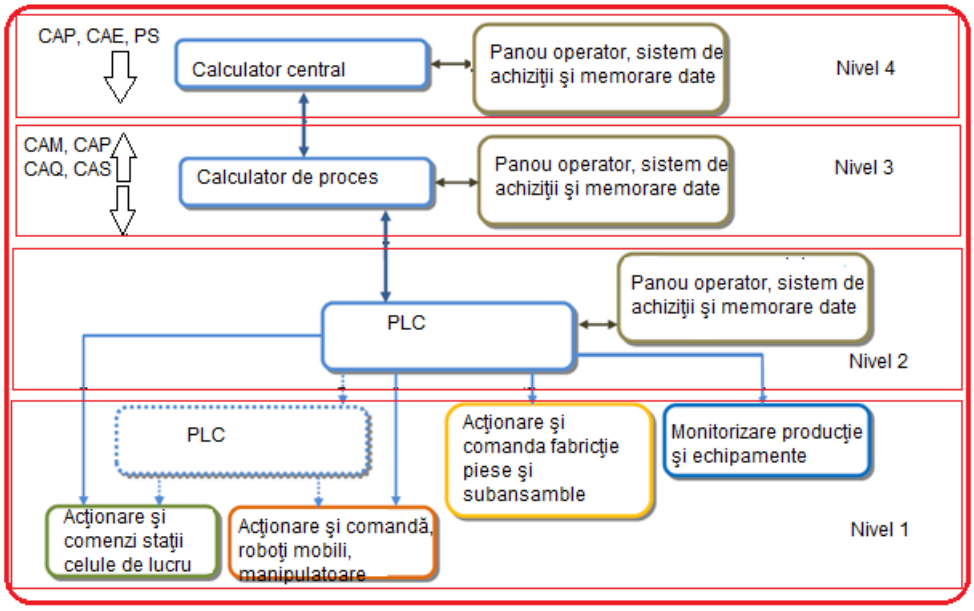


Fig.2.3. Structură de conducere multinivel a unei FML

La nivelele ierarhice 3 și 4 se realizează proiectarea produselor, pregătirea și planificarea fabricației, aceste nivele ierarhice putând funcționa și independent. Pentru realizarea unui concept de produs finit, sistemele de la nivelele inferioare sunt conectate la nivelele superioare, acestea fiind conectate la o structura de procesare de capacitate mare, care realizează automat funcțiile de proiectare a tehnologiei de prelucrare și elaborare a programelor, CAM, de planificare a prelucrărilor și a producției, CAP, de control și supraveghere a proceselor și subsistemelor, CAQ, și de întreținere, CAS. La nivelul 4 se află un calculator care realizează funcțiile de concepție și de proiectare constructivă a produselor, CAD, de analiză a formei și structurii produselor și de rentabilizare, CAE, și de planificare strategică (SP). Structura generală a subsistemelor de comandă dintr-o FML, prezentată în Fig.2.4 este una generală, care să fie aplicabilă pentru orice sistem flexibil de fabricație. În funcție de gradul de complexitate al funcțiilor ce se realizează în cadrul FML și în funcție de modul cum se organizează activitățile pe diferite nivele ierarhice apar modificări, unele dintre nivele putând chiar să nu existe.

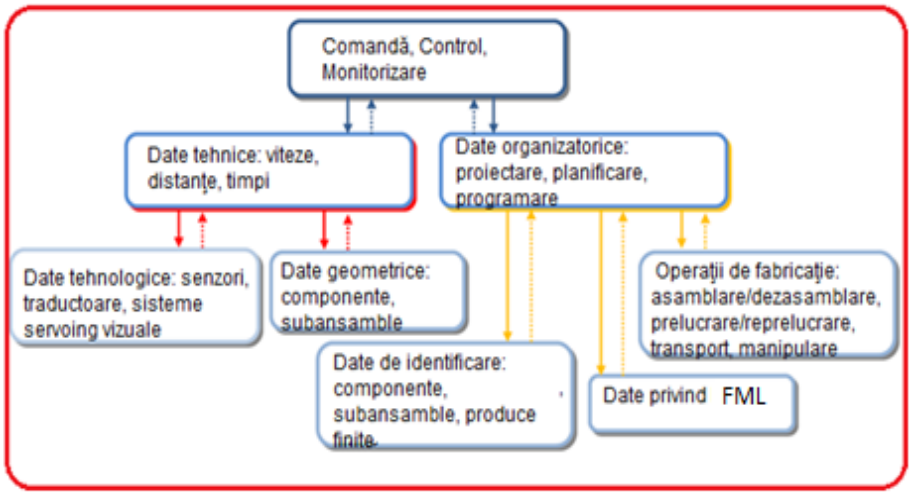


Fig.2 .4. Structura sistemului de monitorizare și conducere a unei FML

2.5. Performanțele și optimizarea FML

În cazul planificării producției, cu deplasări ale componentelor/pieselor sau a echipamentelor de transport, operații de manipulare sau fabricație și durate ale proceselor tehnologice cu timpi de așteptare la stocuri (de componente sau subansamble) intermediare se obțin indicatori de performanță, relativ reduși în cadrul FML, [62], [83]. În acst sens, sunt edificatoare două cifre, reprezentând valori medii, referitore la productivitate:

- din tot timpul consumat pentru a produce o piesă numai 5% este cheltuit de celula de fabricație;
- din tot timpul consumat de celula de fabricație, numai 1,5% este folosit pentru prelucrarea efectivă.

Prima cifră arată că 95% din timpul consumat pentru fabricarea sau prelucrarea unei piese este cheltuit pentru manipulare și transport semifabricate sau piese, pentru așteptări alimentare stocuri (magazii), pentru prelucrare și procesare. În cazul sistemelor flexibile de fabricaţie, timpul efectiv de lucru ajunge la 50-85% din totalul timpului de lucru concomitent cu o creştere a gradului de utilizare a capacităţilor de producţie. Pornind de la acestă constatare, rezultă că indicele de performanță aferent eficienței utilizării celulei de fabricație este foarte scăzut, existând astfel posibilități de îmbunătățire a performanțelor prin planificarea task-urilor procesului de fabricație și mărirea flexibilității SFF prin introducerea de echipamente care îndeplinesc mai multe sarcini. Pentru aprecierea eficienței unui mod de organizare a fabricației pot fi considerate valorile a două tipuri de variabile care caracterizează procesele de producție flexibilă în dinamica lor: variabile referitoare la debite ale produselor care se găsesc în curs de prelucrare (variabile denumite “fluxuri”) și variabile referitoare la acumulări intermediare de produse parțial prelucrate, aflate în stare de stagnare (variabile denumite uneori “nivele”). Cu cât raportul dintre dintre valorile medii ale fluxurilor și nivelurilor este mai mare cu atât este mai ridicată eficiența planificării de fabricație. Datorită acestui motiv, unul dintre obiectivele principale ale perfecționării organizării fabricației este legat de raporul menționat și de metodele abordate de planificare a task-urilor. Dacă fabricația este astfel proiectată și organizată încât prin calcule corespunzătoare să se determine cantitățile maxime de material, semifabricate și piese care se pot găsi un timp cât mai mare în stadiul de prelucrare, deplasare sau transport cu stabilirea succesiunilor și traseelor optime din punct de vedere al micșorării cantităților stocate intermediare și a duratelor de stagnare, atunci rezultă o creștere semnificativă a raportului dintre fluxuri și niveluri și a eficienței planificării fabricației. Introducerea de roboți cu funcții multiple, duce la îmbunătățirea timpului de prelucrare și devine semnificativ în raport cu durata ciclului total de fabricație. Datorită acestor motive, este necesară optimizarea FML, cunoscută și sub denumirea de echilibrare.

O prima etapă în optimizare o constituie conducerea și optimizarea fluxurilor de activităţi (operații) care implică două faze: planificare fluxurilor şi execuţia acestora. Pentru FML trebuie găsite metode adecvate de planificare; acestea trebuie să contribuie la evitarea acţiunilor de blocare şi conflictuale. Planificarea FML-urilor se poate descrie ca o abordare a coordonării, în care din faza de proiectare a secvenţei de acţiuni pe care le execută un echipament care realizează o anumită operație trebuie să se ţină seama de interacţiunile dintre echipamentele din componența FML. Această abordare trebuie să permită sistemului de planificare a operațiilor să construiască un plan care să conţină detalii ale tuturor operațiilor şi interacţiunilor viitoare. În acest fel se realizează propriile scopuri şi se întrepătrunde execuţia operațiilor din FML cu mai multe etape de planificare şi re-planificare.

Varianta cea mai des utilizată de optimizare a FML în cazul sistemelor centralizate, o reprezintă planificarea în spaţiul stărilor, pentru cazul sistemelor descentralizate în care o parte din operații se împart unor utilaje care deservesc FML. Există și planificarea în spaţiul planurilor care poate furniza soluţii adecvate. Dezvoltarea unui algoritm de planificare a operațiilor într-un mediul de fabricaţie flexibil prezintă o mare dificultate cauzată de anumite aspecte de natură practică. De exemplu, un produs are deja un plan de fabricație dezvoltat de proiectant şi deci putem vorbi de o planificare offline a operațiilor de fabricație deja efectuată și cunoscută în prealabil. Planificarea offline a operațiilor va continua cu o planificare online în care planul dezvoltat este completat astfel încât să fie pregătit pentru fabricație.

O altă etapă de optimizare a FML o reprezintă creșterea gradului de flexibilitate a echipamentelor, dezvoltarea și perfecționarea utilajelor pentru a putea executa mai multe operații. Prin mărirea gradului de flexibilitate se reduc timpii în care, asupra produsului, pe linia de fabricație, se execută diverse operații care necesită timp (transport, manipulare, etc.).

2.6. Structura și acționarea FMML, A/DML HERA&HORSTMANN

Structura sistemului flexibil de asamblare/dezasamblare HERA&HORSTMANN, Fig.2.5, este compusă din două mari susbsisteme:

1. Subsistemul de A/D, structura hardware:
   - 5 stații (celule) de lucru (de A/D și depozitare) și o stație depozit produse finale asamblate;
   - sistem de transport dintre stațiile de A/D de tip benzi transportoare;
   - sistem de manipulare și depozitare produse finale asamblate, sistem de tip lift.

FMML, A/DML HERA&HORSTMANN asamblează un produs final din 5 componente, Fig.2.6 și 2.7): 1-palet (pallete); 2-corp (body); 3-capac (cover), 4-cilindru metalic (metal cylinder); 5-cilindru de plastic (plastic cylinder). Stațiile de lucru S1 (Fig.2.8), S2 (Fig.2.9), S3 Fig.2.10) și S4 (Fig.2.11) sunt echipate cu câte o magazie de coomponente, în fiecare magazie găsindu-se un alt tip de componentă care intră în alcătuirea produsului, cu exceptia magaziei care depozitează, în mod aleator, cilindri de metal sau de plastic. În teză se face convenţia că un produs final asamblat este considerat nesatisfăcător calitativ dacă conţine cilindri de materiale diferite. Un astfel de produs final trebuie returnat pe linie pentru dezasamblare, în vederea recuperării componentelor. Fiecare stație eliberează din depozit o componentă și o asamblează. Locaţia magaziei de depozitare a unei componente coincide cu locaţia de asamblare. Fiecare staţie este echipată cu o bandă transportoare. Depunerea unei piese pe banda transportoare se realizează prin intermediul unor elemente de execuție de tip piston pneumatic, acţionat de un sistem pneumatic. Fiecare stație este echipată cu traductori de poziție, pentru o poziționare precisă în dreptul fiecarei magazii şi locaţie de asamblare. Stația S1, Fig.2.8, conține în magazie piesa P1, Fig.2.6-1, denumită și palet, acesta are rolul de a transporta celelalte piese pe benzile transportoare.

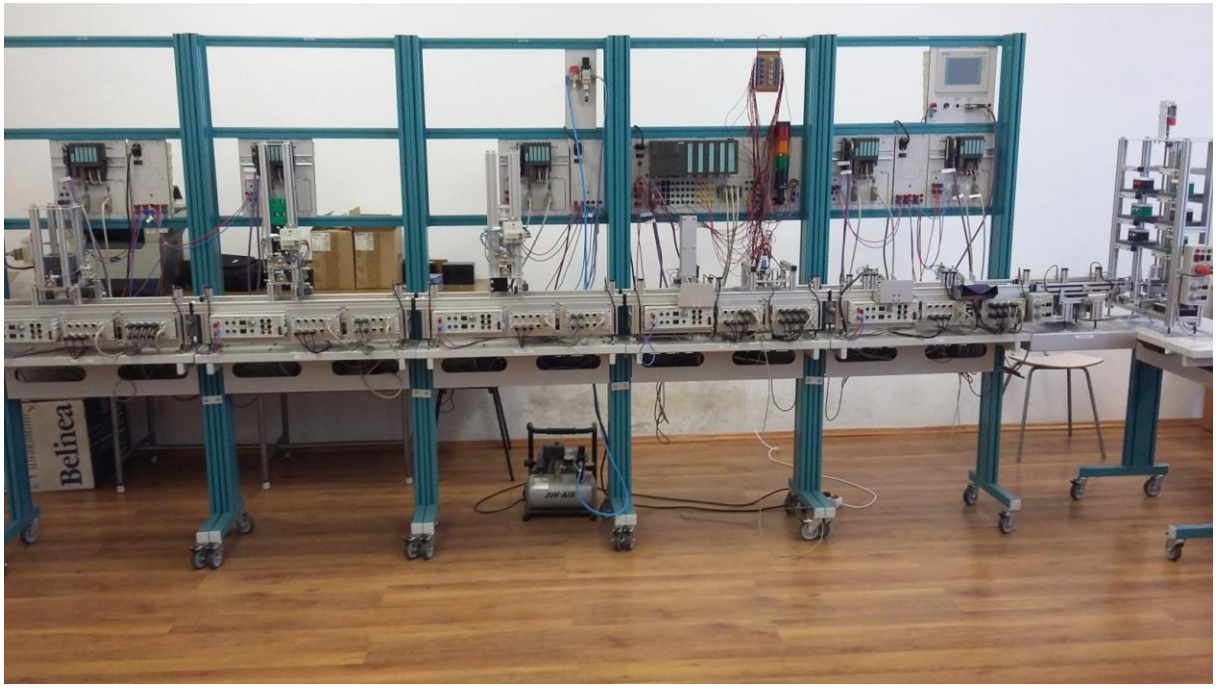


Fig.2.5. FMML cu 6 stații de lucru, A/DML HERA&HORSTMANN

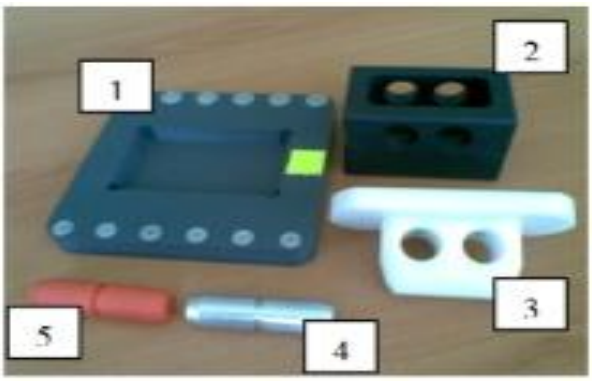


Fig.2.6. Componente

Fig.2.7. Produs final asamblat

Paletul are în componență șase discuri metalice dispuse pe ambele părți laterale, acestea au rolul de a transmite poziția paletului pe banda transportoare prin identificarea numărului de discuri de către traductoarele de poziție. Magazia stației S2, Fig.2.9, conține un corp dreptunghiular, Fig.2.6-2, prevăzut cu o deschidere în partea superioară și două deschideri în părțile laterale. În partea superioară se montează la stația S3, Fig.2.10, piesa P3, denumită și capac, Fig.2.6-3, aceasta se prezintă în două forme constructive, cu margine ascuțită sau cu margine rotundă. La stația S4, Fig.2.11, se montează piesa P4, denumită și cilindru, Fig.2.6-5 şi Fig.2.6-5, în părțile laterale. Această stație mai conține și o altă magazie de depozitare a pieselor P4, în momentul în care, la asamblare s-a produs o eroare sau aceasta nu a fost corect executată, cilindul cade automat în această magazie. Stația S4 mai conține un sistem de testare a produsului final,Fig.2.11, înainte ca acesta să fie transportat și depozitat la stația S6. În funcție de testarea efectuată, stația S4 transmite date despre produs sistemului de conducere, acesta selectează locația unde va fi depozitat noul produs. Testarea este efectuată cu ajutorul a trei traductoare, două dintre ele verifică dacă piesa P4, este din material metalic sau plastic, iar cel de-al treilea traductor verifica piesa P3. Stația S5, Fig.2.12, din cadrul FMML, are rolul de a efectua o dezasamblare parțială a unui produs din depozit, mai exact efectuează dezasamblarea piesei finale asamblate, Fig.2.7. Dezasamblarea este realizată cu ajutorul a două pistoane pneumatice, piesa dezasamblată

cade automat în magaziile aferente fiecărui piston. Staţia de depozitare, S6, Fig.2.13, are rolul de a stoca produsele finite, în 8 locații. Depozitul este compus dintr-un sistem de manipulare de tip lift, care ridică produsul de pe banda transportoare și îl poziționează în locația corespunzătoare. Liftul efectuează atât operații de stocare în depozit cât și operații de scoatere din depozit.

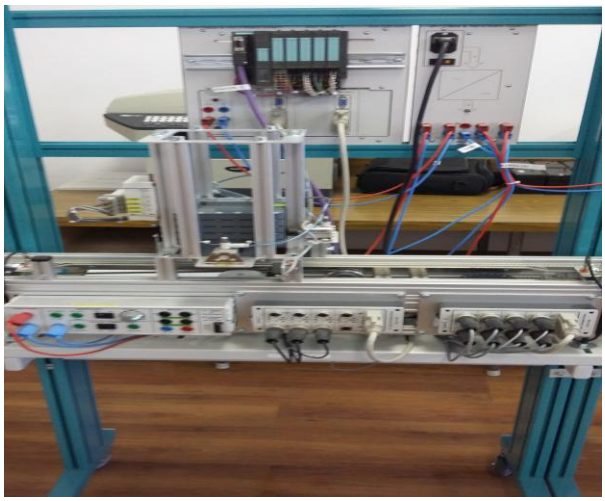


Fig.2.8. S1, depozit/asamblare palet (pallet)

Fig.2.9. S2, depozit/asamblare corp (body)

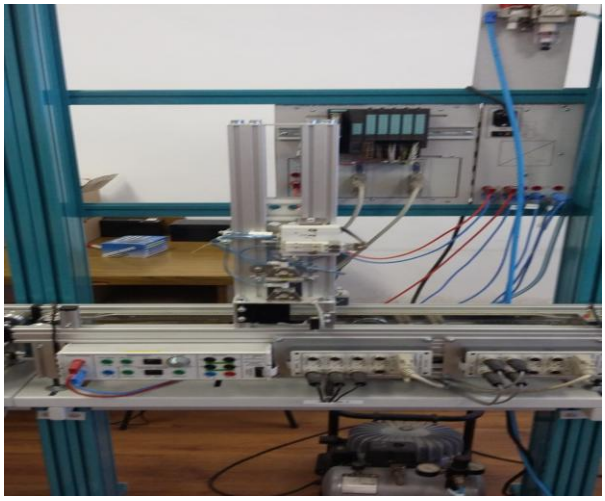


Fig.2.10. S3, depozit/asamblare capac (cover)

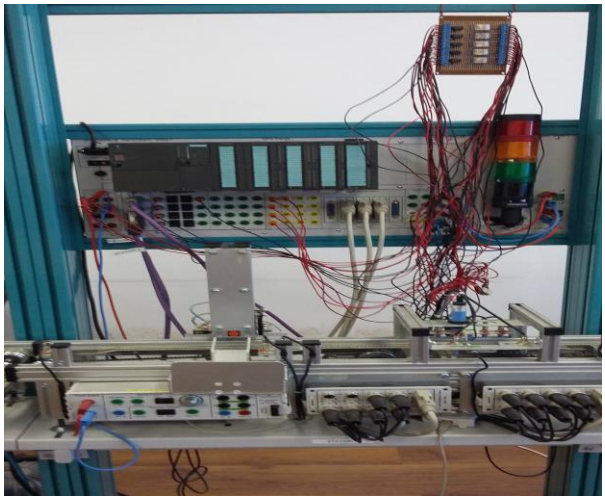


Fig.2.11.S4, depozit/asamblare/test cilindri

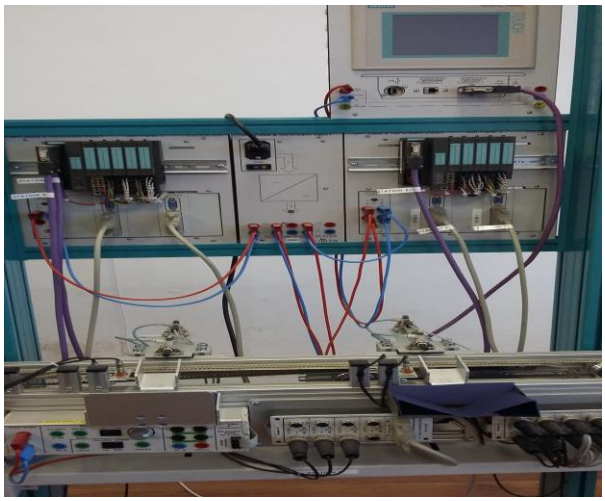


Fig.2.12. S5, dezasamblare cilindri

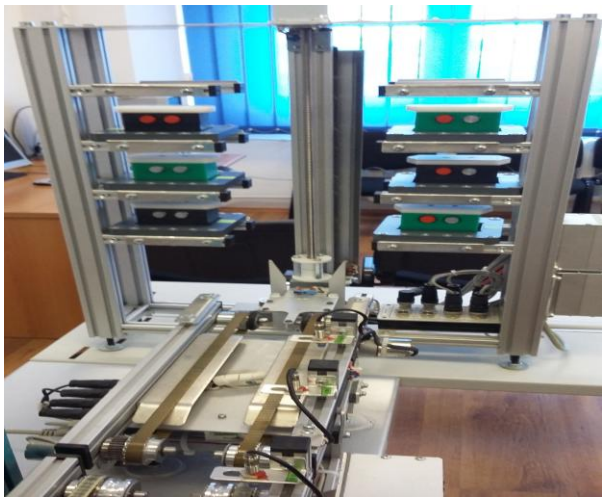


Fig.2.13. S6, depozitare produs final asamblat

2. Acţionarea, achiziţia de date şi conducerea FMML, A/DML HERA&HORSTMANN.

Structura de acţionare şi conducere A/DML HERA&HORSTMANN, Fig.2.14, este de tip distribuită și este formată dintr-un PLC, SIEMENS Simatic S7-300 cu procesor din seria CP 314C-2 DP și modul de comunicație SIEMENS CP 343-2. Acestea se conectează pe magistrala PROFIBUS DP care conectează *modulele auxiliare*, MA, de interfațare I/O de tip SIEMENS ET200S-IM 151-1 distribuite pe fiecare dintre stațiile FMML, A/DML. Fiecare din cele 6 module SIEMENS ET200S-IM 151-1 prezintă module de I/O digitale și analogice, acestea preluând semnale provenite de la traductoare și transmițând comenzi elementelor de execuție. Pe magistrala PROFIBUS DP este conectat un terminal de tip panou operator SIEMENS Simatic HMI TP 177, prin intermediul căruia se poate vizualiza starea sistemului flexibil și se poate pune în execuție un proces de asamblare sau dezasamblare.

Compatibilizarea dintre A/DML, care este acţionată la 24V, şi placa de achizitii date împreună cu calculatorul de proces care operează la 5V, este realizată prin intermediul unei plăci cu relee, plasată pe cadrul staţiei 4, Fig.2.11, iar conexiunile corespunzătoare sunt prezentate în Fig.2.15. Comunicarea dintre automatul programabil al liniei flexibile de mecatronică şi staţia de lucru care asigură sincronizarea cu platforma robotică se face prin intermediul unei plăci de achiziţie DAQ NI USB6008. Deoarece ieşirile şi intrările digitale ale automatului programabil Siemens CPU 314C-2 DP funcţionează cu tensiune de 0-24 V iar placa de achiziţie lucrează cu tensiuni cuprinse între intervalul de 0-5 V s-a folosit o placă cu relee pentru a evita avarierea plăcii de achiziţie. În figura următoare este exemplificată schematic placa cu relee şi legăturile aferente cu intrările/ieşirile digitale ale automatului programabil şi placa de achiziţie.

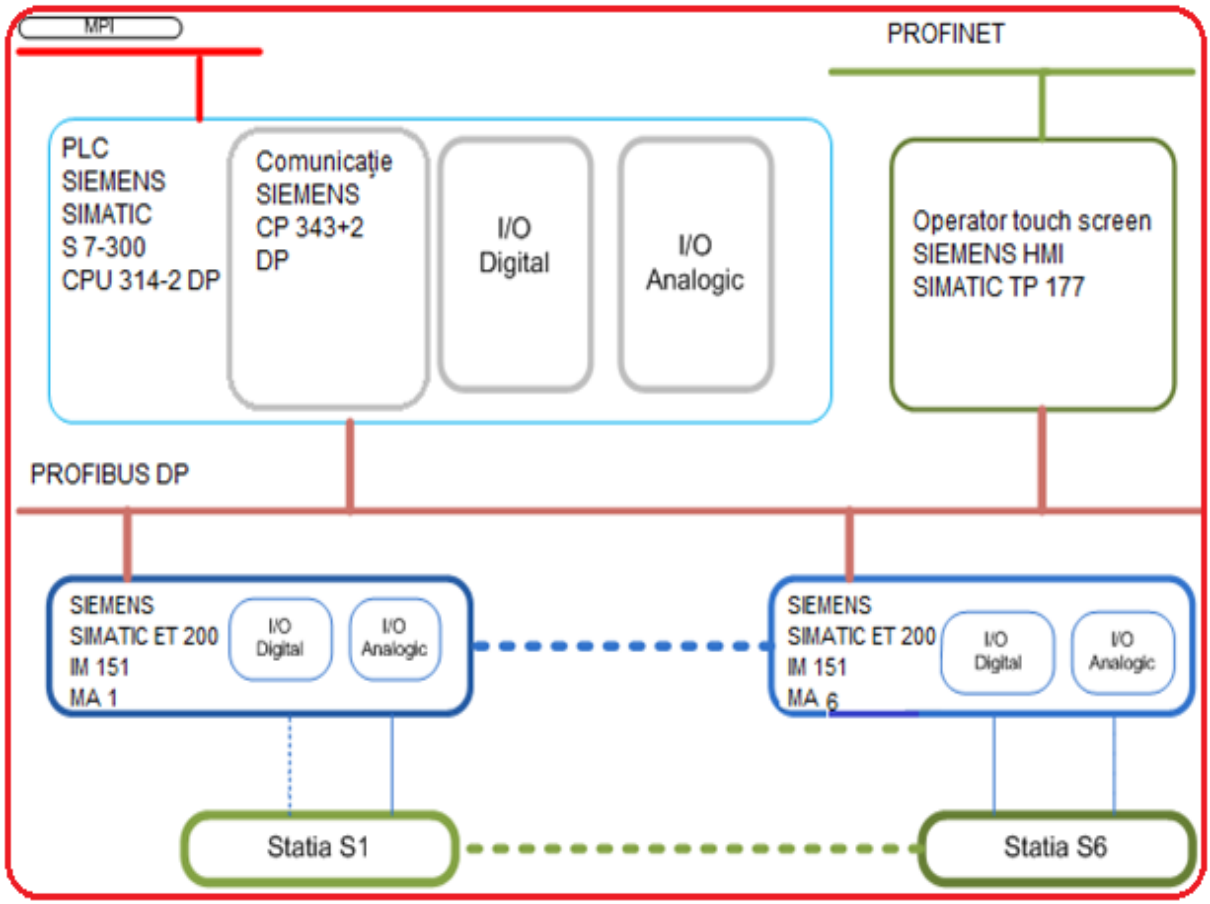


Fig.2.14. Acţionarea, achiziţia de date şi conducerea FMML, A/DML HERA&HORSTMANN

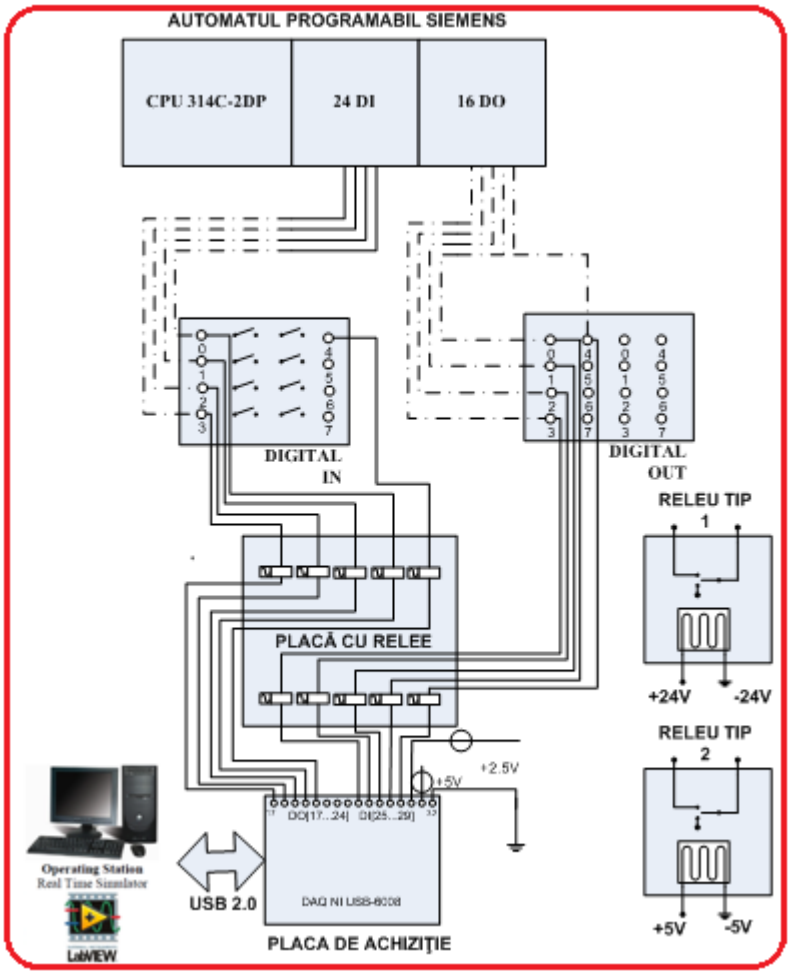


Fig.2.15. Compatibilizare A/DML-placă achiziţii date-calculator proces

2.7. Structura şi acţionarea FMML, P/RML FESTO MPS-200

FMML, P/RML FESTO MPS-200, Fig.2.16, este un sistem mecatronic de laborator configurbil, ce poate fi asimilat unei linii de fabricaţie flexibilă care realizează operaţii de prelucrare, sortare și depozitare. Aceasta este compusă din 4 stații (celule), fiecare realizând operații diferite. Structura FMML, P/RML FESTO MPS-200 este alcătuită din următoarele stații de lucru:

- Staţia de manipulare/sortare, Fig.2.17, preia componenta prelucrată de la stația anterioară cu ajutorul unui manipulator axial echipat cu gripper pneumatic, după ce a fost supusă unui test de culoare. Dacă culoarea diferă față de cea corectă, sistemul pneumatic preia piesa și o depune într-o magazie alăturată. Magazia de componente este compusă din două părți. Fiecarei părți i se atribuie o piesă de o anumită culoare. Dacă piesa a trecut testul de culoare, aceasta nu va mai fi stocată în magazie, sistemul de manipulare axial o depune într-o locație a stației următoare;
- Staţia de prelucrare, Fig.2.18, execută două operații distincte, una de găurire și cealaltă de alezare a componentei provenite de la stația de testare și sortare. Celula de prelucrare prezintă un sistem de acumulare de tip masă rotativă a șase piese. Prin fiecare rotație a sistemului rotativ se poziționează câte două piese simultan pentru a fi prelucrate. Operațiile de găurire și alezare se execută simultan de către două dispozitive mecatronice. După terminarea ambelor prelucrări, sistemul de acumulare rotativ se poziționează în dreptul următoarei stații. Celula de prelucrare este echipată

cu un sistem de senzori de proximitate pentru o poziționare precisă în dreptul celor două puncte de prelucrare;

- Staţia de acumulare (buffer), Fig.2.19, are rolul de a prelua și stoca piesele provenite de la stația de asamblare. Poate stoca un număr de 5 piese, care vor fi trimise individual către stația următoare la anumite intervale de timp. Această celulă este echipată cu un sistem de senzori pentru monitorizarea nivelului de încărcare, un sistem pneumatic de oprire/eliberare piese;
- Staţia de sortare/depozitare, Fig.2.20, în urma procesului de sortare sunt selectate ordinea și locația unde vor fi depozitate produsele finale. Sortarea produselor este realizată cu ajutorul unui senzor de culoare. Sistemul de depozitare este compus din 3 magazii în care fiecărei magazii îi sunt atribuite produse de o anumită culoare. Depozitarea produselor se face cu ajutorul unui manipulator axial echipat cu un gripper pnematic, care preia produsul de pe banda transportoare a stației anterioare și o depune în depozit. Manipulatorul axial este controlat cu ajutorul unui controller MTR-DCI-42S. Celula de depozitare reprezintă ultima stație din cadrul sistemului flexibil de fabricație FESTO MPS-200 având o capacitate de stocare de 18 produse prelucrate.

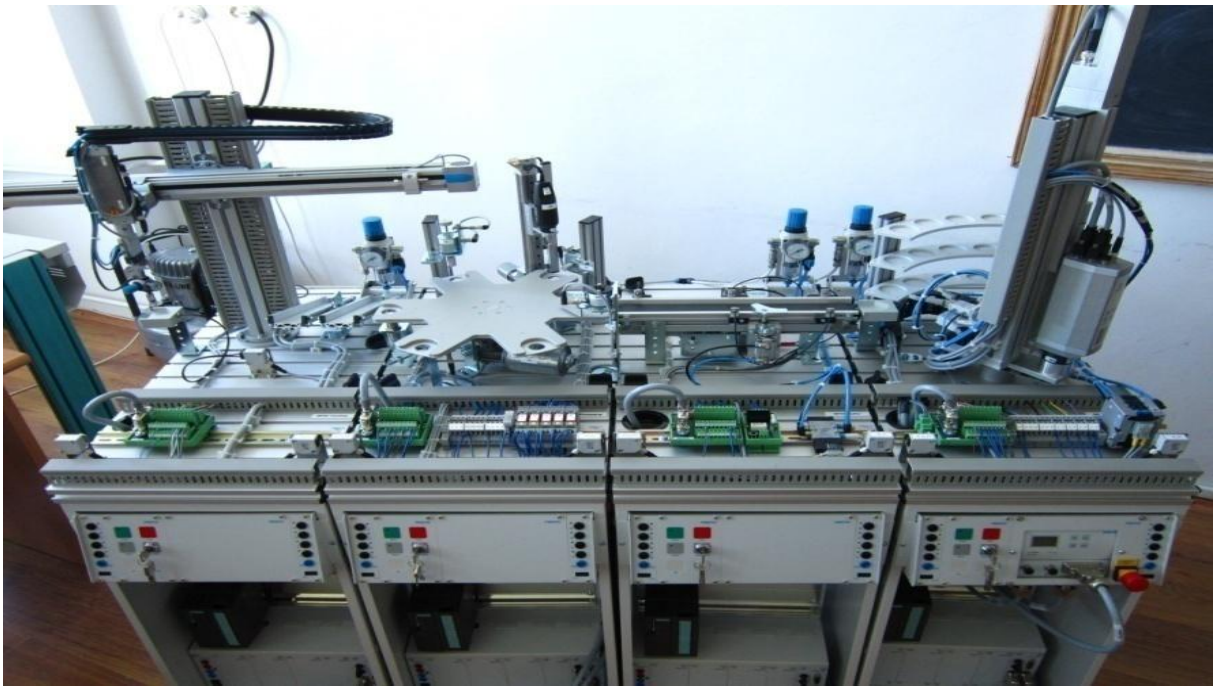


Fig.2.16. FMML cu 4 staţii de lucru, P/RML FESTO MPS-200

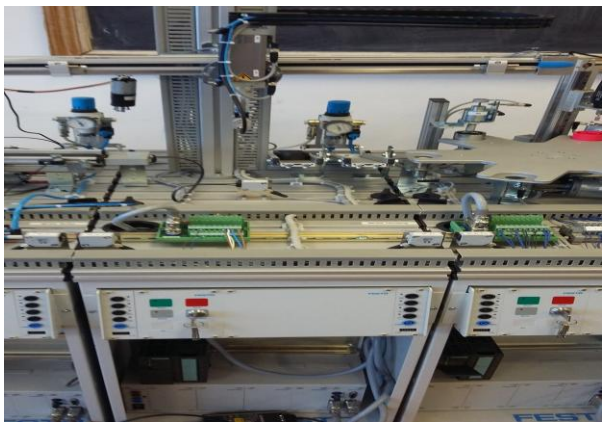


Fig.2.17. Staţie de manipulare

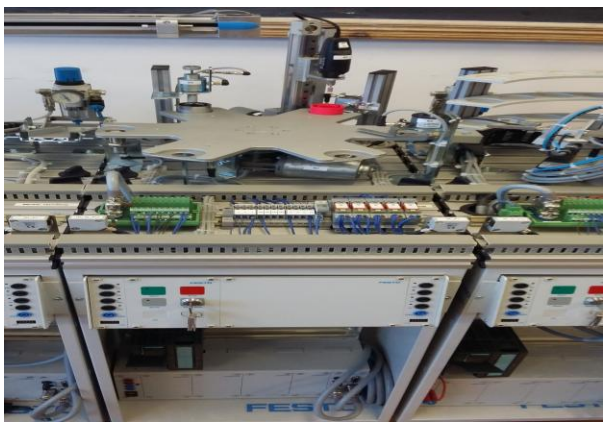


Fig.2.18. Staţie de prelucrare

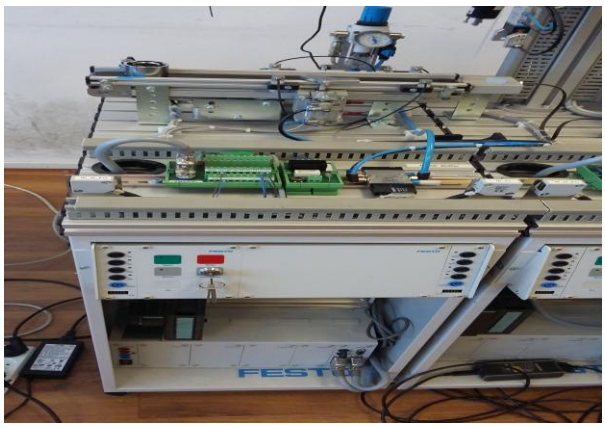


Fig.2.19. Stație acumulare (buffer)

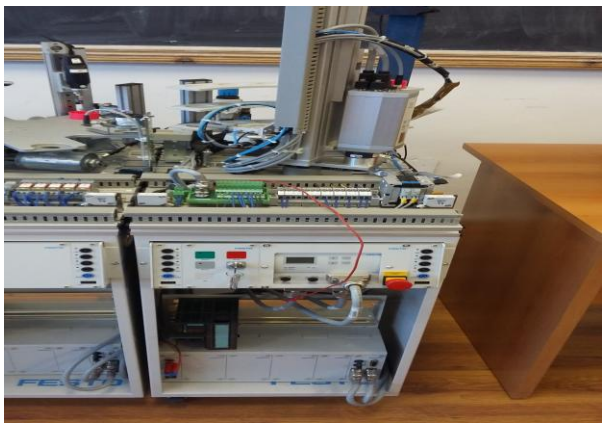


Fig.2. 20. Staţie sortare/depozitare

Structura sistemului de automatizare, Fig.2.21, este locală pe fiecare stație și este compusă dintr-un PLC SIEMENS Simatic S7-300 cu procesor din seria CP 312C-2 DP și module auxiliare de interfațare I/O distribuite pe fiecare dintre stațiile sistemului flexibil. Fiecare din cele 4 PLC-uri prezintă module de I/O digitale și analogice, acestea preluând semnale provenite de la traductoare și transmițând comenzi elementelor de execuție. Stația de depozitare/sortare este echipată cu un controler necesar manipulatorului axial și un modul de comunicație care realizează interfața dintre controler și PLC. Sistemul de automatizare comandă atât sistemul de senzori și traductoare cât și elementele de execuție de tip electric sau pneumatic. Se face și la FMML, FESTO MPS-200, convenţia că piesele de o anumită culoare (roşie) să fie depozitate pe magazia superioară a stației S4. Sunt considerate ca fiind piese care nu satisfac criteriile de calitate şi, prin urmare, vor fi preluate de un WMR echipat cu RM pentru a fi transportate la începutul liniei, la staţia de manipulare/sortare, S1. Aici vor fi supuse unui alt test care va decela între un produs complet compromis (rebut) şi un altul care în urma repetării opreţiilor de prelucrare (reprelucrare) poate fi adus la parametri doriţi.

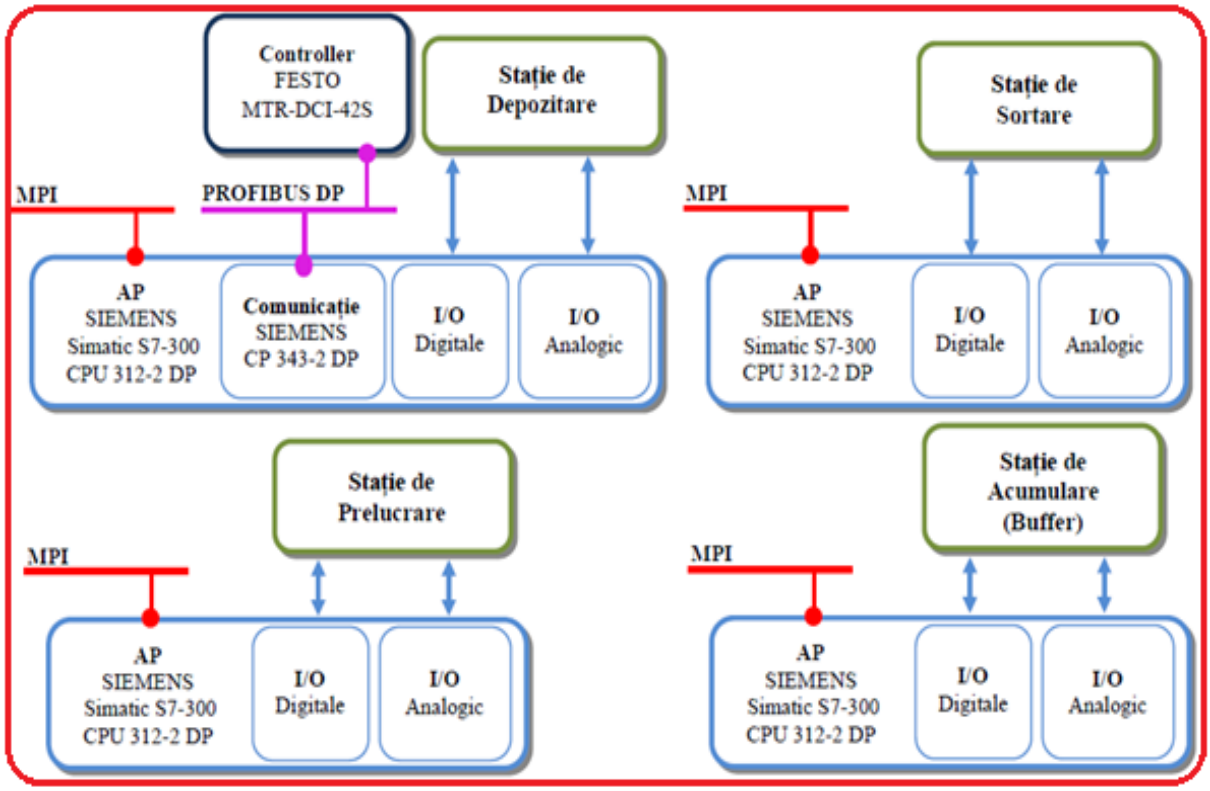


Fig.2.21. Acţionarea şi conducerea FMML, P/RML FESTO MPS-200

2.8. Concluzii

Principalele contribuţii rezultate în urma cercetărilor efectuate în cadrul prezentului capitol au vizat, în mod deosebit, analiza FMMLs cu referire la două tipuri de procese industriale de fabricaţie flexibilă, asamblare şi prelucrare. Scopul este ca pe langă flexibilizarea fabricaţiei să se aduge sarcini noi în sensul că sistemele de fabricaţie capătă valenţe de reversibiltate şi reutilizabilitate de produs finit. În acest sens, sistemele de fabricaţie felexibilă devin capabile să execute şi dezasamblare, respectiv şi reprelucrare. Corespondențele acestor două sisteme industriale, de fabricaţie flexibilă, sunt facute, în teză, cu două FMMLs, A/DML şi P/RML. De asemenea, s-a urmărit elaborarea unor noi structuri flexibile de fabricație capabile să îndeplinească sarcini multiple cu aceleași echipamente în cadrul diferitelor procese de fabricație. Rezultatele analizei obținute, au urmărit desevirea proceselor de fabricație flexibilă cu WMRs echipați cu RMs capabile să îndeplinească două sarcini diferite, transport și manipulare. Ca rezultat al optimizării gradului de flexibilitate al echipamentelor și introducerea structurilor robotice care deservesc A/DML şi P/RML, se pot evidenţia, în Fig.2.22, noi structuri care permit posibilitatea dezasamblării complete a unui produs care s-a asamblat, pe aceleași echipamente, dezasamblare deservită de WMRs echipați cu RMs sau reprelucrarea prin aducerea de către sistemul robotic mobil a produsului pentru a se aplica o a doua prelucrare. Se evidenţiază subsistemele de fabricație flexibilă(prelucrare, transport, manipulare) care pot executa operații A/D și P/R distincte cu aceleași staţii.

Cele mai importante contribuţii din cadrul acestui capitol sunt:

- obținerea unor structuri optimizate de fabricație flexibilă care permit cu aceleași staţii efectuarea de operaţii de A/D şi P/R;
- realizarea de corespondenţe între procesele industriale de fabricaţie flexibilă şi FMMLs;
- introducerea WMRs echipaţi cu RMs pentru reversibilizare şi mărirea gradului de flexibilitate.

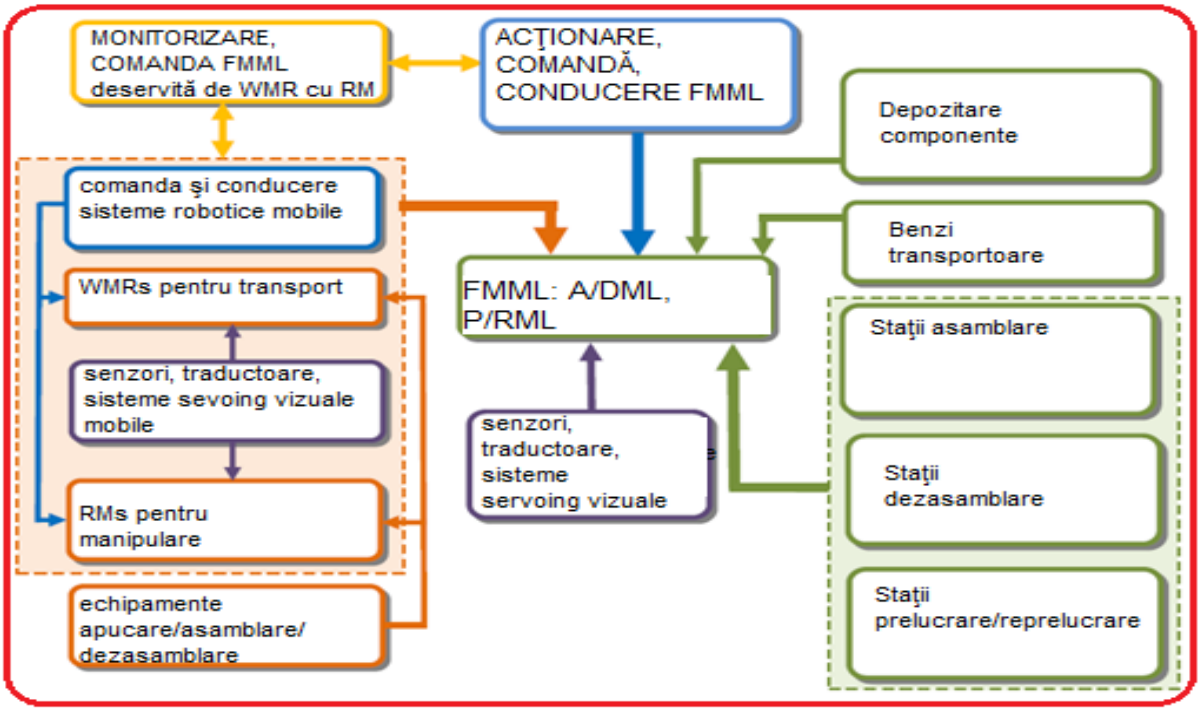


Fig.2.22. FMMLs, de A/D şi P/R, deservite de WMRs echipați cu RMs

Capitolul 3

FMMLs deservite de WMRs echipați cu RMs, ipoteze de lucru, atribuirea și planificarea taskurilor, echilibrarea

În acest capitol se fac ipoteze preliminare, utile la definirea, planificarea taskurilor şi echilibrarea FMMLs, A/DML și P/RML, cu roboți integrați. Mai întâi, pentru A/DML, deservită de una sau două platforme mobile, se tratează generalizarea la N stații de A/D. Urmează particularizarea la A/DML HERA&HORSTMANN cu 5 stații de A/D. Tot la A/DML este propus un criteriu de optimizare care poate fi asimilat și ca un criteriu de echilibrare a liniei pentru secvenţa de operații de dezasamblare. Maximizarea criteriului este echivalentă cu valorificarea maximală a componentelor, în secvență de dezasamblare, datorită reutilizării componentelor în procesul de fabricație, [40], [41], [47], [48].

3.1. Ipoteze preliminare privind A/DML deservită de un WMR echipat cu RM

A/DML deservită de WMR echipat cu RM în decursul fazelor de dezasamblare, Fig.3.1, face ca FMML să devină reversibilă, fiind capabilă să reintroducă în ciclul de fabricaţie componente reutilizabile. Mai mult, WMR este utilizat în procesul de dezasamblare pentru a prelua componentele de la locațiile de dezasamblare şi a le transporta la magaziile de depozitare.

3.1.1. Ipoteze privind asamblarea

Liniile de asamblare sunt sisteme de fluxuri de producţie de cantităţi mari unde, subansamblele şi produsul final satisfac anumite standarde de calitate. În acord cu cele prezentate în [3], [5] şi [6], există câteva scheme de clasificare a liniilor de asamblare care iau în considerație natura produselor, modurile și timpii de operare. Corespunzător acestor clasificări se vor face următoarele ipoteze cu privire la secvența de asamblare dintr-o A/DML.

I.1. A/DML este o linie de fabricație unimodel, după natura produsului asamblat, este “paced-line”, dupa modul de operare deoarece transferul între stațiile de lucru se face sincronizat şi este o linie deterministă, după natura timpilor de operare (timpii de operare sunt cunoscuţi).

I.2. A/DML are un număr fix de staţii de lucru care asigură minimizarea timpului de ciclu de asamblare și echilibrarea A/DML.

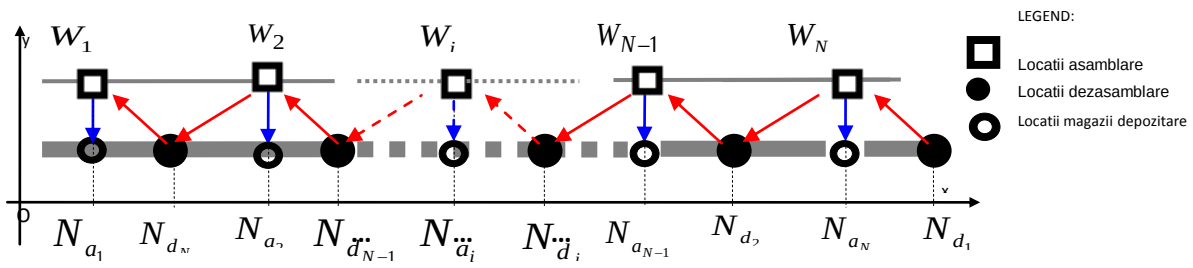


Fig.3.1. Locaţiile de A/D și ale magaziilor de depozitare

3.1.2. Ipoteze privind dezasamblarea

În [43], [64] şi [65], echilibrarea secvenței de dezasamblare a unei A/DML este descrisă pentru dezasamblarea completă a unui produs. Pentru a îmbunatăţi eficienţa unei A/DML în ceea ce priveşte recuperarea componentelor rezultate din dezasamblare, se impune elaborarea unui criteriu a cărui optimizare este echivalentă cu DBL. Pentru a elabora un model şi criteriu pentru DLB se fac suplimentar următoarele ipoteze:

I.3. A/DML în procesul de dezasamblare este “paced-line”;

I.4. Un singur tip de produs se dezasamblează iar fiecare produs din gamă are o configuraţie identică;

I. 5. Se consideră că procesul de dezasamblare este complet sub toate aspectele adică, timpii de executie ale taskurilor, timpii de ciclu, cererea de componente şi costuri sunt cunoscute cu certitudine, deci procesul de dezasamblare este determinist;

I.6. Există N staţii de lucru, asezate în linie, prima staţie, S5, preia produsul care trebuie dezasamblat şi numărul de staţii este același cu numărul de componente care se dezasamblează;

I.7. Fiecare perioadă este asociată unei singure componente care se dezasamblează, deci există N perioade iar fiecare perioadă este referită ca un ciclu elementar. Aceleaşi taskuri se execută în fiecare ciclu;

I.8. Fiecărui task i se asociază un cost, un timp de execuție şi un beneficiu datorat, fie reutilizării componentei care se dezasamblează, fie vânzării acesteia;

I.9. Procesul de dezasamblare porneşte imediat după procesul de asamblare al produsului final care nu trece testul de calitate;

I.10. Locaţiile magaziilor de depozitare coincid cu locaţiile punctelor de asamblare;

I.11. Într-o operaţie de A/D numai o singură componentă se asamblează/dezasamblează;

I.12. Se face convenţia că un produs final asamblat nu trece testul de calitate dacă cilindrii sunt de compoziţie diferită;

I.13. Din momentul când ultima componentă a produsului supus dezasamblării, a fost transportat la magazia de depozitare, porneşte o nouă asamblare;

I.14. Deplasarea WMR, în fiecare ciclu elementar este liniară, cu viteză constantă şi fără obstacole.

Fie N numărul de componente care se asamblează/dezasamblează.
Fie $N_{a_i}, i=\overline{1,N}$ numărul de locaţii de asamblare situate pe sensul pozitiv al axei Ox.
Fie $N_{d_j}, j=\overline{1,N}$ numărul de locaţii de dezasamblare situate pe sensul negativ al axei Ox.
Fie $W_i; i=\overline{1,N}$ locaţiile magaziilor de depozitare, locaţii care sunt identice cu locaţiile punctelor de asamblare.
Fie $D\left(N_{d_j}, W_{N+1-j}\right)$ distanţa între locaţia de dezasamblare, N_{d_j}, şi magazia de depozitare corespunzătoare, W_{N+1-j}.
Fie $D\left(W_{N+1-j}, N_{d_{j+1}}\right)$ distanţa între ultima magazine de depozitare, W_{N+1-j}, şi următoarea locaţie de dezasamblare, $N_{d_{j+1}}$.
Fie $D_{r_j} = D\left(N_{d_j}, W_{N+1-j}\right) + D\left(W_{N+1-j}, N_{d_{j+1}}\right)$ distanţa parcursă de WMR în cel de al $j-lea$ stagiu de dezasamblare.
Variabla $r = 1+3(j-1)$ indexează o stare (‘place’) continuă a WMR, *Pcr*, o tranziție continuă a WMR, *Tcr*, şi o tranziţie discretă a procesului de dezasamblare, *Tdd*. Variabla $k = 1+5(j-1)$ indexează o stare discretă a procesului de dezasamblare, *Pdd*. Variabla $l = 1+4(j-1)$ indexează o tranziţie discretă a WMR, *Tdr*.

3.2. Model și criteriu de optimizare pentru A/DLB

În acest subcapitol, bazat pe ipotezele de mai sus, se prezintă un model al procesului de dezasamblare care se constituie într-un criteriu de optimizare (echilibrare), a cărui extremizare (maximizare) reprezintă o soluţie a problemei A/DBL. Datorită ipotezelor I.1, I.2 și I.10, procesul de asamblare poate fi considerat implicit optimizat (echilibrat), deci problema ALB este implicit rezolvată.

3.2.1. Taskurile aferente procesului de dezasamblare

Fie M numărul total de taskuri necesare dezasamblării unui produs iar M_c numărul de taskuri pe ciclu elementar (perioadă). Taskurile asociate unui ciclu elementar, $TC_i, i=\overline{1,M_c}$, sunt definite mai jos:

TC_1 – Stop linie;

TC_2 – Dezasamblare componentă;.

TC_3 – Poziţionare RM în dreptul locaţiei de dezasamblare;

TC_4 – Prindere componentă care a fost dezasamblată;

TC_5 – Start linie;

TC_6 – Poziţionare RM pentru deplasare WMR;

TC_7 – Deplasare WMR de la locaţia de dezasamblare la magazia de depozitare corespunzătoare;

TC_8 – Poziţionare RM în dreptul magaziei de depozitare;

TC_9 – Depunere componentă în magazia de depozitare corespunzătoare;

TC_{10} – Poziţionare RM pentru deplasare WMR la urmatoarea locaţie de dezasamblare;

TC_{11} – Deplasare WMR de la magazia de depozitare la următoarea locaţie de dezasamblare.

Observația 3.1. Datorită I.11, ultima operație de dezasamblare nu va mai avea loc şi, prin urmare, taskul TC_5 va dispărea din ultimul ciclu elementar.

3.2.2. Criteriu de maximizat pentru DLB

Fie CT timpul de ciclu maxim permis pentru toate ciclurile elementare.

Fie $t_{ij}, i=\overline{1,M},\ j=\overline{1,M_c}$ timpul de execuţie a taskului i aparţinând ciclului j, astfel, t_{ij} este timpul de execuţie al taskului TC_{ij}.

Fie $d_{ij}, i=\overline{1,M},\ j=\overline{1,M_c}$ cererea pentru componenta rezultată din dezasamblare prin execuţia taskului T_{ij}.

Fie NR_{ij} valoarea netă obţinută ca urmare a execuţiei taskului TC_{ij}, i.e. diferenţa dintre valoarea prin reutilizarea sau vinderea componentei rezultată din dezasamblare şi costul de excuţie al taskului:

$$NR_{ij} = R_{ij} - C_{ij} \tag{3.1}$$

unde R_{ij} este valoarea obţinută prin execuţia taskului, TC_{ij}, iar C_{ij} este costul execuţiei taskului, TC_{ij}. Dacă în urma execuţiei taskului, TC_{ij}, nu se dezasamblează o componentă, atunci $R_{ij}=0$.

Fie $DV_{ijk},\ i=\overline{1,M},\ j=\overline{1,N},\ k=\overline{1,N}$ variabilele de decizie prin care se asignează taskuri la staţiile de lucru în fiecare ciclu elementar. Aceste atribuiri se explică astfel: daca taskul i este asignat staţiei j în ciclul k, atunci $DV_{ijk}=1$, altminteri, $DV_{ijk}=0$.

Se consideră următoarele restricţii:

R.1. Cererea pentru o componentă trebuie să satisfacă inegalitatea

$$\sum_{j=1}^{N}\sum_{k=1}^{N} DV_{ijk} \geq d,\ i=\overline{1,M},\ d=\sum_{i=1}^{M}\sum_{j=1}^{M_c} d_{ij} \tag{3.2}$$

R. 2. Timpul de ciclu trebuie să satisfacă inegalitatea

$$\sum_{i=1}^{M} t_{ij} DV_{ijk} \leq CT, \forall j, \forall k \tag{3.3}$$

R.3. Un task nu poate fi asignat decât la o singură staţie în fiecare perioadă

$$\sum_{j=1}^{N} DV_{ijk} \leq 1,\ i=\overline{1,M},\ k=\overline{1,N} \tag{3.4}$$

R. 4. Varibilele de decizie trebuie să fie pozitive deoarece au valori binare, semnificând asignarea fiecarui task la o staţie de lucru în fiecare perioadă

$$DV_{ijk} \geq 0,\ i=\overline{1,M},\ j=\overline{1,N},\ k=\overline{1,N} \tag{3.5}$$

Funcţia obiectiv a cărei maximizare asigură o soluţie pentru DLB, respectând restricțiile R.1, R.2, R. 3 şi R.4, este o sumă a valorilor nete obţinute în urma execuţiei fiecărui task.

$$J_{DLB} = Max \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{N} NR_{ij} DV_{ijk} \tag{3.6}$$

3.3. Particularizare la FMML, A/DML HERA&HORSTMANN, deservită de un WMR echipat cu RM

3.3.1. Descriere procese de A/D

În cadrul descrierii, definirii şi planificării taskurilor liniei flexibile de A/D, HERA&HORSTMANN, se introduc ipoteze de funcționare a proceselor de A/D. Procesul de asamblare, Fig.3.2, se declanșează la pornirea liniei flexibile HERA&HORSTMANN, şi constă în următoarele acţiuni:

- 4 magazii aferente celor patru stații de asamblare sunt încărcate cu componente;
- START proces asamblare la stația S1;
- eliberare palet (componenta C1) pe banda transportoare, în dreptul primei magazii.
- transport C1 la stația S2, pe banda transportoare;
- revenire la starea inițială stație S1 şi START bandă transportoare staţie S2;
- transport C1 la magazia staţiei S2;
- START asamblare corp, C2 pe C1;
- transport C1+C2 la stația S3;
- revenire la starea inițială stație S2 şi START bandă transportoare staţie S3;
- transport C1+C2 la magazia staţiei S3;
- START asamblare capac, C3 pe C2;
- transport C1+C2+C3 la stația S4;
- revenire la starea inițială stație S3 şi START bandă transportoare staţie S4;
- transport C1+C2+C3 la magazia stației S4;
- START asamblare cilindru 1, C4 în C2, primul orificiu;
- transport C1+C2+C3+C4 la al doilea orificiu al C2;
- START asamblare cilindru 2, C5 în C2, al doilea orificiu;
- transport C1+C2+C3+C4+C5 la testul de calitate (test compoziţia cilindrilor), pe banda transportoare staţie S4;
- transport C1+C2+C3+C4+C5 la staţia S5;
- revenire la starea inițială stație S4 şi START bandă transportoare staţie S5;
- transport la staţia de depozitare, S6;
- transport (urcare) cu liftul C1+C2+C3+C4+C5;
- depozitare C1+C2+C3+C4+C5, stânga sau dreapta, după cum piesa asamblată conţine cilidri din acelaşi material sau din materiale diferite;
- revenire (coborâre) lift şi revenire la starea iniţială staţie S6.

Descrierea procesului de dezasamblare deservit de un WMR echipat cu RM, Fig.3.3, porneşte de la constatarea că produsul final este invalidat calitativ. Astfel, se declanșează procesul de dezasamblare. Dezasamblarea este un proces care se declanșează aleator, atunci când produsul asamblat nu trece testul de calitate. Un produs asamblat se consideră invalidat calitativ dacă în componența sa are cilidrii de materiale diferite. Dezasamblarea asistată de un WMR echipat cu RM constă din următoarele acţiuni:

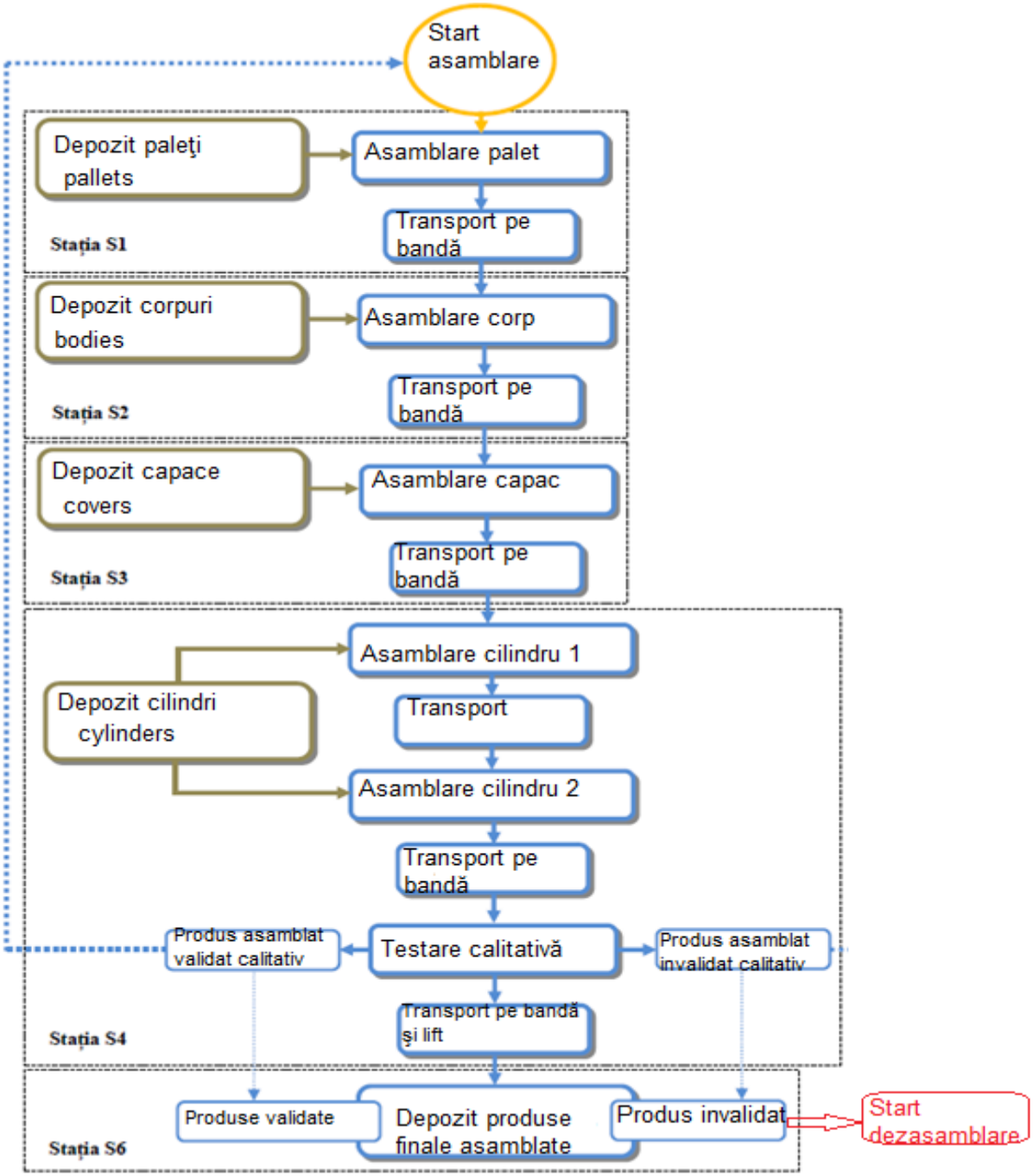


Fig.3.2. Procesul de asamblare pe FMML, A/DML HERA&HORSTMANN

- START dezasamblare staţia S6, transport (coborâre) cu liftul şi cu banda transportoare C1+C2+C3+C4+C5 (produs asamblat rebut) la staţia S5;
- transport C1+C2+C3+C4+C5 pe bandă transportoare staţie S5 la locaţia de de dezasamblare cilindru 1, C4;
- STOP bandă transportoare staţie S5 şi START dezasamblare;
- START ciclu elementar WMR echipat cu RM (poziţionare, apucare componentă, transport la locaţie depozitare component, revenire la urmatoarea locaţie de dezasamblare) şi START bandă transportoare staţie S5;
- transport C1+C2+C3+C5 pe banda transportoare a staţiei S5 la urmatoarea locţie de dezasamblare cilindru 2, C5;
- STOP bandă transportoare staţie S5 şi START dezasamblare;

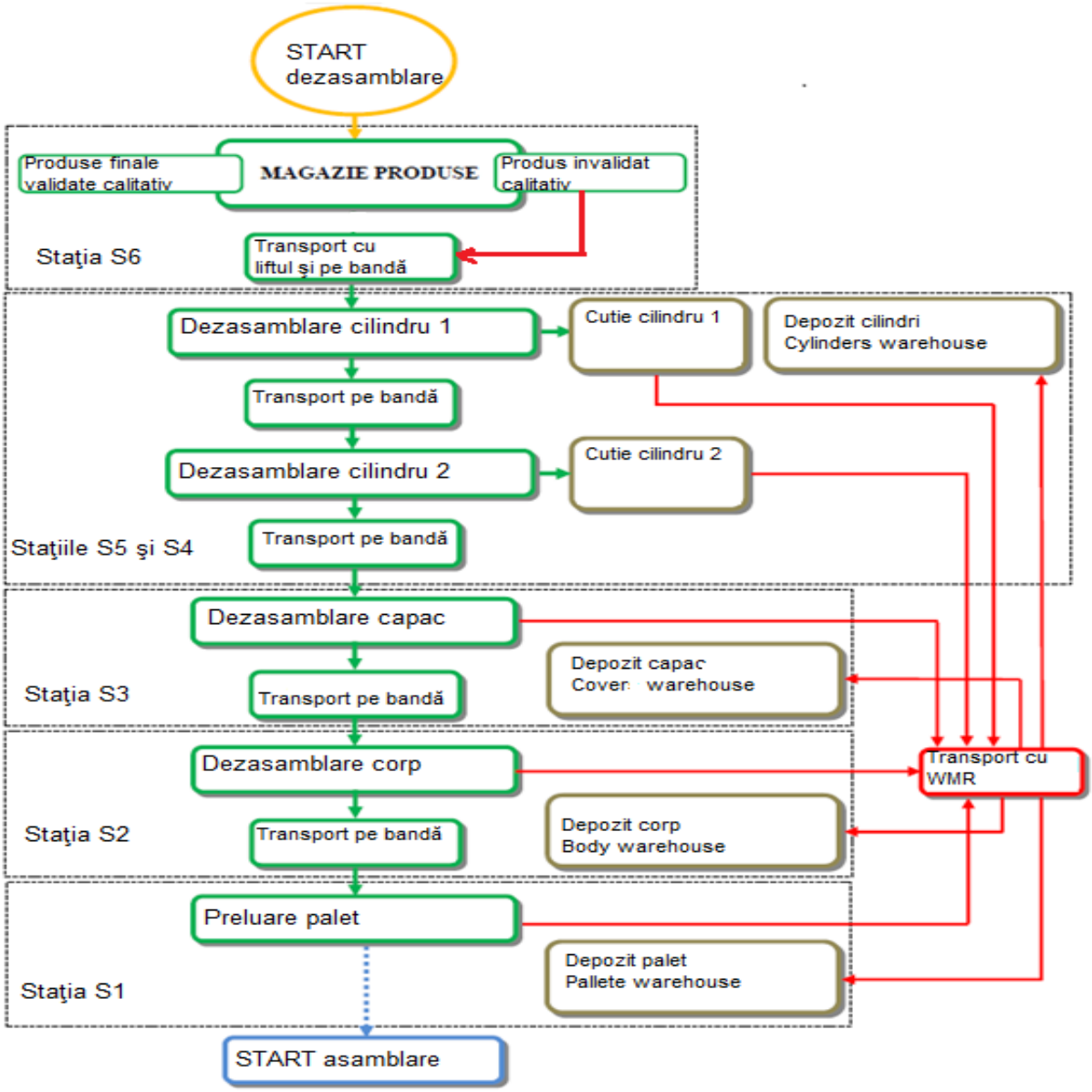


Fig.3.3. Procesul de dezasamblare pe FMML, A/DML HERA&HORSTMANN deservită de WMR echipat cu RM

- START ciclu elementar WMR echipat cu RM (poziţionare, apucare componentă, transport la locaţie depozitare componentă, revenire la următoarea locaţie de dezasamblare) şi START bandă transportoare staţie S5;
- transport C1+C2+C3 pe benzile transportoare ale staţiilor S5 şi S4, până la staţia S3, unde se află următoarea locaţie de dezasamblare;
- STOP bandă transportoare staţie S3 şi START dezasamblare capac;
- START ciclu elementar WMR echipat cu RM (poziţionare, apucare componentă, transport la locaţie depozitare componentă, revenire la următoarea locaţie de dezasamblare) şi START bandă transportoare staţie S3;
- transport C1+C2 pe banda transportoare a staţiei S3, până la staţia S2, la următoarea locaţie de dezasamblare;
- STOP bandă transportoare staţie S2 şi START dezasamblare corp;

- START ciclu elementar WMR echipat cu RM (poziţionare, apucare componentă, transport la locaţie depozitare componentă, revenire la următoarea locaţie de dezasamblare) şi START bandă transportoare staţie S2;
- transport C1pe banda transportoare a staţiei S2, până la staţia S1, la următoarea locaţie de dezasamblare;
- STOP bandă transportoare staţie S1 şi START ciclu elementar WMR echipat cu RM (poziţionare, apucare palet, transport la locaţie depozitare palet, revenire la prima locaţie de dezasamblare);
- START o nouă asamblare;

3.3.2. Planificarea taskurilor pentru A/DML deservită de un WMR echipat cu RM

Proceselor de A/D le se asociază taskuri, care eventual se pot executa în paralel, corespunzând poziţionarii piesei de-a lungul A/DML, a deplasării WMR şi a operaţiilor de pozitionare ale RM, asa cum este prezentat în [6], [26], [29], [35], [43], [52], [64], [65], [75] şi [76]. Strategia hibridă ce corespunde procesului de dezasamblare este bazată pe modelul ierarhic prezentat în [46] şi [75], care utilizează o reprezentare grafică a produsului dezasamblat unde cuplajele dintre componente sunt reprezentate prin săgeţi, Fig.3.4. Utilizând acest graf se elaborează o planificare a taskurilor prin care se determină secvența după care componentele sunt dezasamblate și transportate la magaziile de depozitare, [34], [35], [38], [51], [52].

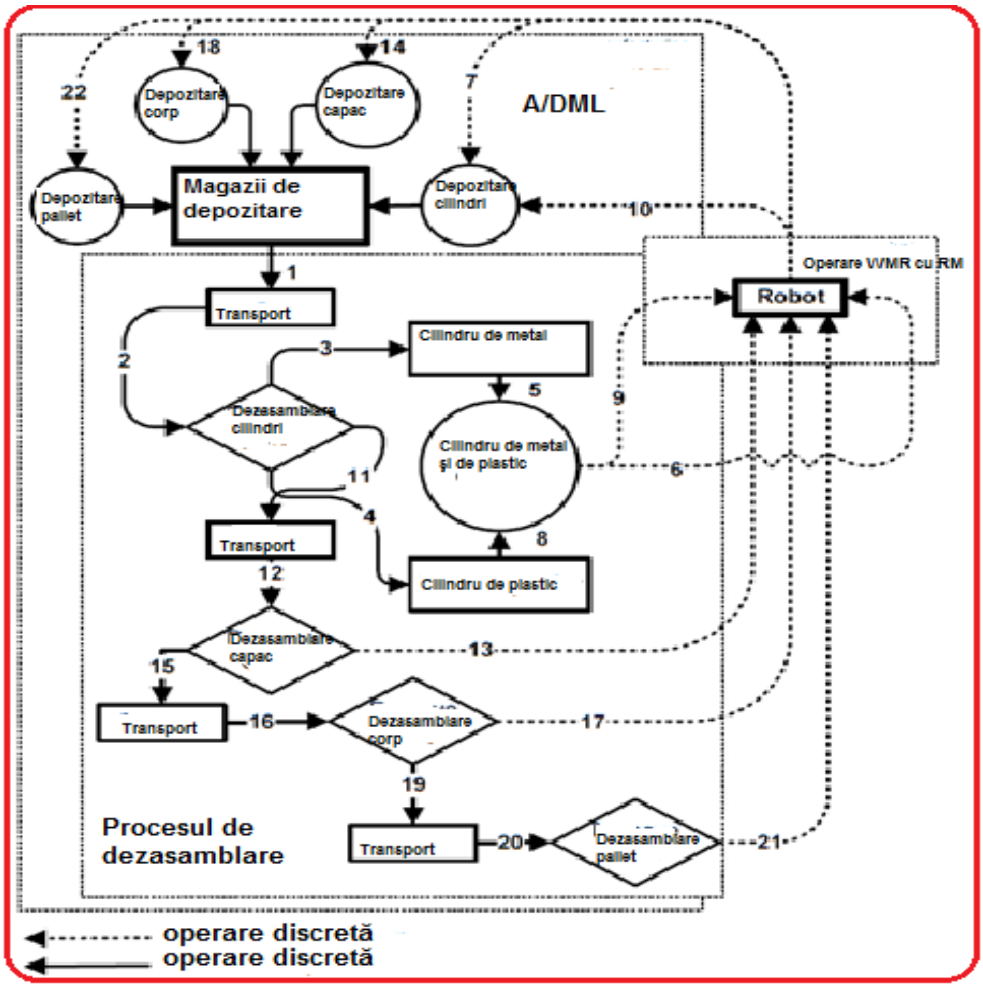


Fig.3.4. Planificarea taskurilor asociate procesului de dezasamblare deservit de un WMR echipat cu RM

În teză, se propune o stategie hibridă bazată pe modelul ierarhic propus în [61], [62], [63] şi [66]. Reprezentările din Fig.3.5, Fig.3.6, Fig.3.7 si Fig 3.8. prezintă împărțirea pe zone, distanţele parcurse de WMR şi operaţiile de poziţionare ale RM de-a lungul A/DML, HERA&HORSTMANN. WMR transportă componenta, rezultată din dezasamblare, de la locaţia unde se produce dezasamblarea la magazia de depozitare corespunzătoare. La fiecare staţie de lucru, RM preia componenta rezultată din dezasamblare şi o depune la magazia de depozitare după ce WMR parcurge distanţele şi se poziţionează în dreptul locaţiilor corespunzătoare. Astfel, componentele pot fi reintroduse în procesul de asamblare.

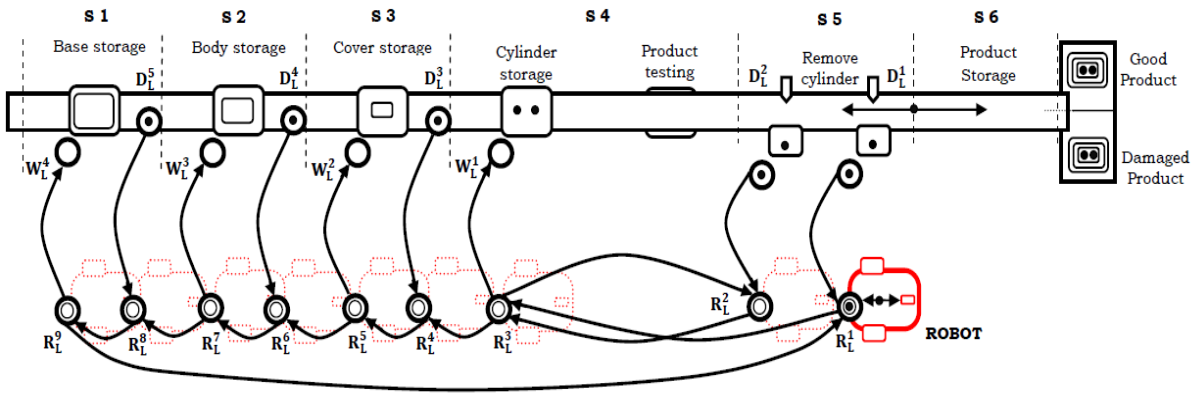


Fig.3. 5. Ciclul complet al WMR echipat cu RM

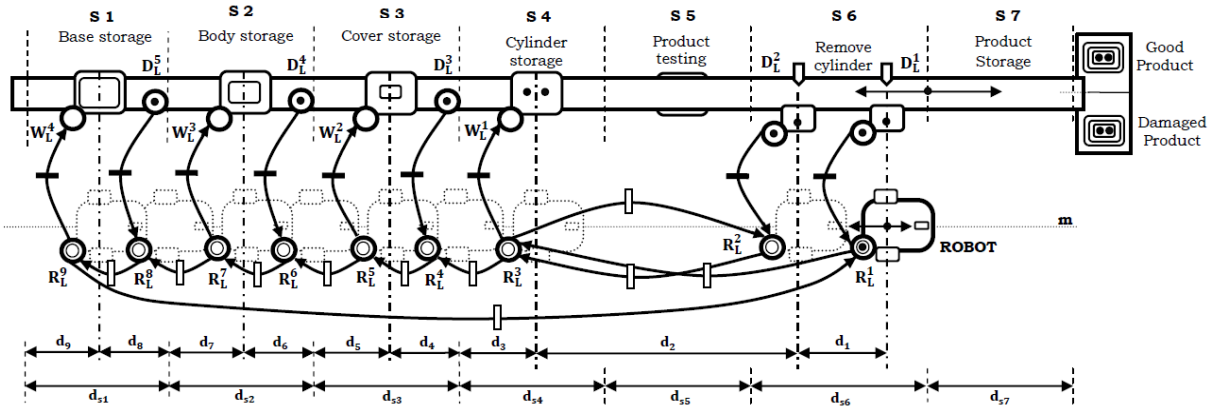


Fig.3. 6. Distanţele parcurse de WMR, poziţionări RM

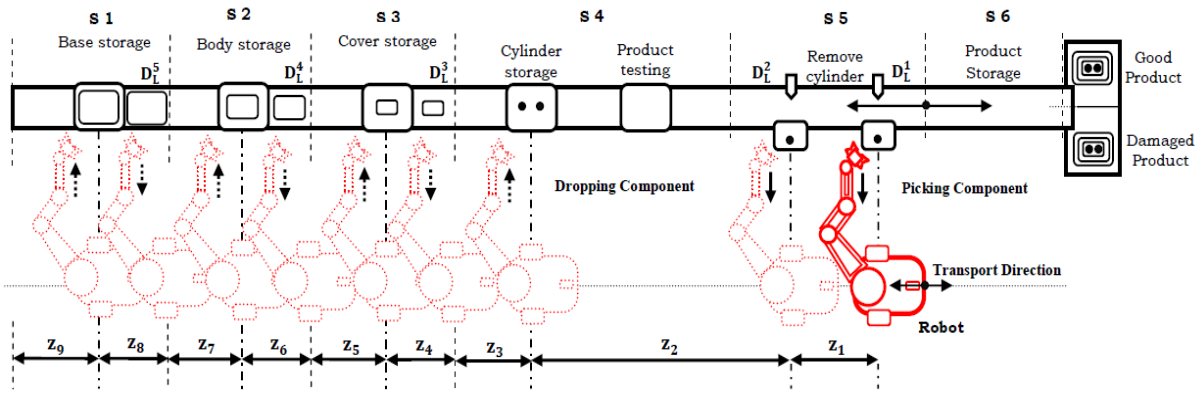


Fig.3.7. Acţiuni pe zone ale WMR echipat cu RM

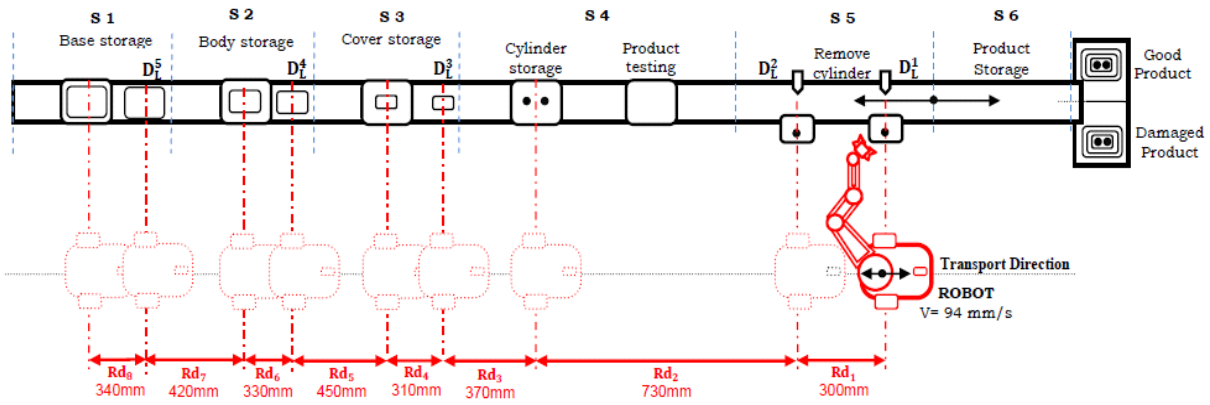


Fig.3.8. Distanţele parcurse de WMR

- R_L^1, …, R_L^9 : numărul și poziția locațiile în care se găsește WMR echipat cu RM în cadrul procesului de dezasamblare;
- D_L^1 : numărul și locația unde se dezasamblează cilindrul numărul 1;
- D_L^2 : numărul și locația unde se dezasamblează cilindrul numărul 2;
- D_L^3 : numărul și locația unde se dezasamblează capacul P3;
- D_L^4 : numărul și locația unde se dezasamblează corpul P2;
- D_L^5 : numărul și locația unde se dezasamblează paletul P1;
- W_L^1 : numărul și locația unde se depozitează cilindrul numărul 1;
- W_L^2 : numărul și locația unde se depozitează cilindrul numărul 2;
- W_L^3 : numărul și locația unde se depozitează capacul P3;
- W_L^4 : numărul și locația unde se depozitează corpul P2;
- W_L^1 : numărul și locația unde se depozitează paletul P1;
- Z_1, …, Z_9 : numărul și zonele aferente fiecărei stații la A/DML HERA&HORSTMANN pe care WMR echipat cu RM trebuie să le parcurgă în cadrul procesului de dezasamblare.

3.4. Particularizare la FMML, A/DML HERA&HORSTMANN, deservită de doi WMRs

3.4.1. Descriere hardware

A/DML HERA&HORSTMANN este deservită de doi WMRs, unul dintre ei echipat cu RM, este utilizat pentru manipulare, celălalt fiind utilizat pentru transport. Ambii WMRs deservesc A/DML pe durata procesului de dezasamblare. Utilizarea a doi WMRs își are motivaţia într-o mai mare flexibilitate în manipularea şi transportul componentelor provenite din dezasamblare, în ceea ce priveşte intervale mai mari de variaţie ale greutăţilor şi dimensiunilor. Şi în acest caz, scopul este de a face A/DML reversibilă, adică să permită și dezasamblare. Se formuleaza ipoteze şi taskuri în cazul general, Fig.3.10, ca apoi să se facă particularizare la A/DML, HERA&HORSTMANN care asamblează o piesă din 5 componente, aşa cum este aratat în Fig.3 9. Cei doi WMRs, Pioneer3-DX echipat cu RM Pioneer 5-DOF Arm și PatrolBot au încorporate on-board microcontrolere care sunt în măsură să furnizeze informația despre poziție pe care o transmite, via WI-FI link, la un PC aflat la distanţă pe care rulează şi algoritmul de conducere. Calculatorul aflat la distanță calculează comanda pe care o transmite celor doi WMRs. Comanda manipulatorului se face în buclă deschisă, eventual prin intermediul unui sistem servoing visual. De asemenea, PC-ul comunică și transmite date PLC-ului (SIEMENS SIMATIC S7-300) cu care este echipat A/DML, PLC care are 5 module

distribuite conectate la magistrala Profibus. Reprezentarea schematică generală a FMML, A/DML cu *N* stații de lucru de A/D, este prezentată în Fig.3.10 iar în Fig.3.11 este prezentată FMML, A/DML, HERA&HORSTMAN, deservită de doi WMRs, lucrând în paralel, colaborativ și sincron. La fiecare stație de lucru, WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, manipulează componenta eliberată prin dezasamblare și o poziționează pe WMR transportor, PatrolBot, pentru a fi transferată şi depozitată la magazia corespunzătoare, cu scopul de a fi reutilizată în procesul de asamblare.

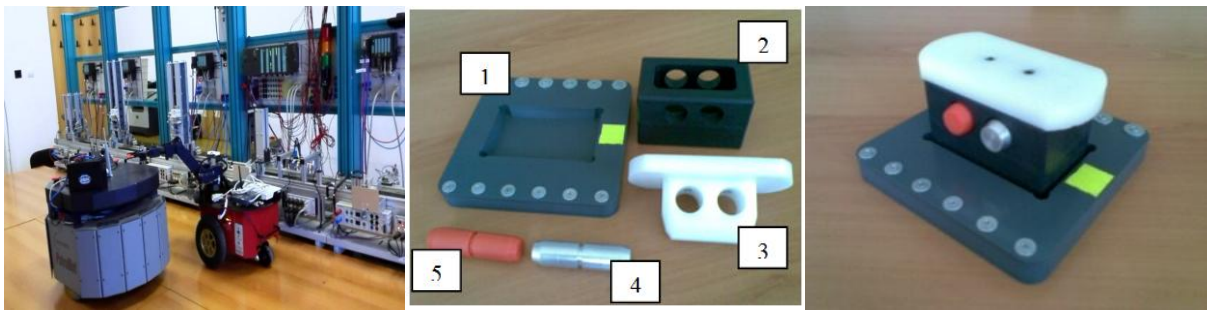


Fig 3.9. A/DML, HERA&HORSTMANN dervită WMRs: Pioneer 3-DX, chipat cu RM, Pioneer 5-DOF Arm and PatrolBot; componente; produsul final asamblat

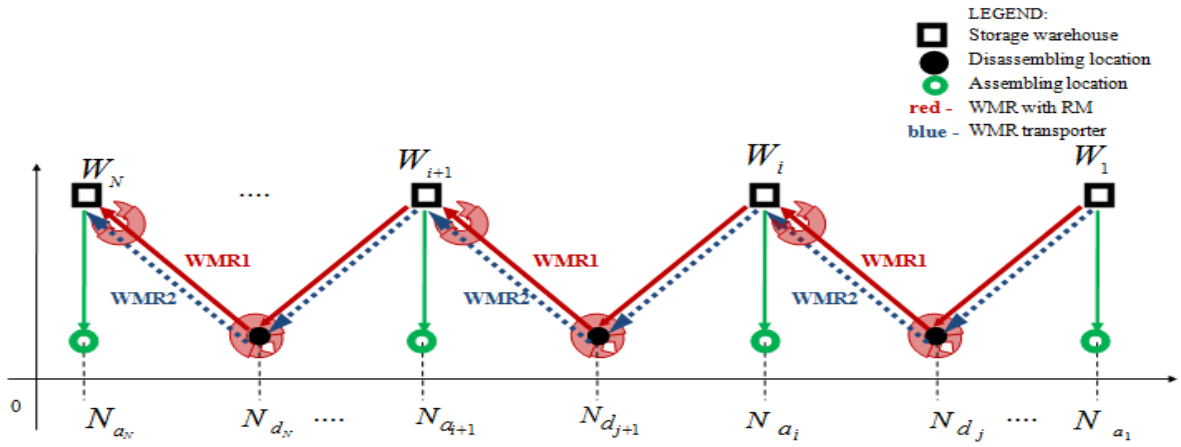


Fig.3.10. Reprezentarea A/DML cu *N* componente de A/D şi magazii de depozitare

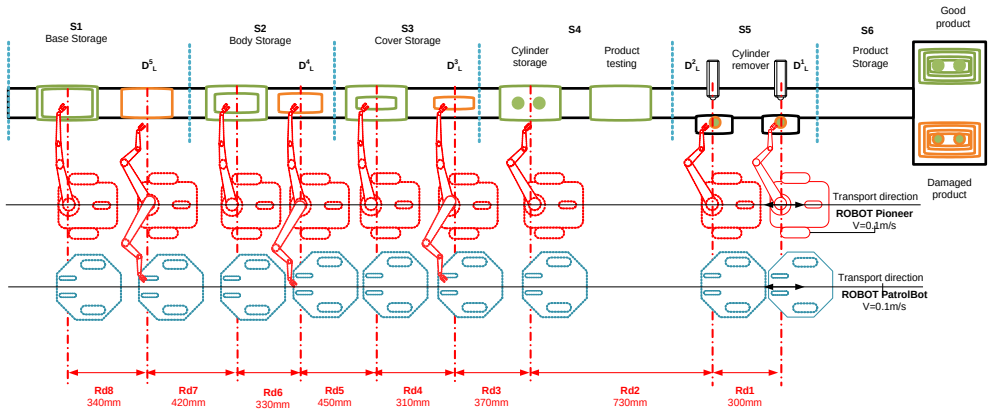


Fig.3.11. A/DML cu 5 componente, deservită de doi WMRs, lucrând în paralel, unul dintre ei echipat cu RM

3.4.2. Atribuire, planificare taskuri şi A/DLB

În aceasta secțiune se atribuie și se propune o planificare de taskuri pentru A/DML deservită de doi WMRs, unul echipat cu RM este destinat manipulării componentelor, iar celălalt este destinat transportului. Ipotezele de mai sus, de la ***I.1*** până la ***I.14***, stau la baza modelului ce corespunde procesului de dezasamblare care se constituie într-un criteriu de optimizare (echilibrare), a cărui extremizare (maximizare) reprezintă o soluţie a problemei A/DBL. Și aici, datorită ipotezelor **I.1, I. 2** şi **I.10**, procesul de asamblare poate fi considerat implicit optimizat (echilibrat), deci problema ALB este implicit rezolvată.

Fie M numărul total de taskuri necesare dezasamblarii unui produs şi M_c numărul de taskuri pe ciclu elementar (perioadă). Taskurile asociate unui ciclu elementar, TC_i, $i = \overline{1,M_c}$, sunt definite mai jos:

TC_1 – Transportul produsului (pe banda transportoare) supus procesului de dezasamblare, de-a lungul A/DML în sensul negativ al axei Ox;

TC_2 – Stop A/DML și dezasamblare componentă;

TC_3 – Poziționare RM (pentru prindere componentă) al WMR1 (poziționat în dreptul locaţiei unde se produce dezasamblarea);

TC_4 – Prindere componentă;

TC_5 – Start A/DML;

TC_6 – Manipulare RM pentru depunere componentă pe WMR2 (utilizat ca transportor);

TC_7 – Poziționare RM al WMR1 pentru deplasare la următoarea locație;

TC_8 – WMR1 cu RM deplasare de la locația de dezasamblare la locația magaziei de depozitare;

TC_9 – Deplasare WMR2 de la locația de dezasamblare la locația magaziei de depozitare;

TC_{10} – Poziționare RM al WMR1 și prindere componentă de pe WMR2;

TC_{11} – Pozitionare RM al WMR1 pentru depozitare componentă la magazia corespunzătoare;

TC_{12} – Depozitare componentă;

TC_{13} – Poziţionare RM al WMR1 pentru deplasare la locația următoare;

TC_{14} – Deplasare WMR1 de la locație magazie de depozitare la următoarea locație de desasamblare;

TC_{15} – Deplasare WMR2 de la locație magazie de depozitare la următoarea locație de dezasamblare.

Planificarea taskurilor este prezentată în Tab.3.1.

Observația 3.2. Deoarece WMR1 și WMR2, se deplasează sincron, patru taskuri se execută în paralel.

Tab. 3.1. Planificare taskuri

	TC2	TC5	TC1						
TC3	TC4	TC6	TC7	TC8	TC10	TC11	TC12	TC13	TC14
				TC9					TC15

Cu planificarea taskurilor, definită mai sus, criteriul de optimizare (3.6), în condițiile de restricții (3.2), (3.3), (3.4) și (3.5) ramâne valabil și în acest caz când A/DML este deservită de doi WMRs, lucrând în paralel, cu deplasare sincronă.

3.5. WMR echipat cu RM integrat în FMML, P/RML FESTO MPS-200

3.5.1. Descrierea hardware a FMML, P/RML FESTO MPS-200

Configurația P/RML, FESTO MPS-200, Fig.3.12, conţine 4 stații de lucru care procesează o piesă, Fig.3.13, prin execuția mai multor taskuri cum ar fi: manipulare, sortare, depozitare, găurire, alezare, acumulare. Două sisteme servoing vizuale sunt plasate pe prima şi ultima stație de lucru, prin intermediul cărora se execută operaţiunile de preluare şi depozitare piesă. Fiecare stație este echipată cu un PLC, SIEMENS SIMATIC S7-300. La stadiul actual, în teză, se consideră că P/RML este deservită de un singur WMR, Pioneer 3DX, echipat cu RM, Pioneer5-DOF Arm, Fig.3.12, 3.13, 3.14.

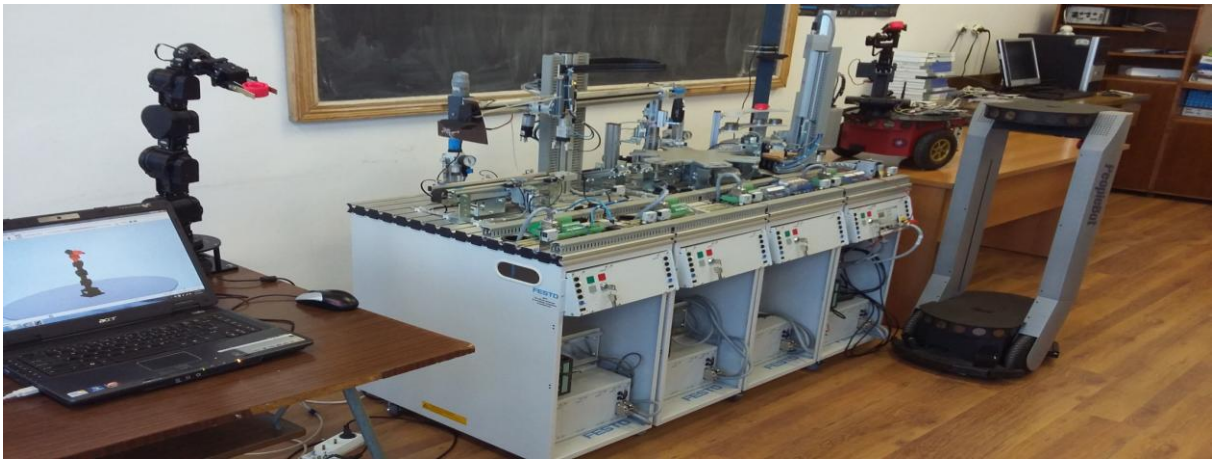


Fig.3.12. FMML, P/RML FESTO-MPS 200 deservită de WMRs, Pioneer 3-DX, PeopleBot, și RMs, Pioneer 5-DOF Arm și Cyton 1500

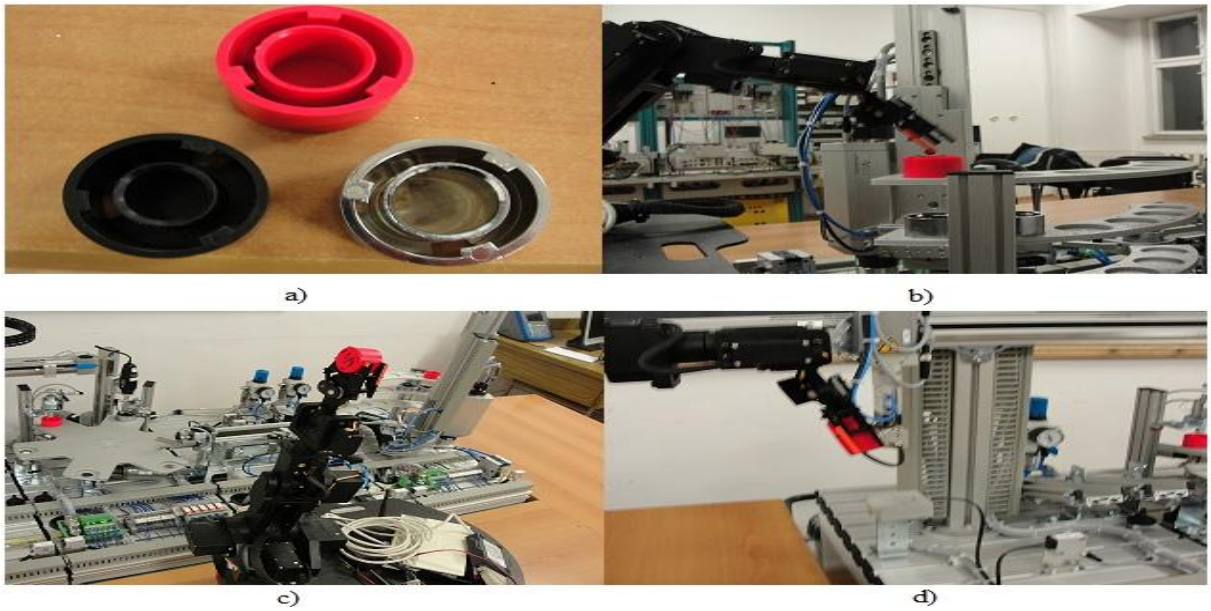


Fig.3.13. a) piese de lucru ; b), c) şi d) WMR, Pioneer P3-DX, echipat RM, Pioneer 5-DOF, deservind P/RML, FESTO MPS-200

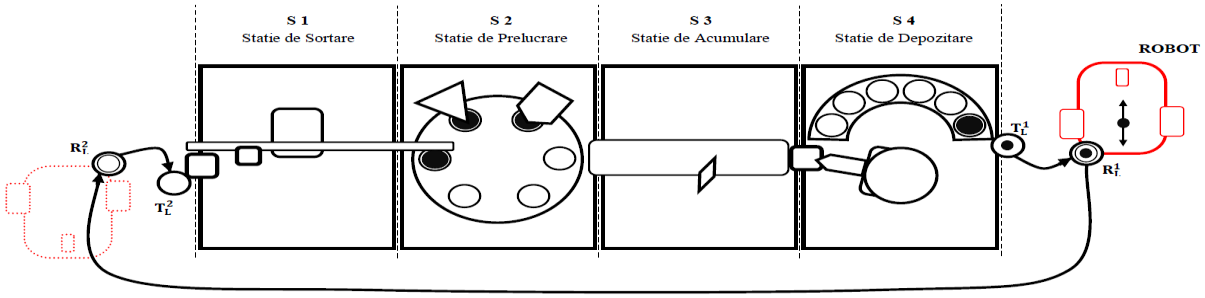


Fig.3.14. Pioneer 3-DX integrat în P/RML FESTO MPS-200

Se consideră următoarele operații de P/R pe FMML, FESTO MPS-200:

- prezenţă piesă staţia S1 (manipulare/sortare) declanşează START FMML FESTO MPS-200;
- test culoare piesă;
- transport piesă neagră la magazia de piese rebut;
- transport piesă roşie sau gri la staţia S2 (prelucrare);
- START S2, transport piesă la locaţia de alezare;
- operaţie alezare;
- transport piesă la locaţia de găurire;
- operaţie găurire;
- transport piesă staţia S3 (acumulare);
- transport piesă la staţia S4 pe banda transportoare a staţiei S3;
- test culoare;
- poziţionare manipulator axial, staţia S4;
- apucare piesă manipulator axial;
- transport şi depozitare piesă de culoare gri pe rafturile inferioare;
- transport şi depozitare piesă de culoare roşie pe raftul superior;
- START WMR echipat cu RM;
- START sistem servoing vizual staţie S4 şi poziţionare RM;
- apucare piesă;
- START ciclu deplasare WMR echipat cu RM;
- START sistem servoing visual staţie S4 şi poziţionare RM;
- eliberare piesă;
- START ciclu revenire la poziţia iniţială a WMR echipat cu RM.

Observaţia 3.1. La stația buffer sunt acumulate piese în vederea depozitării pe rafturile staţiei S4.

Observaţia 3.2. În urma acumulării pieselor la stația buffer, acestea sunt eliberate una câte una pentru a fi preluate cu ajutorul unui manipulator axial în vederea depozitării pe rafturi. Acumulându-se piese de diferite culori la stația buffer, la operația de depozitare, acestea sunt sortate după culoare.

Observaţia 3.3. Piesele stocate pe nivelul superior sunt considerate piese care nu satisfac parametrii de calitate și, prin urmare, vor fi aduse la prima stație, unde, supuse unui nou test, vor urma un proces de reprelucrare sau vor fi rebutate definitiv. Readucerea pieselor la prima stație se face cu WMR, iar poziționarea RM, pentru prindere şi eliberare piesă, se realizează cu sistem servoing vizual.

În Fig.3.15, sunt prezentate secțiunile de traiectorie și distanțele pe care le parcurge WMR echipat cu RM pentru a deservi FMML, P/RML FESTO MPS-200.

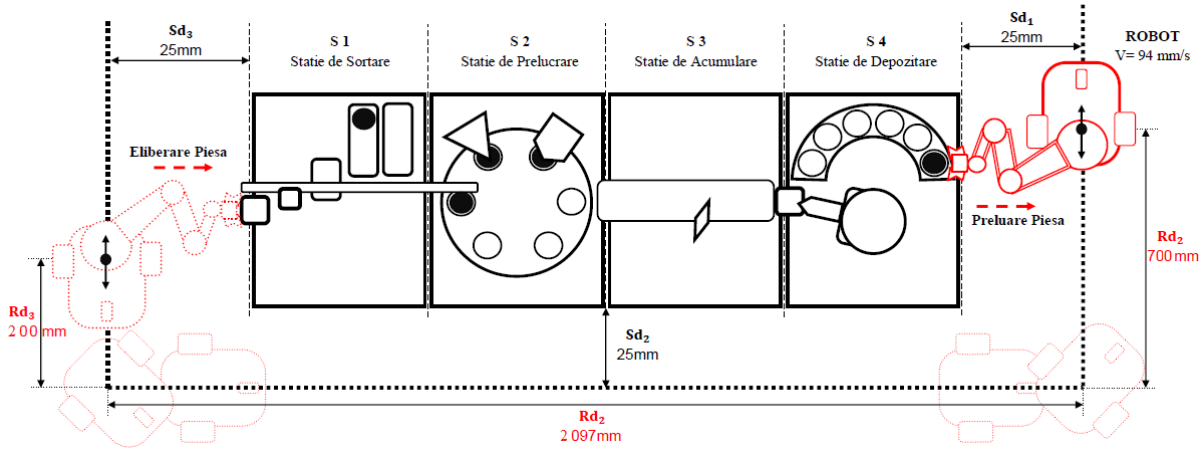


Fig.3.15. Secțiunile de traiectorie și distanțele parcurse de WMR

- R_L^1: numărul și poziția locației unde se află WMR echipat cu RM în cadrul procesului de preluare a piesei;
- R_L^2: numărul și poziția locației în care se află WMR echipat cu RM în cadrul procesului de depunere a piesei;
- T_L^1: numărul locației unde se stochează piesa în vederea preluării acesteia de către WMR echipat cu RM;
- T_L^2: numărul locației unde se depune piesa de către WMR echipat cu RM în vederea unei noi operații de prelucrare.

3.5.2. Ipoteze și planificare taskuri

I.1. În ceea ce privește natura produsului, P/RML este o linie de fabricație unimodel, deterministă, după timpii de operare (taskurile au timpi de execuției bine stabiliți) și "paced-line", după modul sincronizat cum se face transferul piesei de la o stație la alta;
I.2. FMML, P/RML are un număr fix de stații;
I.3. Reprocesarea se aplică aceluiași tip de produs;
I.4. Se consideră că procesele de P/R sunt complete în toți parametrii, adică, sunt cunoscuți cu exactitate: timpii de execuție a taskurilor, cererea de piese, costurile;
I.5. P/RML are 4 stații de lucru, așezate în linie, iar piesa supusă reprocesării este preluată de prima stație;
I.6. Fiecărui task i se specifică timpul de execuție;
I.7. În procesele de P/R este implicată o singură piesă;
I.8. Prin convenție se stabilește că piesa finală care nu satisface parametrii de calitate este de culoare roșie.

Conform ipotezelor de la ***I.1*** până la ***I.8***, se propune planificarea taskurilor aferente liniei flexibile FMMl, P/RML FESTO MPS-200, ilustrată în fig.3.16.

Strategia de sortare, prelucrare și depozitare este bazată pe un graf de reprezentare a produsului prelucrat în care relațiile dintre stații sunt exprimate prin săgeţi, [15]. Utilizţnd acest graf se elaborează o planificare a taskurilor prin care se determină secvența de P/R. Dacă o componentă este validată sau nu la testul de culoare, planificarea taskurilor furnizează cea mai bună secvență pentru efectuarea prelucrării și depozitării acesteia.

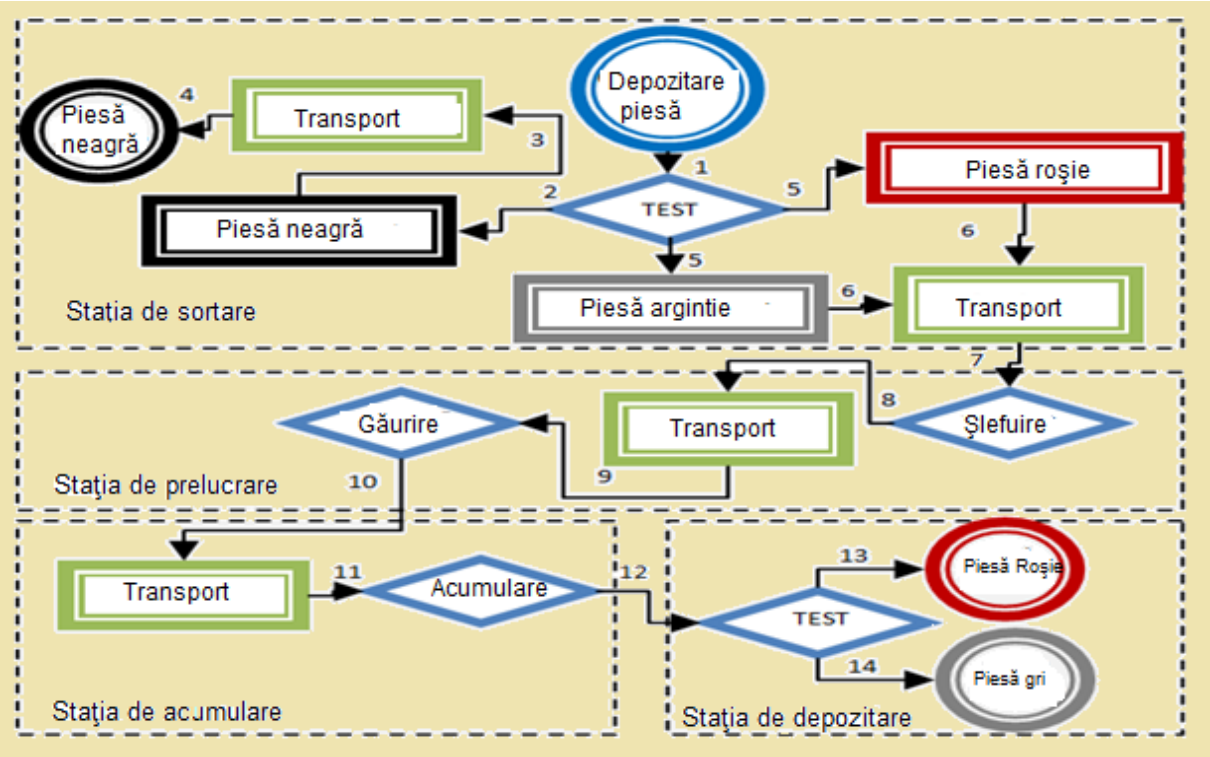


Fig.3.16. Planificarea taskurilor pentru sortare, alezare, găurire, acumulare și depozitare

3.6. Concluzii

Rolul acestui capitol a fost stabilirea premizelor, ipotezelor, taskurilor şi condiţiilor de funcţionare în vederea modelării şi conducerii FMMLs-urilor deservite de sisteme robotice mobile echipate cu manipulatoare. Pentru A/DML s-a propus o abordare generală, cu N staţii de lucru, abordare care permite şi stabilirea unui criteriu de optimizare, a cărui maximizare este echivalentă cu echilibrarea liniei de fabricaţie. Asignarea şi planificarea taskurilor sunt etape indispensabile în acţiunea de automatizare a tehnologiilor de fabricaţie flexibilă.

Contribuţiile revendicate ar fi următoarele:

- Ipotezele de lucru şi de funcţionare aferente FMML, A/DML cu N staţii de lucru deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare;
- Ipotezele de lucru şi de funcţionare aferente FMML, P/RML deservită de un sistem robotic mobil echipat cu manipulator;
- Asignarea si planificarea taskurilor pentru operaţiile derulate pe A/DML cu N staţii de lucru deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare;
- Echilibrarea A/DML cu N staţii de lucru, deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare, prin rezolvarea unei probleme de optimizare cu restricţii;
- Asignarea si planificarea taskurilor pentru operaţiile derulate pe A/DML HERA&HORSTMANN deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare;
- Asignarea si planificarea taskurilor pentru operaţiile derulate pe P/RML FESTO MPS-200 deservită de un sistem robotic mobil echipat cu manipulator.

Capitolul 4

Modelarea hibridă și simularea FMMLs, A/DML și P/RML, deservite de WMRs echipați cu RMs

Acest capitol este dedicat modelării hibride și testării modelelor asociate FMMLs deservite de WMRs echipați cu RMs prin simulare. Se prezintă modelul SHPN și testarea prin simulare în Sirphyco pentru A/DML deservită de un WMR echipat cu RM, în formă generală și particularizată. De asemenea, se prezintă modelul SHPN și simularea în Sirphyco a două sisteme robotice mobile integrate în A/DML. Tot aici sunt prezentate modelul SHPN și simularea în Sirphyco pentru P/RML deservită de un WMR echipat cu RM, unde sincronizarea se face și prin intermediul unui sistem servoing vizual cu poziție fixă sau mobilă.

4.1. Modelul SHPN asociat A/DML deservită de un WMR echipat cu RM

4.1.1. Structura modelului SHPN

A/DML deservită de roboți mobili au caracteristici hibride, având două dinamici, una continuă și cealaltă discretă, bazată pe evenimente. HPNs sunt instrumente de modelare a unor astfel de sisteme. Aspectul hibrid al modelului este determinat de variabilele asociate distanțelor parcurse de WMR, distanțe considerate între locaţiile unde se produc operaţiile de dezasamblare şi locațiile magaziilor de depozitare. Variaţia acestor variabile este în concordanţă cu viteza de deplasare a WMR între locaţiile A/DML. Pentru a dezvolta un model pentru A/DML deservită de WMR echipat cu RM, se va considera aspectul hibrid al proceselor de A/D. Pentru modelare se va apela la instrumentul THPN, [1], [14], care integrează aspectul discret al proceselor de A/D cu aspectul continuu al mișcării WMR-ului și manipulării componentelor de catre RM. Modelul global este de tip SHPN deoarece este interfaţat cu evenimente externe pentru sincronizare, evenimentele ne fiind altceva decât semnale provenite de la senzori. Aceste evenimete sunt utile atât la modelare/simulare cât şi la conducerea în timp real. Structura SHPN, din Fig.4.1, şi reprezentarea pe blocuri, din Fig.4.2, corespunde modelării discrete a proceselor de A/D şi a dinamicii continue a WMR echipat cu RM, care deserveşte A/DML în procesul de dezasamblare. Structura internă a modelului SHPN integrează trei modele PN, fiecare dintre ele având o tipologie specifică: TPN (Rețea Petri Temporizată), SPN (Rețea Petri Sincronizată) si THPN (Rețea Petri Hibridă Temporizată).

Aceste modele descriu următoarele operații care se execută automat:

- Asamblare/depozitare în magazii care este o tipologie TPN;
- Dezasamblare produs asamblat care nu a trecut testul de calitate, de tipologie SPN și TPN;
- Integrarea WMR echipat cu RM în A/DML în procesul de dezasamblare, tipologie THPN.

4.1.2. Secvențe repetitive de A/D

În Fig.4.2 se poate observa un bloc de tipul e-TPN care corespunde unei singure operații de asamblare. Modelul TPN asociat acestui bloc este prezentat în Fig.4.3. Acest model elementar se repetă pentru fiecare operație de asamblare. În Fig.4.4 este prezentat modelul TPN generalizat corespunzător întregului proces de asamblare, proces care include secvențe repetitive, asamblarea celor doi cilindri și testul de material, care s-a convenit a fi asimilat testului de calitate. De asemenea o secvență repetitivă se poate evidenția și în decursul procesului de dezasamblare, secvență care conține operația de dezasamblae și asistența WMR echipat cu RM. Secvența repetitivă din cadrul procesului de dezasamblare poate fi modelată ca o secvență elementară SHPN, e-SHPN, și este marcată în Fig.4.2. Se poate observa pe figură că modelul e-SHPN, conține două submodele, e-THPN asociată WMR și e-SPN+TPN asociată A/DML în procesul de dezasamblare. În Fig 4.5, este redat modelul e-SHPN corespunzător primei operaţii de dezasamblare, model care va fi testat prin simulare, [62], [63].

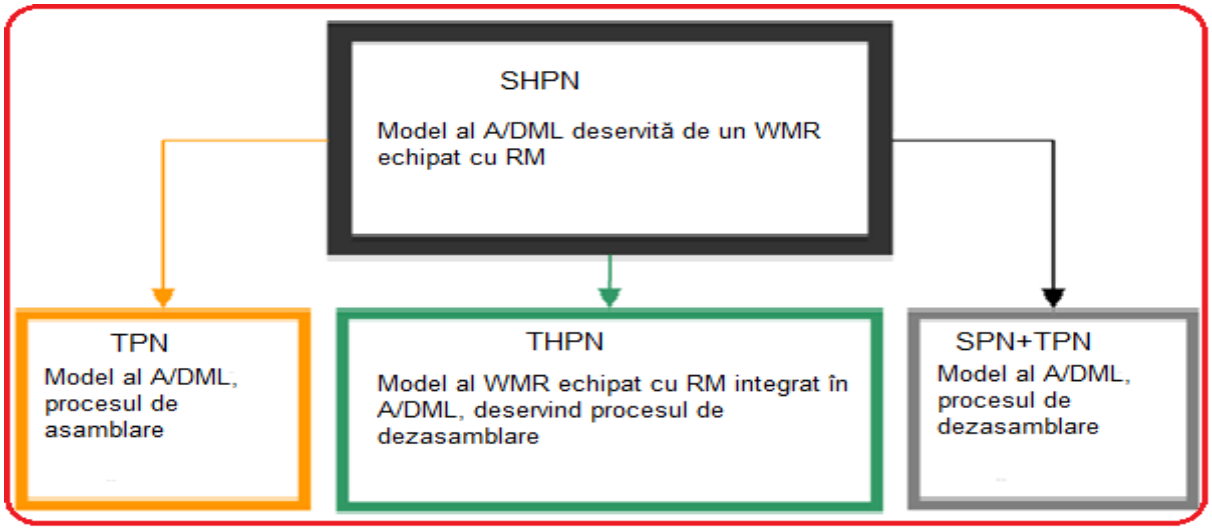


Fig.4.1. Structura modelului SHPN asociat A/DML deservită de un WMR echipat cu RM

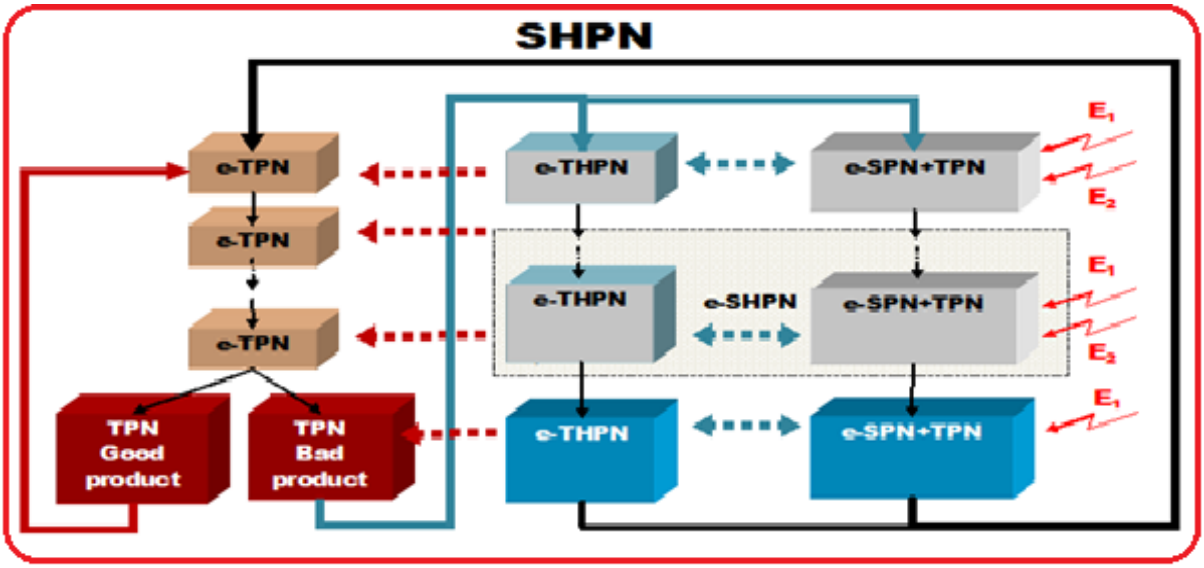


Fig.4.2. Modelul SHPN cu module elementare: e-TPN asamblare, e-THPN WMR echipat cu RM, e-SPN+TPN dezasamblare, e-SHPN dezasamblare deservită de WMR echipat cu RM

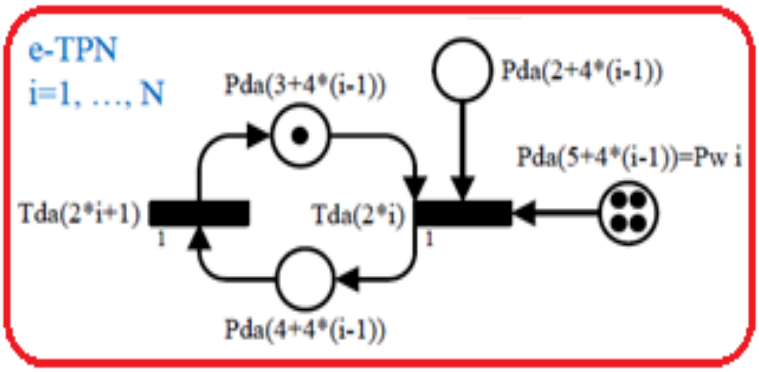


Fig.4.3. Modelul e-TPN pentru o operaţie elementară de asamblare

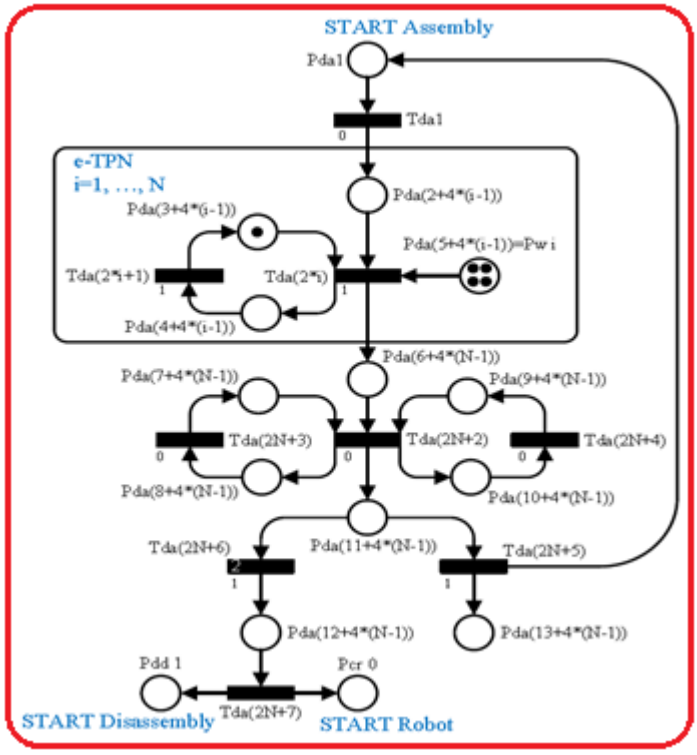


Fig.4.4. Modelul TPN generalizat asociat asamblării a N componente

Tot în Fig.4.2, sunt marcate evenimentele externe provenite de la senzori $E^1_{dd(j)}$ și $E^2_{dd(j+2)}$, evenimente utilizate pentru sincronizarea A/DML cu platforma robotică, WMR echipat cu RM. $E^1_{dd(j)}$ este un semnal de sincronizare extern care are drept efect “STOPPING” A/DML și “STARTING” dezasamblare. $E^2_{dd(j+2)}$ este un semnal de sincronizare extern care are drept effect “PICKING UP” componentă rezultată din operația de dezasamblare și “STARTING” A/DML.

Testarea prin simulare a modelului SHPN, fară semnalele de sincronizare, adică simularea în mod autonom a modelului HPN, este utilă pentru a conecta dinamica discretă a A/DML cu dinamica continuă a WMR echipat cu RM. Deoarece, după ultima operaţie de dezasamblare nu se mai produce o alta, modelul SHPN corespunzător, Fig.4.6, este diferit de modelul implicat în procesul repetitiv. Şi acest model va fi testat prin simulare. Prin concepţie, modelul SHPN, produce oprirea liniei în cazul apariţiei unor evenimente nedorite sau

incertitudini, cum ar fi: defectarea unor senzori, semnale recepţionate defectuos sau defecţiuni în funcţionarea A/DML şi/sau WMR echipat cu RM. În situţia apariţiei unor astfel de evenimente, FMML se opreşte şi îşi reia funcţionarea după ce defectul este remediat.

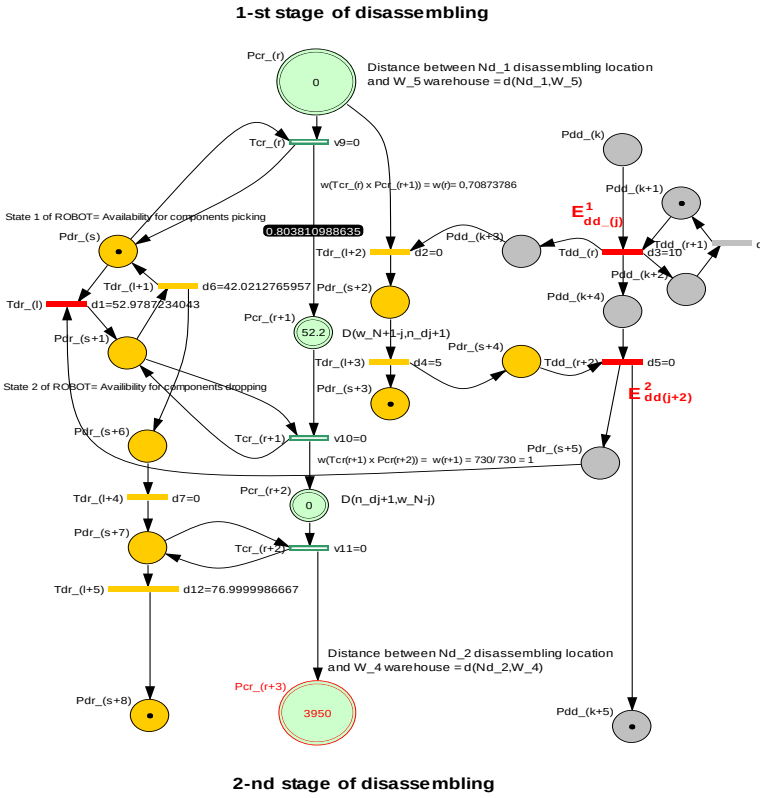


Fig.4.5. Modelul e-SHPN pentru prima operaţie elementară de dezasamblare $j=1$

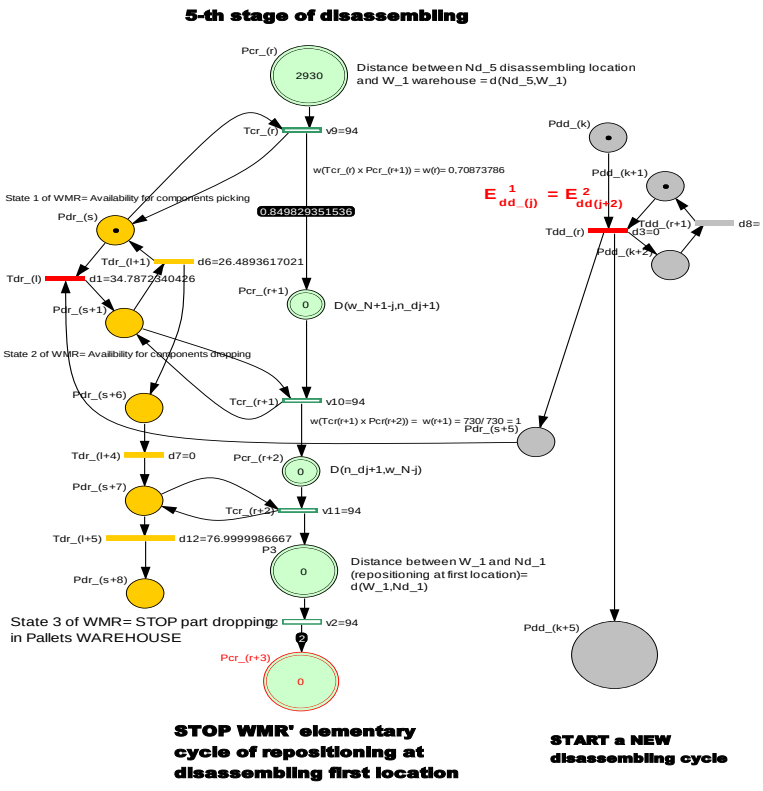


Fig.4.6. Modelul e-SHPN pentru ultima operaţie elementară de dezasamblare $j=N$

4.1.3. Formalismul modelării cu SHPN

Modelul SHPN asociat A/DML este un triplet

$$SHPN = \langle THPN,\ \ E,\ \ Sync \rangle \tag{4.1}$$

unde: $THPN$ este un sept-uplu

$$THPN = \langle P, T, Pre,\ Post,\ m_0, h, tempo \rangle \tag{4.2}$$

E este o mulţime de evenimente externe

$$E = \left\{Edd_i^1, Edd_j^2\right\} \cup \{e\}, i = 1+3(k-1), j = 3(k-1),\ k = \overline{1,N} \tag{4.3}$$

$Sync$ este o aplicaţie de la mulţimea tranziţiilor la mulţimea evenimentelor externe

$$Sync : T \to \left\{E^1, E^2\right\} \cup \{e\} \tag{4.4}$$

unde e este evenimentul cu apariţie continuă (este elementul neutru al monoidului E^*)

$$Sync : \{Tdd_r\}_{r=1+3(k-1),k=\overline{1,N}} \to \left\{E^1, E^2\right\} \tag{4.5}$$

$$Sync : \{Tdd_i\}_{i=3(k-1),\ k=\overline{2,N}} \to \left\{Edd_i^2\right\}_{i=3(k-1),k=\overline{2,N}} \tag{4.6}$$

$$Sync : T \setminus \{Tdd_r\}_{r=\overline{1,3+3(N-1)}} \cup \{Tdr_l\}_{l=\overline{1,4+5(N-1)}} \cup \{Tcr_r\}_{r=\overline{1,3+3(N-1)}} \to e \tag{4.7}$$

$$P = \{P_1, P_2, ...P_n\} = P^D \cup P^C \tag{4.8}$$

este o mulţime finită, nevidă, în care P^D este mulţimea stărilor

$$P^D = \{Pda_i\}_{i=\overline{1,13+4(N-1)}} \cup \{Pdd_r\}_{r=\overline{1,5+5(N-1)}} \cup \{Pdr_s\}_{s=\overline{1,4+8(N-1)}} \tag{4.9}$$

iar P^C este mulţimea stărilor continue

$$P^C = \{Pcr_k\}_{k=\overline{0,3+3(N-1)}} \tag{4.10}$$

Pentru $N = 5$, A/DML Hera&Horstmann, (4.9) şi (4.10) devin

$$P^D = \{Pda_i\}_{i=\overline{1,29}} \cup \{Pdd_j\}_{j=\overline{1,25}} \cup \{Pdr_k\}_{k=\overline{1,41}} \tag{4.11}$$

$$P^C = \{Pcr_k\}_{k=\overline{0,15}} \tag{4.12}$$

unde: $\{Pda_i\}_{i=\overline{1,29}}$ este mulţimea stărilor discrete din modelul asociat procesului de asamblare;

$\{Pdd_j\}_{j=\overline{1,25}}$ este mulţimea stărilor discrete din modelul asociat procesului de dezasamblare;

$\{Pdr_k\}_{k=\overline{1,41}}$ este mulţimea stărilor discrete din modelul asociat WMR care deserveşte A/DML în decursul procesului de dezasamblare;

$\{Pcr_k\}_{k=\overline{1,15}}$ este mulţimea stărilor continue din modelul asociat distanţelor parcurse de WMR, distanţe corespunzătoare fiecărei operaţii de dezasamblare, necesare pentru a transporta componenta de la locaţia de dezasamblare la locaţia magaziei de depozitare.

$$T = \{T_1, T_2, ..., T_m\} = T^D \cup T^C \tag{4.13}$$

este o mulţime finită, nevidă de tranziţii, la care T^D este mulţimea traziţiilor discrete

$$T^D = \{Tda_i\}_{i=\overline{1,7+2N}} \cup \{Tdd_r\}_{r=\overline{1,3+3(N-1)}} \cup \{Tdr_l\}_{l=\overline{1,4+5(N-1)}} \tag{4.14}$$

iar T^C este mulţimea tranziţiilor continue

$$T^C = \{Tcr_r\}_{r=\overline{1,3+3(N-1)}} \tag{4.15}$$

Pentru $N = 5$ (4.13), (4.14) şi (4.15) devin

$$T^D = \{Tda_i\}_{i=\overline{1,17}} \cup \{Tdd_j\}_{j=\overline{1,15}} \cup \{Tdr_k\}_{k=\overline{1,24}}, \tag{4.16}$$

$$T^C = \{Tcr_k\}_{k=\overline{1,15}}, \tag{4.17}$$

unde: $\{Tda_i\}_{i=\overline{1,17}}$ este mulţimea tranziţiilor discrete din modelul asociat procesului de asamblare; $\{Tdd_j\}_{j=\overline{1,15}}$ este mulţimea tranziţiilor discrete din modelul asociat operaţiilor de dezasamblare; $\{Tdr_k\}_{k=\overline{1,24}}$ este mulţimea tranziţiilor discrete din modelul associat WMR echipat cu RM deservind A/DML în decursul procesului de dezasamblare; $\{Tcr_k\}_{k=\overline{1,15}}$ este mulţimea tranziţiilor continue din modelul asociat distanţelor parcurse de WMR corespunzătoare fiecărei operaţii de dezasamblare. Viteza liniară maximă a WMR este asociată acestor tranziţii.

Observaţia 1: Mulţimile P şi T sunt disjuncte, $P \cap T = \varnothing$;

$$Pre \quad : P \times T \to Q_+ \; or \; N \tag{4.18}$$

este aplicaţia incidenţelor de intrare;

$$Post \quad : P \times T \to Q_+ or \; N \tag{4.19}$$

este aplicaţia incidenţelor de ieşire.

Observaţia 2: În definiţiile incidenţelor Pre şi $Post$, N corespunde cazului unde $P_i \in P^D$, iar Q_+ sau R_+ corespund cazului unde $P_i \in P^C$.

$$m_0 : P \to R_+ \; or \; N \tag{4.20}$$

este marcajul iniţial al stărilor;

$$h : P \cup T \to \{D, C\} \tag{4.21}$$

numită "funcția hibridă" care indică în ce măsură fiecare nod este un nod discret (mulțimile P^D și T^D) sau este un nod continuu (mulțimile P^C și T^C),

$$h : P^D \cup T^D \to \{D\}; \; h : P^C \cup T^C \to \{C\} \tag{4.22}$$

$tempo$ este o aplicație de la mulțimea tranzițiilor, T, la mulțimea numerelor raţionale pozitive, inclusiv valoarea zero,

$$tempo \quad : T \to Q_+ \cup \{0\} \tag{4.23}$$

Dacă $T_j \in T^D$, atunci $d_j = tempo(T_j)$ este durata timpului asociat tranziției T_j.

Pentru fiecare tranziție asociată unei operații de asamblare din mulțimea

$$T_a^D = \{Tda_i\}_{i=2k, k=\overline{1,N}} \cup \{Tda_{2(N+1)}\} \tag{4.24}$$

atunci

$$tempo(Tda_i) = d_{da_i} \tag{4.25}$$

unde d_{da_i} reprezintă durata, în secunde, asociată operației de asamblare corespunzătoare.

Pentru fiecare tranziție discretă asociată unei operații de dezasamblare, din mulțimea

$$T_d^D = \{Tdd_r\}_{r=1+3(k-1), k=\overline{1,N}} \tag{4.26}$$

d_{ddr} este durata operației de dezasamblare corespunzătoare.

Pentru fiecare tranziție discretă asociată WMR din mulțimea

$$T_r^D = \{Tdr_l\}_{l=4+5(k-2),k=\overline{2,N}} \tag{4.27}$$

d_{drl} este durata de poziționare a RM pentru a prinde sau a elibera în magazie componenta rezultată din dezasamblare.

Pentru $N = 5$, (4.24) și (4.25) devin:

$$T_a^D = \{Tda_i\}_{i=\{2,4,6,8,10\}} \cup \{Tda_{12}\} \tag{4.28}$$

$$tempo(Tda_i)_{i=\{2,4,6,8,10,12\}} = \{9.5, 9.3, 8.5, 0.5, 4.75, 27.2\} \tag{4.29}$$

unde d_{da_i} reprezintă durata (în secunde) operației de asamblare curentă împreună cu timpul de transport la următoarea operație de asamblare, pentru $i \in \{2,4,6,8,10\}$, împreună cu durata testului de calitate şi transportului cu ajutorul liftului a produsului final asamblat la magazia de depozitare, pentru $i \in \{12\}$. Particularizând, (4.26) și (4.27) devin

$$T_d^D = \{Tdd_r\}_{r=\{1,4,7,10,13\}} \tag{4.30}$$

$$d_{ddr,r=\{1,4,7,10,13\}} = 1 \tag{4.31}$$

unde d_{ddr} este durata operației de dezasamblare curente

$$T_r^D = \{Tdr_l\}_{l=\{4,9,14,19\}} \tag{4.32}$$

$$d_{drl} \in \{5.1, 21.2, 8.9, 7.8\} \tag{4.33}$$

unde d_{drl} este durata poziționării RM pentru a apuca sau a elibera componenta rezultată din dezasamblare.

Dacă $T_{cr} \in T^C$, atunci

$$U_r = \frac{1}{tempo(T_{cr})} \tag{4.34}$$

este rata de flux asociată tranziției continue T_{cr}.

Pentru

$$T^C = \{Tcr_r\}_{r=3+3(k-1),k=\overline{1,N}} \tag{4.35}$$

$$U_{cr_r} = U_r; U_{r\max} = V_r \tag{4.36}$$

unde U_{cr} este este variabila de flux asociată deplasării WMR între locațiile associate procesului de dezasamblare. Se considera viteza medie de mișcare a WMR, $V_r = 94mm/s$. Această viteză, fiind suportată de WMR, permite integrarea WMR echipat cu RM în A/DML.

Definiția 1: Gradul de activare, ED, (ED-enabling degree) a unei tranziții continue ($C-$ transition), T_j, pentru o marcă, m, denumită $ED(T_j, m)$, este gradul de activare a tranziției, T_j, după ce toate arcele de la o stare continuă ($C-$ place) la o tranziție continuă ($C-$ transition), au fost sterse

$$ED(T_j, m) = \min_{P_i \in {}^0T_j \cap P^D} \left\lfloor \frac{m_i}{Pre(P_i, T_j)} \right\rfloor \tag{4.37}$$

Definiția 2: Viteza maximă de aprindere a unei tranziții continue a WMR, T_{cr_r} , este produsul dintre propria rată de flux, U_r , și gradul de activare, ED .

Ca urmare a definiţiilor 1 şi 2, pentru A/DML, în cazul general avem următoarele expresii:

$$ED\left(T_{cr_j}, m_{cr(j+1)}\right) = \{0,1\} \tag{4.38}$$

$$m_{cr(j+1)} = V_j \cdot w\left(Tcr_j \times Pcr_{(j+1)}\right) \tag{4.39}$$

$$w\left(Tcr_r \times Pcr_{r+1}\right) = D\left(W_{N+1-j}, N_{d_{j+1}}\right) / D\left(N_{d_j}, W_{N+1-j}\right) \tag{4.41}$$

unde $m_{cr(j+1)}$ este marca asociată unei stări continue iar $w\left(Tcr_{(r)} \times Pcr_{(r+1)}\right)$ este ponderea arcului de la o traziţie continuă la o stare continuă a WMR.

Pentru $N = 5$ arcele $\left(P_i XT_j\right)$ au o pondere egală unu, unde

$$P_i = \{ROBOT\ state1,\ ROBOT\ state2\} \in^o \{Tcr_k\}_{k=\overline{1,10}} \cap P^D \tag{4.42}$$

Observaţia 3: Pentru o PN sincronizată, o tranziţie este activată când fiecare din intrările sale conţine suficiente jetoane. Dacă tranziţia este activată, atunci ea poate fi aprinsă când apare un eveniment asociat.

$Sync$, funcţia din (4.4), devine în forma particularizată pentru FMML, A/DML HERA&HORSTMANN deservită dxe WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF:

$$Sync : \{Tdd_j\}_{j=\{1,3,4,6,7,9,12,13\}} \to \{Edd^1, Edd^2\} \tag{4.43}$$

$$Sync : \{Tdd_i\}_{i=\{1,4,7,13\}} \to \{Edd_i^1\}_{i=\{1,4,7,13\}} \tag{4.44}$$

$$Sync : \{Tdd_i\}_{i=\{3,6,9,12\}} \to \{Edd_i^2\}_{i=\{3,6,9,12\}} \tag{4.45}$$

$$Sync : \{Tdd_j\}_{j=\overline{1,16}} \cup \{Tdr_k\}_{k=\overline{1,21}} \cup \{Tcr_k\}_{k=\overline{1,10}} \to e \tag{4.46}$$

În Fig.4.7 este prezentat modelul complet, SHPN, pentru FMML, A/DML HERA&HORSTMANN, deservită de un WMR, Pioneer 3-DX, echipat cu RM, Pioneer 5-DOF.

4.2. Simularea modelului SHPN

Modelul SHPN a fost testat, analizat şi verificat prin intermediul pachetului Sirphyco, [97]. Utilitatea modelului SHPN constă în a găsi viteza maximă a WMR care produce timpul minim al ciclului general asociat procesului de dezasamblare. Această viteză trebuie să corespundă caracteristicilor şi limitărilor constructive ale WMR. În fig.4.8 este prezentat răspunsul simulat al starilor continue şi discrete ale WMR echipat cu RM pentru modelul HPH din Fig.4.5. Mărcile stărilor continue ale WMR, înainte şi după simulare urmăresc distanţele din Fig.3. 8. Evoluţia stărilor continue şi discrete ale WMR, pentru prima operație de dezasamblare, $j = 1$, poate fi descrisă astfel: $Pcr(r)$– este variația în timp a distanței parcurse de WMR între locația unde se produce prima dezasamblare și magazia numarul 5 (1031mm așa cum este prezentat Fig.3.8); $Pcr(r+1)$, $Pcr(r+2)$, $Pcr(r+3)$ –reprezintă variația distanței care trebuie parcursă de robot în următorul ciclu elementar (730mm, Fig.3.8) corelat cu evenimentele de sincronizare Edd_1^1 și Edd_3^2. $Pdr(s), Pdr(s+1), Pdr(s+2)$, $Pdr(s+6)$, $Pdr(s+7)$ – reprezintă acțiunile discrete al WMR (apucare și eliberare piesă, închidere gripper, repoziționare la următoarea stație de dezasamblare) corelate cu deplasarea pe banda transportoare între stațiile de lucru a piesei care se dezasamblează.

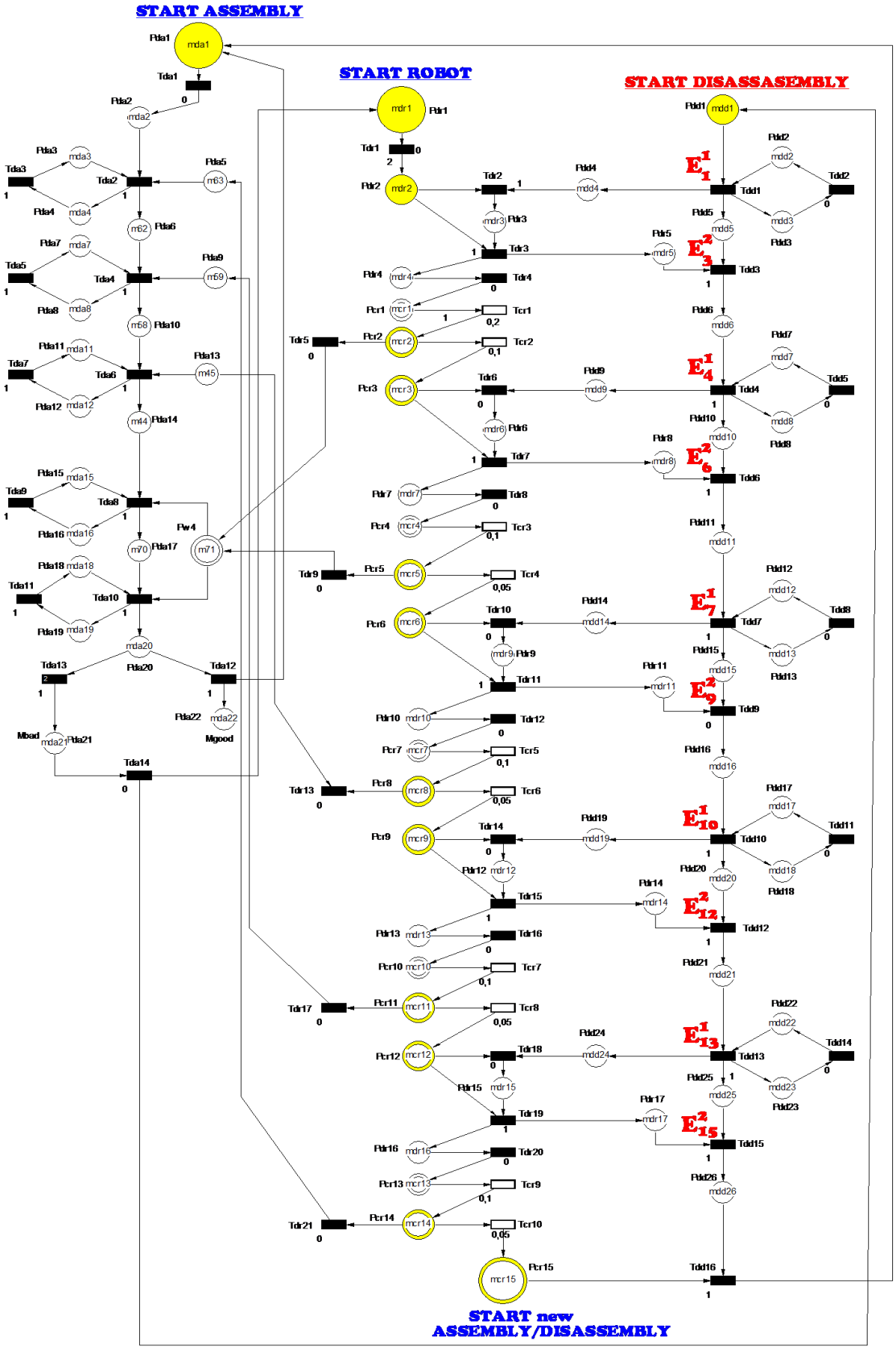


Fig.4.7. Modelul complet SHPN pentru FMML, A/DML HERA&HORSTMANN, deservită de un WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm

Pentru ultimul ciclu elementar de dezasamblare, răspunsul simulat este prezentat în Fig.4.9. Valoarea maximă a marcajelor $Pcr(r+3)|_{j=5} = Pcr(0)|_{j=5}$ este egală cu distanța parcursă de WMR în timpul primului stagiu al dezasamblării (deplasarea robotului de la dreapta la stânga), $Pcr(r) = 1031$, și corespunde inițializării unui nou ciclu de dezasamblare. Similar, reprezentarea grafică a stărilor discrete $Pdr(s)$, $Pdr(s+1)$, $Pdr(s+2)$, $Pdr(s+6)$ și $Pdr(s+7)$ nu cuantifică temporizările induse de recepția evenimentului Edd^2 (”STARTING” următorul stagiu de dezasamblare) astfel că, dezasamblarea este terminată.

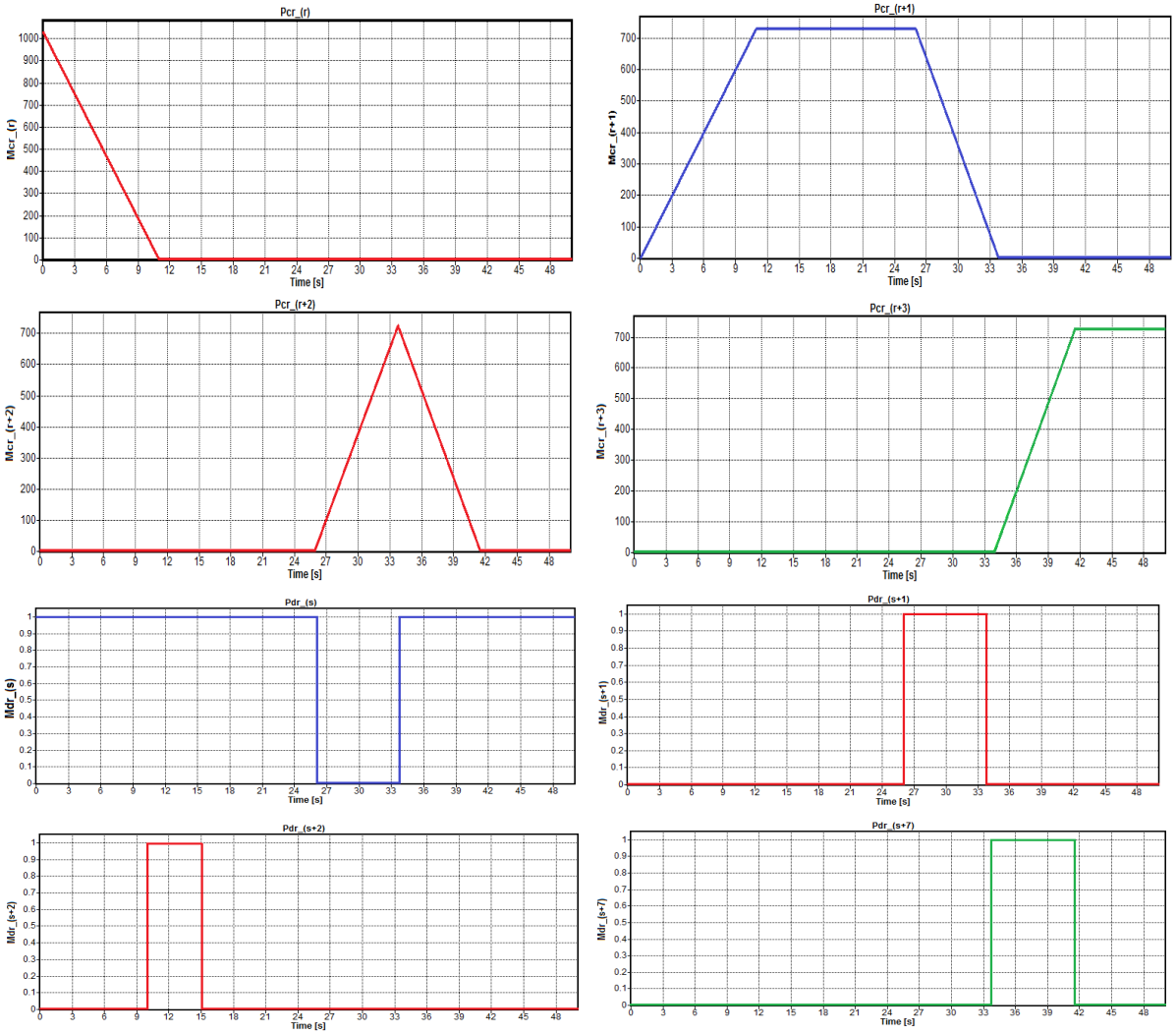
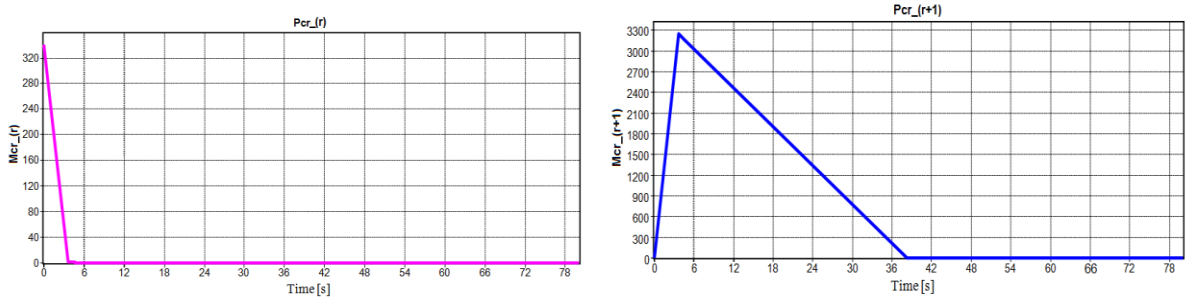


Fig.4.8. Variaţia stărilor continue şi discrete asociate primului ciclu elementar de dezasamblare



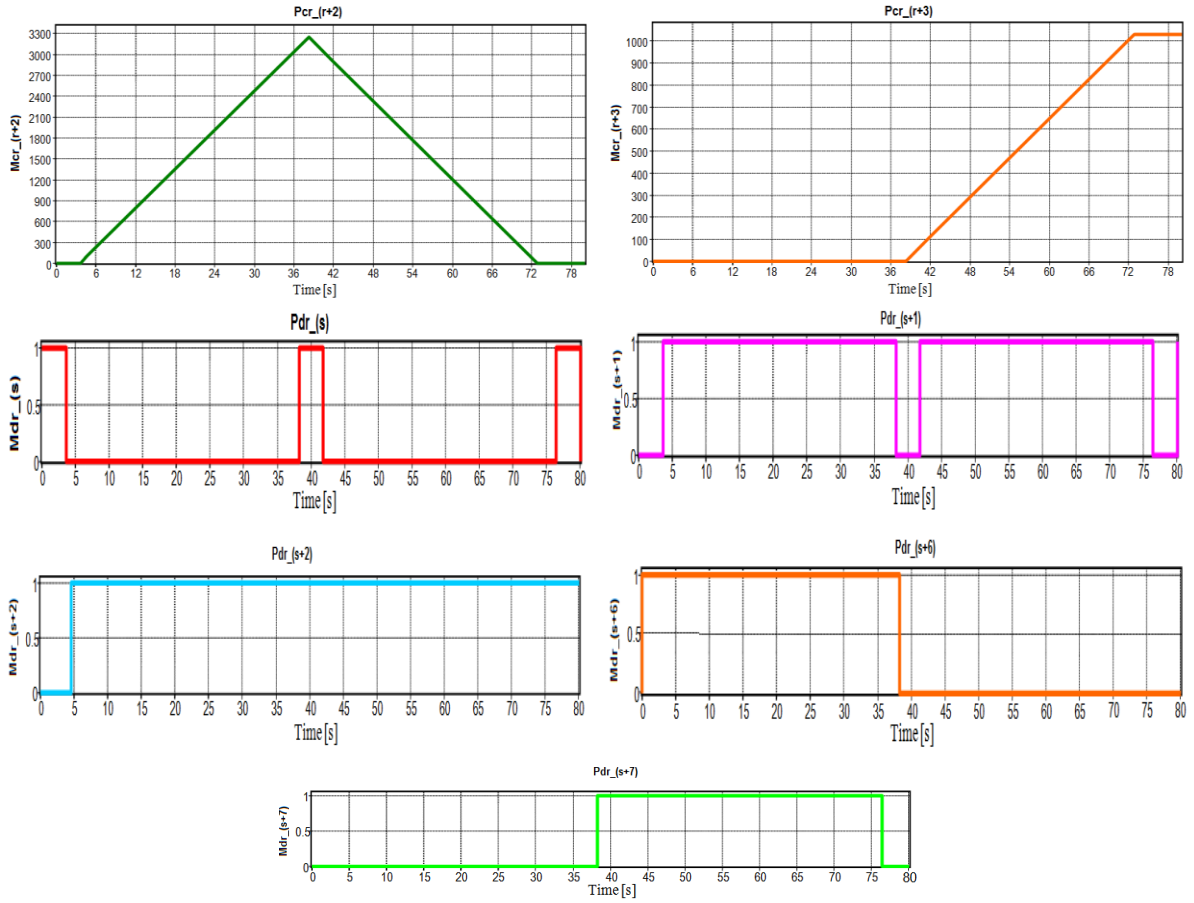


Fig.4.9. Variaţia stărilor continue şi discrete asociate ultimului ciclu elementar de dezasamblare

4.3. Două sisteme robotice mobile integrate în A/DML

4.3.1. Structura modelului

A/DML este deservită de doi WMRs lucrând în paralel, unul este echipat cu RM, utilizat pentru manipulare, iar al doilea, fară RM, este utilizat pentru transport. Cele două sisteme robotice, lucrează în paralel, sincronizat. WMR (WMR1) echipat cu manipulator (RM1) preia componenta de la locația unde se produce dezasamblarea și o depune pe WMR transportor (WMR2). Apoi WMR1 și WMR2 se deplasează simultan la locația de depozitare, unde RM1 preia componenta de pe platforma WMR2 și o eliberează în magazie. Și în acest caz, apectul hibrid al modelului este determinat de variabilele asociate distanţelor parcurse de WMR1 și WMR2, distanțe considerate între locaţiile unde se produc operaţiile de dezasamblare şi locaţiile magaziilor de depozitare. Variaţia acestor variabile este în concordanţă cu viteza de deplasare a celor două platforme, între locaţiile A/DML. Pentru a dezvolta un model A/DML deservită de WMR1 echipat cu RM1 și WMR2 ,se va considera aspectul hibrid al proceselor de A/D. Pentru modelare și aici, se va apela la instrumentul THPN, [1], [15], instrument care integrează aspectul discret al proceselor de A/D cu aspectul continuu al mişcării WMR şi manipulării componentelor de catre RM1. Modelul global este de tip SHPN deoarece este interfaţat cu evenimente externe pentru sincronizare, evenimente care nu sunt altceva decât semnale provenite de la senzori. Aceste evenimente sunt utile atât la modelare/simulare cât şi la conducerea în timp real. Structura SHPN, din Fig.4.10, şi reprezentarea pe blocuri, din Fig.4.11, corespunde modelării discrete a proceselor de A/D şi

a dinamicii continue a celor doi roboți, WMR1 echipat cu RM1 și WMR2, care deservesc A/DML în procesul de dezasamblare. Structura internă a modelului SHPN integrează trei modele PN, fiecare dintre ele având o tipologie specifică: TPN (Rețea Petri Temporizată), SPN (Rețea Petri Sincronizată) şi THPN (Rețea Petri Hibridă Temporizată). Aceste modele descriu urmatoarele operații care se execută automat: asamblare/depozitare în magazii, de tipologie TPN; dezasamblare produs asamblat care nu a trecut testul de calitate, de tipologie SPN și TPN; Integrarea WMR1 echipat cu RM1 și WMR2 în A/DML, în procesul de dezasamblare, de tipologie THPN. În Fig.4.11 se poate observa un bloc de tipul e-TPN care corespunde unei singure operații de asamblare. Modelul TPN asociat acestui bloc este prezentat în Fig.4.3. Acest model elementar se repetă pentru fiecare operație de asamblare. În Fig.4.4 este prezentat modelul TPN generalizat corespunzător întregului proces de asamblare, proces care include secvența repetitivă, asamblarea celor doi cilindri si testul de material, care s-a convenit a fi asimilat testului de calitate. Secvența repetitivă care se evidențiază în decursul procesului de dezasamblare, conține operația de dezasamblare şi asistența din partea WMR1 echipat cu RM1 și WMR2. Secvența repetitivă din cadrul procesului de dezasamblare poate fi modelată ca o secvență elementară SHPN, e-SHPN, și este marcată in Fig.4.11. Se poate observa pe figură ca modelul e-SHPN, conține două submodele e-THPN, e-THPN1 asociat WMR1 echipat cu RM1și e-THPN2 asociat lui WMR2. Liniei de A/D, în procesul de dezasamblare, i se asociază un submodel e-SPN+TPN. În Fig 4.13, este redat modelul e-SHPN corespunzător primei operaţii de dezasamblare, model care va fi testat prin simulare. Testarea prin simulare a modelului SHPN, fară semnalele de sincronizare, adică, simularea în mod autonom a modelului HPN este utilă în a conecta dinamica discretă a A/DML cu dinamica continuă a WMR1 echipat cu RM1 și dinamica WMR2.

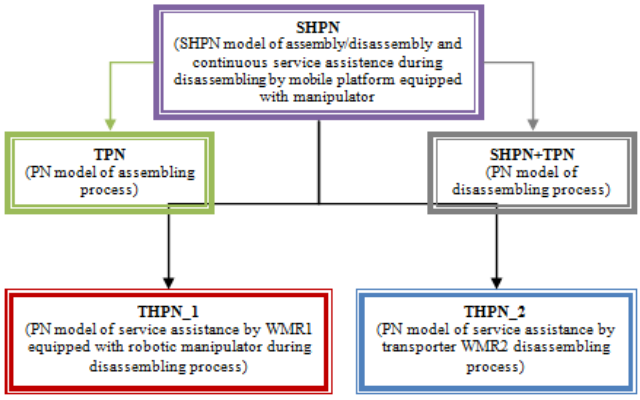


Fig.4.10. Structura modelului SHPN pentru doi WMRs integrați în A/DML

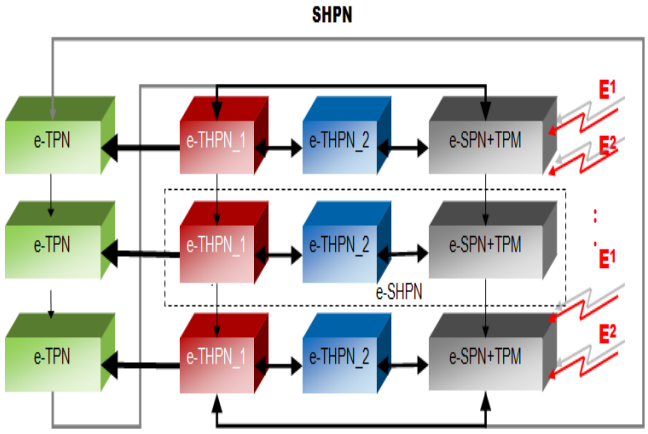


Fig.4.11. Modelul SHPN cu doi WMRs operând în paralel, sincron, integrați în A/DML, cu blocurile repetitive elementare, e-SHPN

4.3.2. Simularea modelului SHPN

În Fig.4.13 este prezentat modelul SHPN corespunzător primului stagiu de dezasamblare, pentru doi WMRs, lucrând în paralel, integrați în A/DML: WMR1, Pioneer 3-DX, echipat cu RM1, Pioneer 5-DOF Arm, utilizat pentru manipulare componentă și WMR2, PatrolBot, utilizat pentru transport. Sunt necesare 4 semnale de sincronizare pentru acțiunile de apucare componentă de la locația de dezasamblare și eliberare la magazia de depozitare aferentă:
Edd1-STOP A/DML și START dezasamblare;
Edd2-RM1 apucare componentă și START A/DML;
Edd3-RM1 de pe WMR1 în poziție pentru deplasare după ce a preluat componenta de la prima locație de dezasamblare și a eliberat-o pe platforma WMR2;
Edd4-RM1 de pe WMR1 în poziție pentru deplasare după ce a preluat componenta de pe platforma robotică transportoare WMR2 pentru a fi eliberată la magazia W1.
De asemenea, s-au utilizat alte trei variabile:
$P16$-RM1 de pe WMR1 în poziție de deplasare după ce a fost eliberată pe platforma WMR2 componenta rezultată din dezasamblare;
$P19$-RM1 în poziție de deplasare după ce a fost eliberată componenta în magazia W1;
$P23$-RM1 apucare componentă eliberată din dezasamblare.
Intervalele de timp ale acțiunilor WMRs și stările piesei care se dezasamblează în timpul unui ciclu elementar sunt prezentate în Fig.4.12. Evoluția stărilor continue și discrete, în urma simulării în Sirphyco, sunt prezentate în Fig.4.14. Modelul SHPN este util pentru a determina valoarea maximă a platformelor mobile astfel încât să se obțină un timp de ciclu minim al procesului de dezasamblare. Această viteză trebuie să fie setată de acord cu limitările fizice ale celor două platforme. Marcajele stărilor continue ale WMRs, înainte și după simulare, se verifică cu distanțele prezentate în Fig.3.11. Evoluția marcajelor stărilor discrete și continue ale WMRs, corespunzătoare $j=1$ este următoarea: $Pcr(8)$ și $Pcr(13)$ sunt variația temporală a distanţei care va trebui parcursă de WMRs de la prima locație de dezasamblare la ultima magazie de depozitare, $Pcr(26)$– variația temporală a distanței ce trebuie parcursă de WMRs în următorul stagiu de dezasamblare (730mm cum este prezentat în Fig.3.10) în corelație cu evenimentele de sincronizare Edd_1^1 and Edd_3^2. Stările discrete: $Pdr(11)$, $Pdr(17)$, $Pdr(12)$, și $Pdr(18)$, –reprezintă temporizările asociate la acțiunile discrete ale WMRs (prindere sau eliberare componentă, închidere gripper, repoziționare RM1 la următoarea stație de lucru) corelate cu distanța parcursă pe benzile trasportoare, de către piesa care se dezasamblează, în interiorul unei stații de lucru sau între stațiile de lucru.

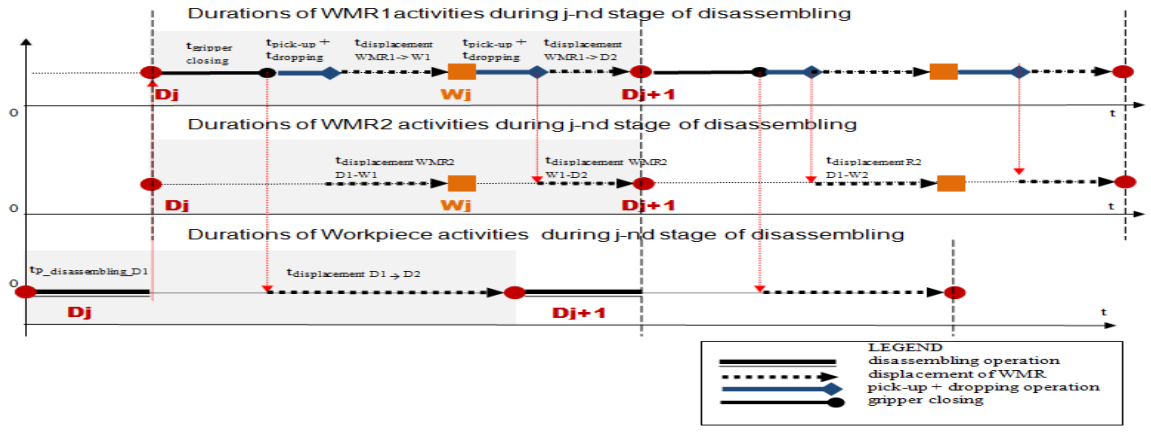


Fig.4.12. Intervalele de timp corespunzătoare ciclului elementar de dezasamblare ”j” pentru doi WMRs integrați în A/DML

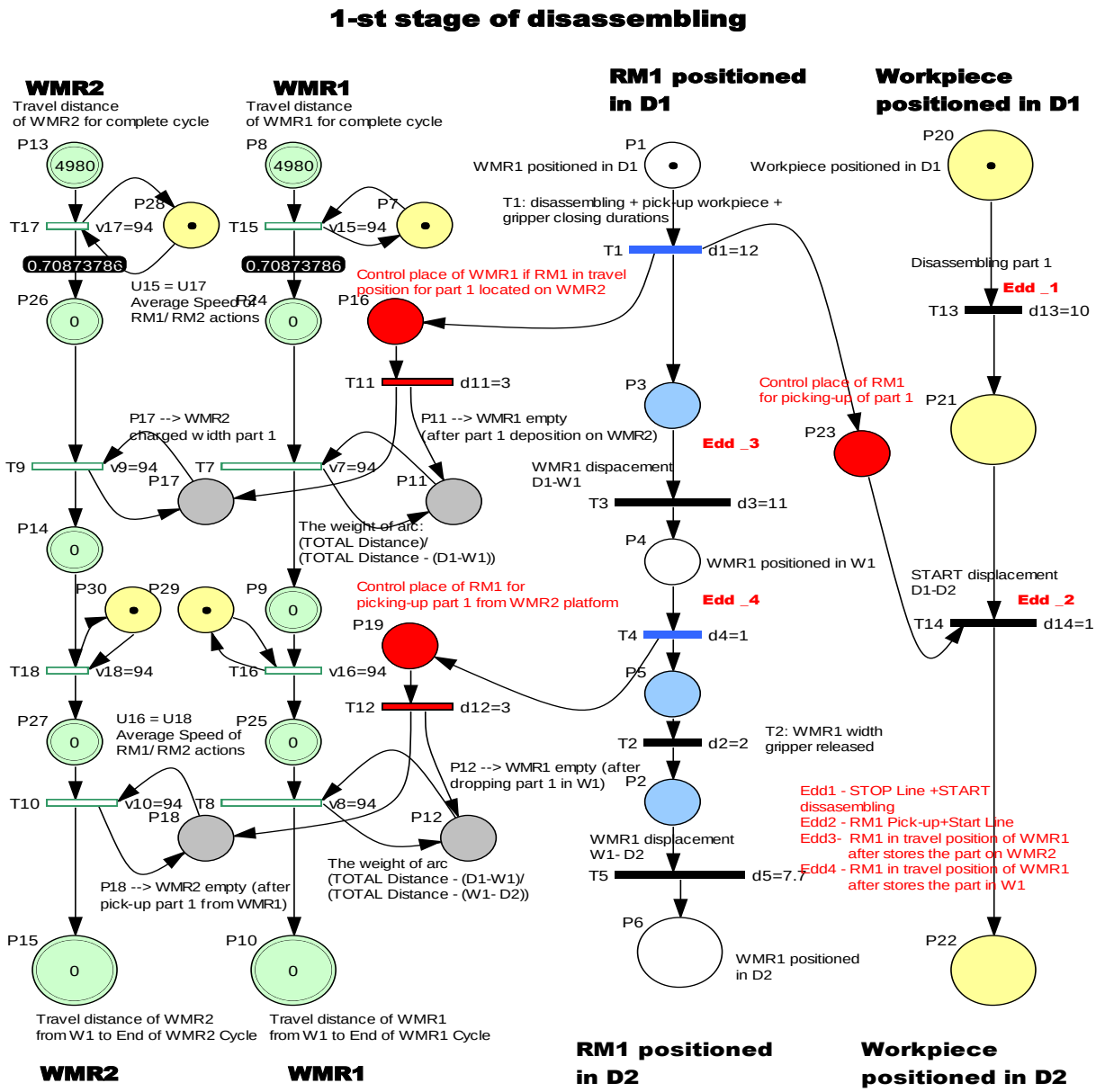
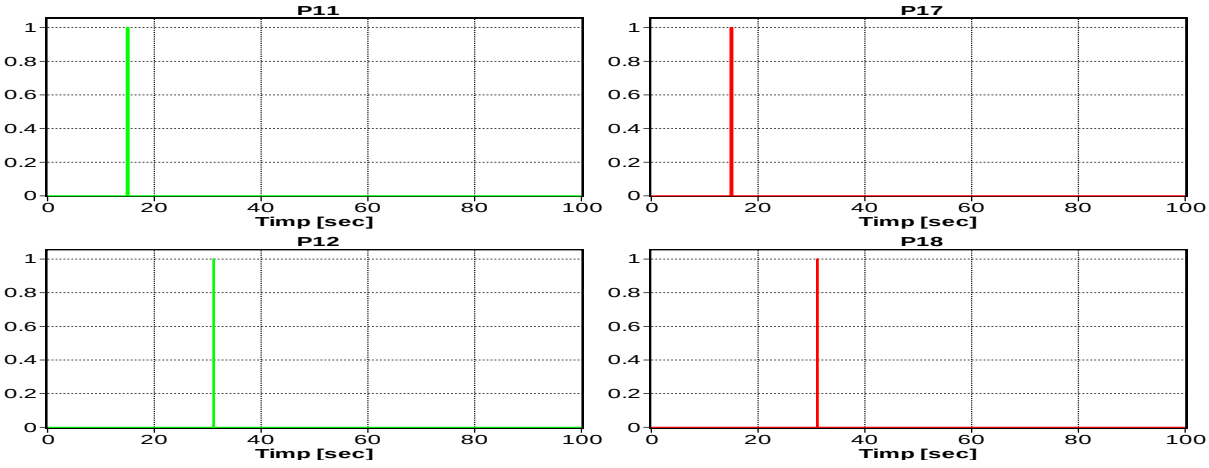


Fig.4.13. Modelul SHPN corespunzător primului ciclu elementar de dezasamblare, pentru doi WMRs integrați în A/DML



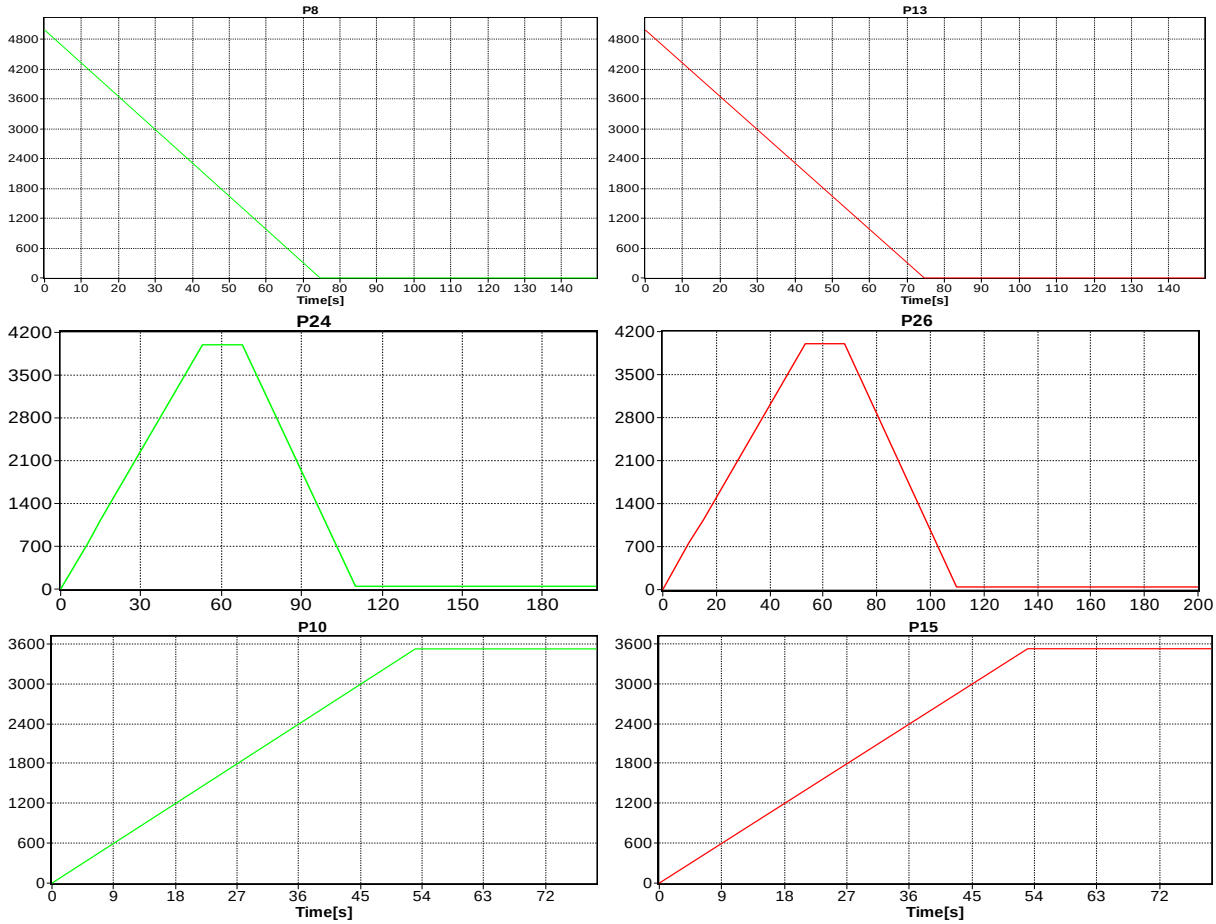


Fig.4.14. Variația stărilor discrete și continue corespunzătoare primului ciclu elementar de dezasamblare pentru doi WMRs integrați în A/DML

4.4. Modelul SHPN pentru FMML, P/RML FESTO MPS-200 deservită de un WMR echipat cu RM

4.4.1. Structura modelului

Aspectul hibrid al modelului este determinat de variabilele asociate distanţelor parcurse de WMR echipat cu RM. Aceste distanţe sunt parcurse de WMR între ultima staţie de lucru (storage workstation) şi prima staţie (handling workstation) a P/RML. Pentru modelare se va apela la instrumentul SHPN [1], [15], care integrează aspectul discret al proceselor de P/R cu aspectul continuu al deplasării WMR-ului şi manipulării componentelor de către RM. Modelul global este de tip SHPN deoarece este interfaţat cu evenimente externe pentru sincronizare, evenimentele fiind semnale provenite de la cele două sisteme servoing vizuale. Aceste evenimente sunt utile atât la modelare/simulare, cât şi la conducerea în timp real. Structura SHPN, din Fig.4.15, corespunde modelării discrete a proceselor de P/R şi a dinamicii continue a WMR-ului echipat cu RM, care deserveşte P/RML pentru a aduce piesa de la ultima la prima staţie de lucru pentru a fi reprelucrată. Structura internă a modelului SHPN integrează două modele PN, fiecare dintre ele având o tipologie specifică: TPN (Rețea Petri Temporizată) şi SPN+TPN (Rețea Petri sincronizată+Rețea Petri Temporizată) si THPN (Rețea Petri Hibridă Temporizată). Aceste modele descriu următoarele operații care se execută automat: procesare (modelare cu TPN), reprocesare (modelare cu TPN) şi asistenţă din partea WMR echipat cu RM pentru prindere piesă şi aducerea ei în vederea reprocesării.

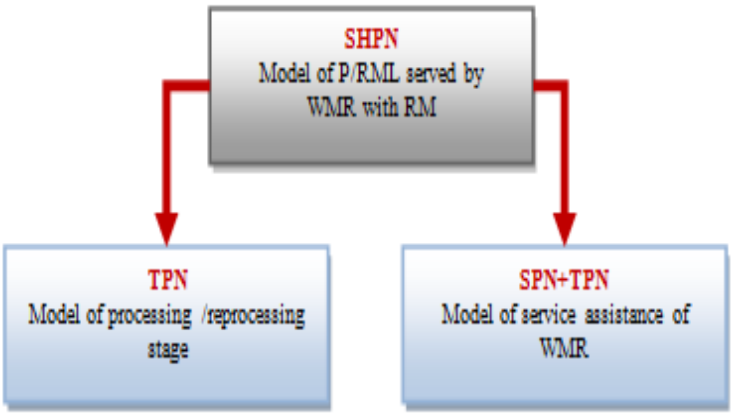


Fig.4.15. Structura modelului SHPN

4.4.2. Formalismul SHPN asociat WMR echipat cu RM integrat în P/RML

Modelul final este de tip SHPN deoarece este interfațat cu evenimente externe de sincronizare, într-o abordare a modelării/simulării în timp real. În cazul abordării hibride cu rețele Petri temporizate, THPN este un septuplu:

$$THPN = \langle P, T, Pre, Post, m_0, h, tempo \rangle \tag{4.47}$$

$$P = P_d \cup P_c \tag{4.48}$$

este o mulțime finită de stări, diferită de mulţimea vidă, unde P_d este mulţimea stărilor discrete

$$P_d = \{P_{dp_i}\}_{i=\overline{1,13}} \cup \{P_{dr_j}\}_{j=\overline{1,9}} \tag{4.49}$$

iar P_c este mulţimea stărilor continue

$$P_c = \{P_{cr_k}\}_{k=\overline{1,3}} \tag{4.50}$$

$$T = T_d \cup T_c \tag{4.51}$$

este o mulțime de tranziții, finită, diferită de mulţimea vidă, unde T_d reprezintă mulțimea tranzițiilor discrete

$$T_d = \{T_{dp_i}\}_{i=\overline{1,12}} \cup \{T_{dr_j}\}_{j=\overline{1,11}} \tag{4.52}$$

iar T_c mulțimea tranzițiilor continue

$$T_c = \{T_{cr_k}\}_{k=\overline{1,3}} \tag{4.53}$$

Observația 4: Mulțimile P și T sunt disjuncte, $P \cap T = \varnothing$.

$$Pre : P \times T \to Q_+ \tag{4.54}$$

este funcția de incidență la intrare

$$Post : P \times T \to Q_+ \tag{4.55}$$

este funcția de incidență la ieșire.

Observația 5: În definirea funcțiilor Pre, $Post$ și m_0, N corespunde cazului în care $P_i \in P_d$, și Q_+ sau R_+ corespund cazului în care $P_i \in P_c$.

$$m_0 : P \to R_+ \tag{4.56}$$

reprezintă marca inițială, iar

$$h: P \cup T \to \{D, C\} \tag{4.57}$$

reprezintă funcția hibrid și indică pentru fiecare nod, tipul acestuia, discret (mulțimile P_d și T_d) sau continuu (mulțimile P_c și T_c),

$$h: P_d \cup T_d \to \{D\}; h: P_c \cup T_c \to \{C\} \tag{4.58}$$

tempo este o funcție definită pe mulțimea tranzițiilor, T, la mulțimea numerelor raționale

$$tempo: T \to Q_+ \cup \{0\} \tag{4.59}$$

Dacă $T_j \in T_d$, atunci

$$d_j = tempo(T_j) \tag{4.60}$$

este temporizarea asociată lui T_j. Dacă $T_{cr} \in T_c$ atunci

$$U_r = \frac{1}{tempo(T_{cr})} \tag{4.61}$$

reprezintă rata de flux asociată lui T_{cr}.

Pentru

$$T_c = \{T_{cr_k}\}_{k=\overline{1,3}}, U_{cr_k} = U_r; U_{r\max} = V_r \tag{4.62}$$

unde U_{cr} fluxul variabil a robotului mobil între două locații continue. Se consideră viteza robotului $V_r = 94 mm/s$.

4.4.3. Simularea modelului

Modelul SHPN din Fig.4.16 a fost testat, analizat şi verificat prin intermediul pachetului Sirphyco. Utilitatea modelului SHPN constă în a găsi viteza maximă a WMR care produce timpul minim al ciclului general asociat P/R. Această viteză trebuie să corespundă caracteristicilor şi limitărilor constructive ale WMR. În fig.4.17 este prezentat răspunsul simulat al stărilor continue și discrete ale WMR echipat cu RM pentru modelul HPH din Fig.4.5. Mărcile stărilor continue ale WMR, înainte și după simulare urmăresc distanțele din Fig.3.15. În Fig.4.16 sunt prezentate duratele tranziţiilor discrete prin care P/RML execută următoarele operaţii: manipulare, sortare, alezare, găurire, transport şi depozitare. În timpul procesării, tranzițiile care corespund stărilor procesului care se produc instantaneu au valoarea zero : $T_{dp2} = 0$, $T_{dp3} = 0, T_{dp9} = 0$ și $T_{dp10} = 0$. Tranzițiile pentru care s-a evidențiat o durată de execuție, au următoarele valori: $T_{dp1} = 3.5s$; $T_{dp4} = 6.9s$; $T_{dp5} = 4.8s$; $T_{dp6} = 3.9s$; $T_{dp7} = 3.4s$; $T_{dp8} = 10.9s$; $T_{dp11} = 11.8s$; $T_{dp12} = 13.3s$. După ce se realizează inițializarea WMR-ului, se aşteaptă ca o piesă care nu a trecut testul de calitate să intre în acţiunea de reprocesare sau rebutare, acţiune care se derulează la prima staţie de lucru după ce piesa este adusă și eliberată de WMR echipat cu RM. Prin utlizarea sistemului servoing vizual este detectată o astfel de piesă pe staţia de depozitare, detectare care produce un eveniment extern care are drept efect declaşarea acţiunii platformei robotice. S-au considerat necesare două semnale de sincronizare între P/RML şi WMR echipat cu RM:

Edd_1: semnal de sincronizare a WMR cu START PROCESSING/REPROCESSING;

Edd_2: semnal de sincronizare a WMR cu P/RML pentru start REPROCESSING.

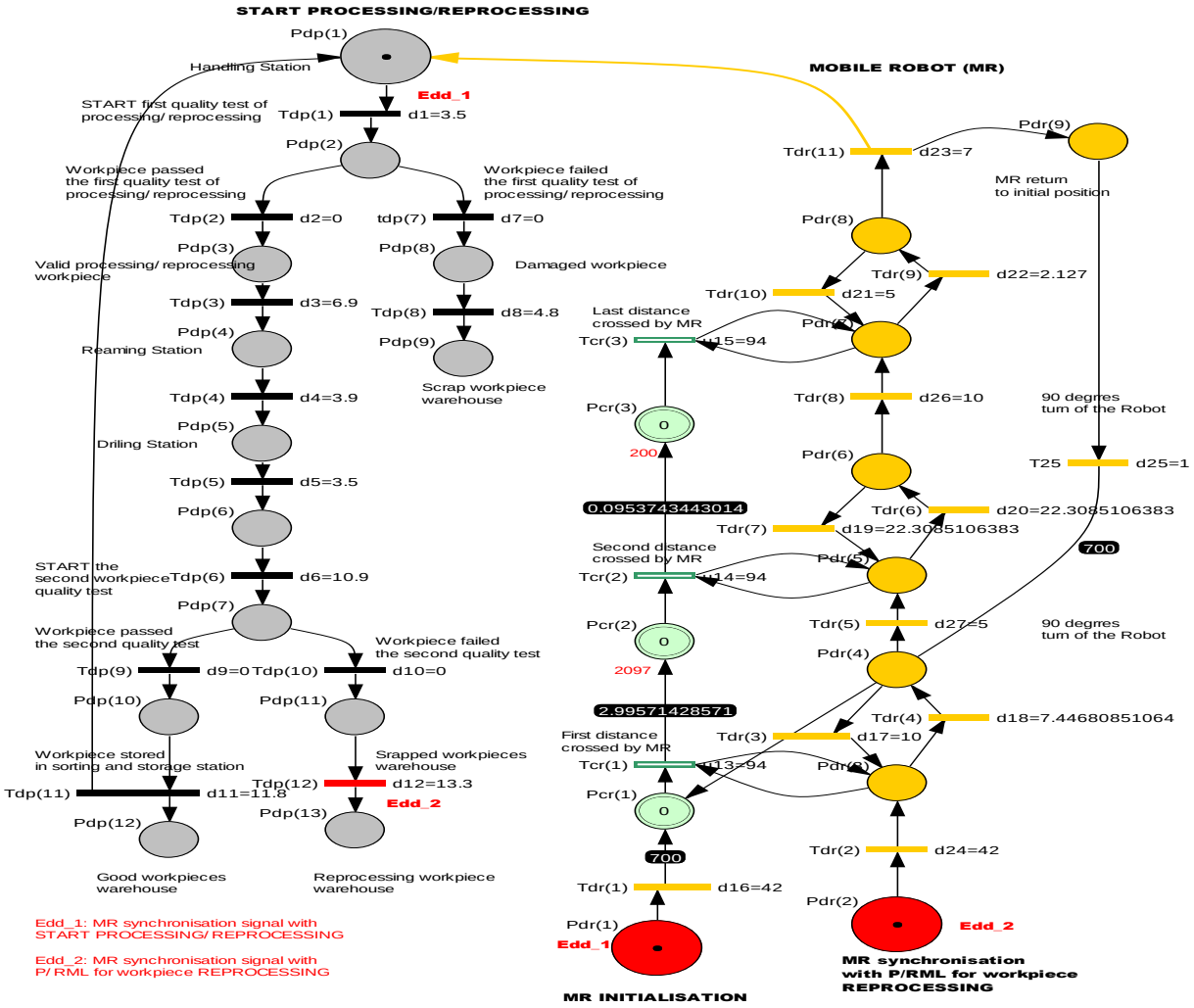
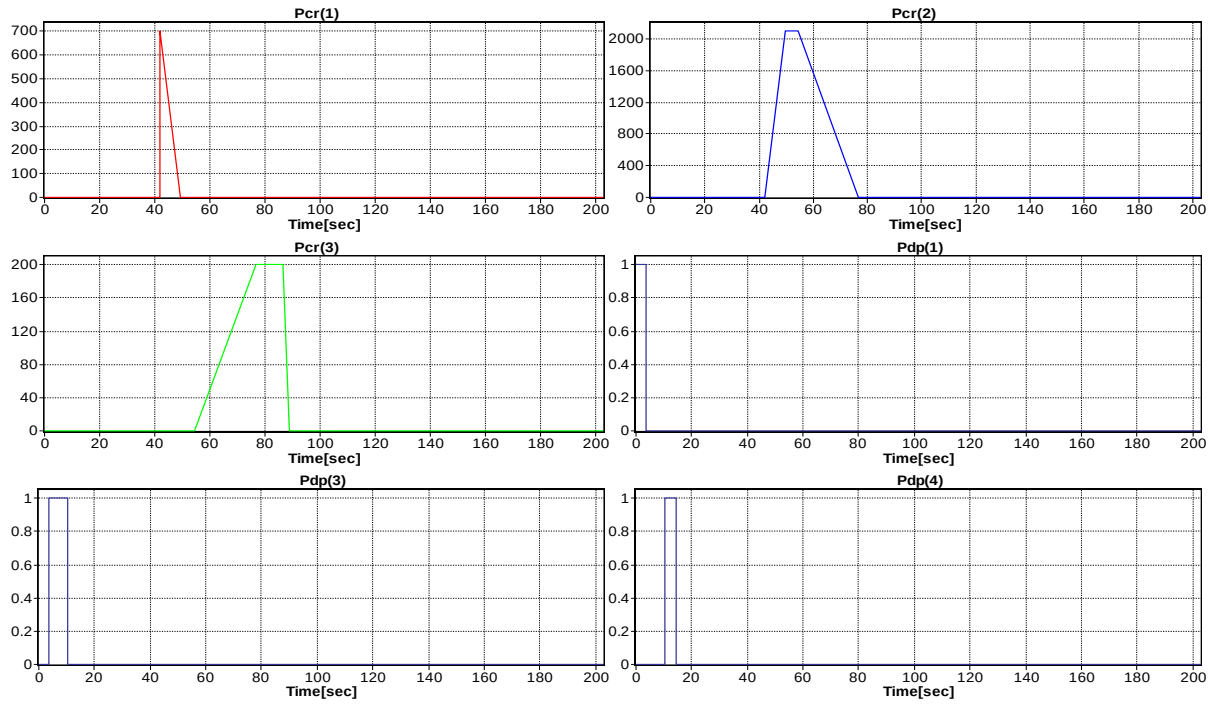


Fig.4.16. Modelul SHPN al WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm integrat în P/RML FESTO MPS-200



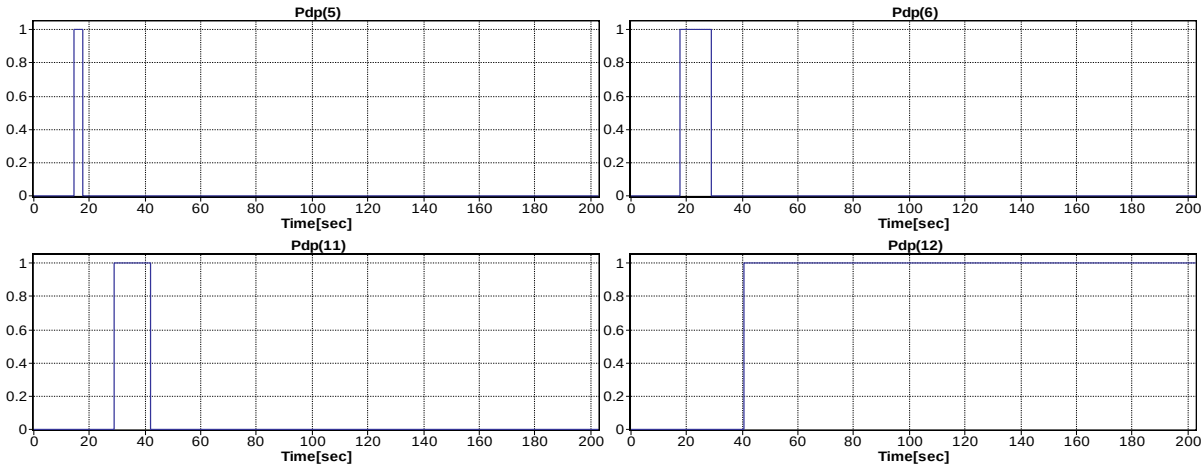


Fig.4.17. Variația stărilor continue și discrete ale WMR echipat cu RM integrat în P/RML

4.5. Concluzii

În acest capitol se pot revendica drept contribuţii următoarele:

- Structura, modelul SHPN şi formalismul general, cu evidenţierea submodelelor repetitive, pentru A/DML deservită de un WMR echipat cu RM;
- Structura, modelul SHPN şi formalismul particularizat, cu evidenţierea submodelelor repetitive, pentru A/DML HERA&HORSTMANN deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm;
- Simularea în pachetul Sirphyco a modelului SHPN pentru A/DML HERA&HORSTMANN deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, primul şi ultimul ciclu elementar de dezasamblare;
- Structura, modelul SHPN şi formalismul particularizat, cu evidenţierea submodelelor repetitive, pentru A/DML HERA&HORSTMANN deservită de doi WMRs, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm şi PatrolBot;
- Simularea în pachetul Sirphyco a modelului SHPN pentru A/DML HERA&HORSTMANN deservită de doi WMRs, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm şi PatrolBot, primul ciclu elementar de dezasamblare;
- Structura, modelul SHPN şi formalismul particularizat pentru P/RML, FESTO MPS-200 deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm;
- Simularea în pachetul Sirphyco a modelului SHPN pentru P/RML, FESTO MPS-200 deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm.

Capitolul 5

Acţionarea şi conducerea FMMLs cu roboţi integraţi, în timp real

Utilizând, după caz, platforme LabView, Visual C++ și MATLAB, în acest capitol se prezintă acţionarea şi conducerea în timp real a roboților mobili integrați în A/DML și P/RML, conducere bazată pe modelele SHPN. Sunt tratate următoarele probleme pentru conducerea FMMLs:

- Testarea și validarea în laborator a tehnologiei hibride de conducere, în timp real, a fabricației flexibile pe A/DML, HERA&HORSTMANN, deservită de un WMR, Pioneer 3-DX, echipat cu un RM, Pioneer 6-DOF Arm;
- Testarea și validarea în laborator a tehnologiei hibride de conducere, în timp real, a fabricației flexibile pe A/DML, HERA&HORSTMANN, deservită de doi roboți mobili colaborativi, lucrând în paralel, sincron, unul dintre ei, Pioneer 3-DX, echipat cu RM, Pioneer 6-DOF Arm, utilizat pentru manipulare, și un al doilea, PatrolBot, utilizat pentru transpot;
- Testarea și validarea în laborator a tehnologiei hibride de conducere, în timp real, a fabricației flexibile pe P/RML, FESTO-MPS 200, deservită de un WMR echipat cu RM.

5.1. A/DML, HERA&HORSTMANN, deservită de un WMR echipat cu un RM

5.1.1. Structura hardware de acţionare şi conducere

Acţionarea şi conducerea tehnologiei hibride de fabricaţie flexibilă pe A/DML, HERA&HORSTMANN, deservită de un WMR, Pioneer 3-DX, echipat cu un RM, Pioneer 6-DOF Arm, utilizează pachetele software, Vizual C++, LabVIEW 2010 şi Simatic STEP 7, [73], [85], [87]. Această aplicație permite acţionarea, sincronizarea şi conducerea în timp real a tehnologiei hibride, tehnologie bazată pe modelul hybrid SHPN, a proceselor de A/D deservite de un WMR echipat cu RM [78], [79], [80]. Structura de acționare şi conducere este prezentată în Fig.5.1. Aplicaţia permite conducerea complet automatizată a thenologiei de fabricaţie flexibilă, pentru un lot de producţie dat.

Comunicarea dintre PLC-ul liniei flexibile de mecatronică şi staţia de lucru care asigură sincronizarea cu platforma robotică se face prin intermediul unei plăci de achiziţie DAQ NI USB6008. Deoarece ieşirile şi intrările digitale ale automatului programabil Siemens CPU 314C-2 DP funcţionează cu tensiune de 0-24 V iar placa de achiziţie lucrează cu tensiuni cuprinse între intervalul de 0-5 V s-a folosit o placă cu relee pentru a evita avarierea plăcii de achiziţie, precum este prezentat în Fig.2.15 împreună cu legăturile aferente între intrările/ieşirile digitale ale PLC-ului şi placa de achiziţie. [81], [82].

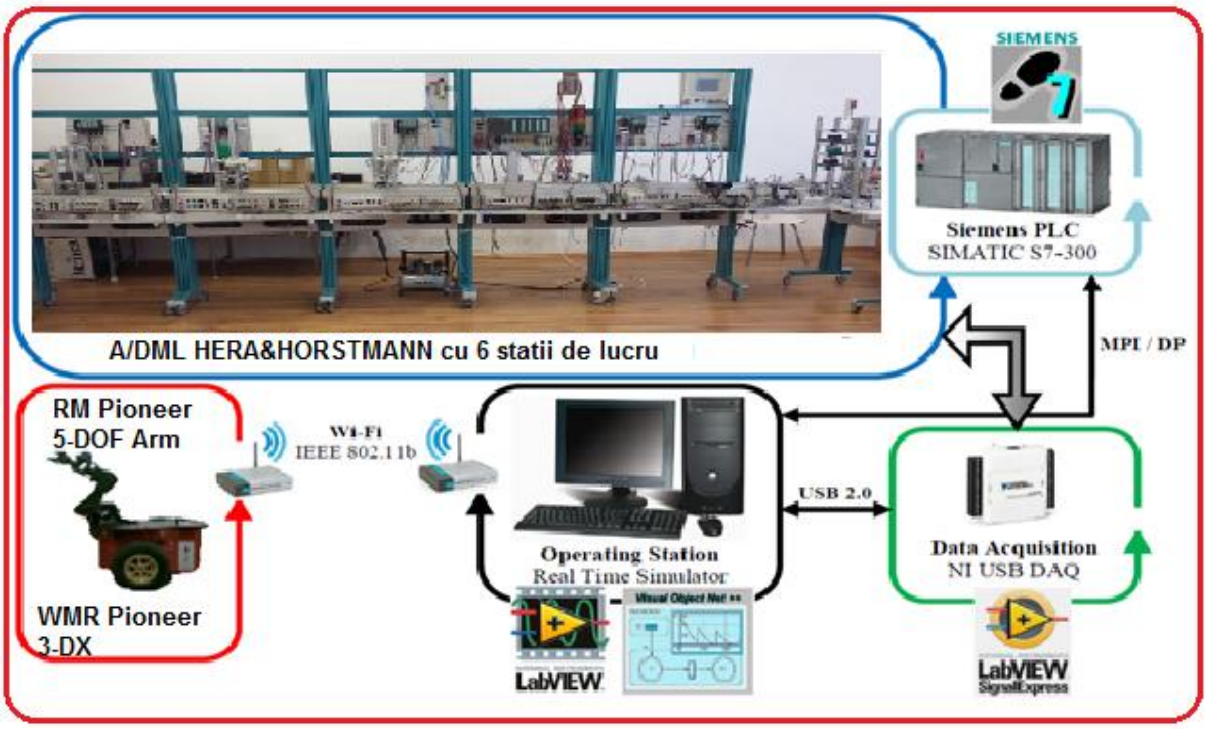


Fig.5.1. Structura de conducere A/DML HERA&HORSTMANN deservită de WMR, Pioneer 3-DX, echipat cu RM, Pioneer 5-DOF Arm

5.1.2. Structura software

Structura aplicaţiei de conducere este compusă din două bucle locale de conducere, Fig.5.2, fiecare dintre ele fiind dedicată conducerii celor două entităţi, A/DML, respectiv, WMR echipat cu RM:

- bucla de conducere a FMML, A/DML HERA&HORSTMANN este implementată în PLC-ul SIEMENS S7-300 cu procesor 314-2 DP și programată în mediul de programare Simatic STEP 7;

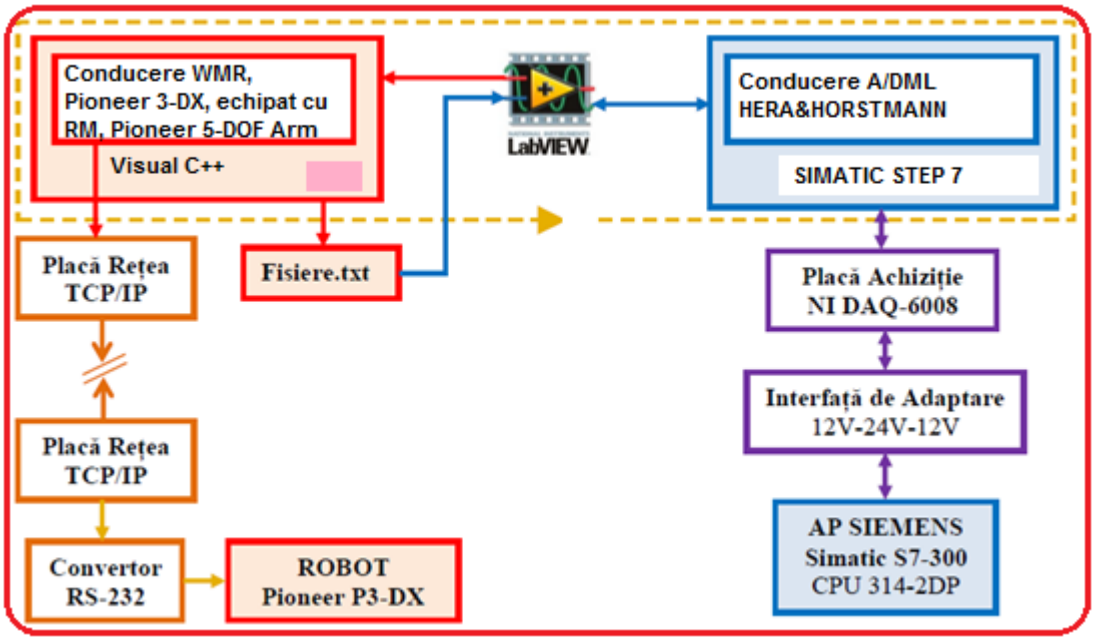


Fig.5.2. Structura software de conducere în LabView

- bucla de conducere WMR, Pioneer P3-DX echipat cu RM, Pioneer 3-DX, este implementată într-un fişier de tip executabil, Visual Studio C++, care este lansat în aplicație de programul LabVIEW. În urma execuției fişierului executabil sunt scrise în fișiere de tip .txt variabile care reprezintă poziția robotului mobil echipat cu manipulator din cadrul procesului de dezasamblare. Programul LabVIEW citește variabilele din fișierele .txt, actualizează interfața grafică cu noua poziție a WMR și intervine în algoritmul de sincronizare a operațiilor de dezasamblare deservită de robotul mobil echipat cu manipulator.

Comunicația dintre aceste două bucle locale de conducere se realizează printr-o interfațare dintre linia flexibilă și un calculator de proces, printr-o placă de achiziție, NI DAQ-6008, Fig.5.3, care culege și transmite date din procesul de asamblare și dezasamblare și o comunicație wireless pe un protocol TCP/IP dintre robotul mobil și calculatorul de proces. Pe acest calculator de proces se găsește programul de conducere implementat în mediul LabVIEW care are rolul de a sincroniza în timp real cele două bucle de conducere în cadrul procesului de dezasamblare.

5.1.3. Interfaţa grafică utilizator

Interfața grafică de acţionare şi conducere realizată în mediul LabVIEW este prezentată în Fig 5.3 iar în Fig.5.4 este prezentat panoul de monitorizare şi comandă.

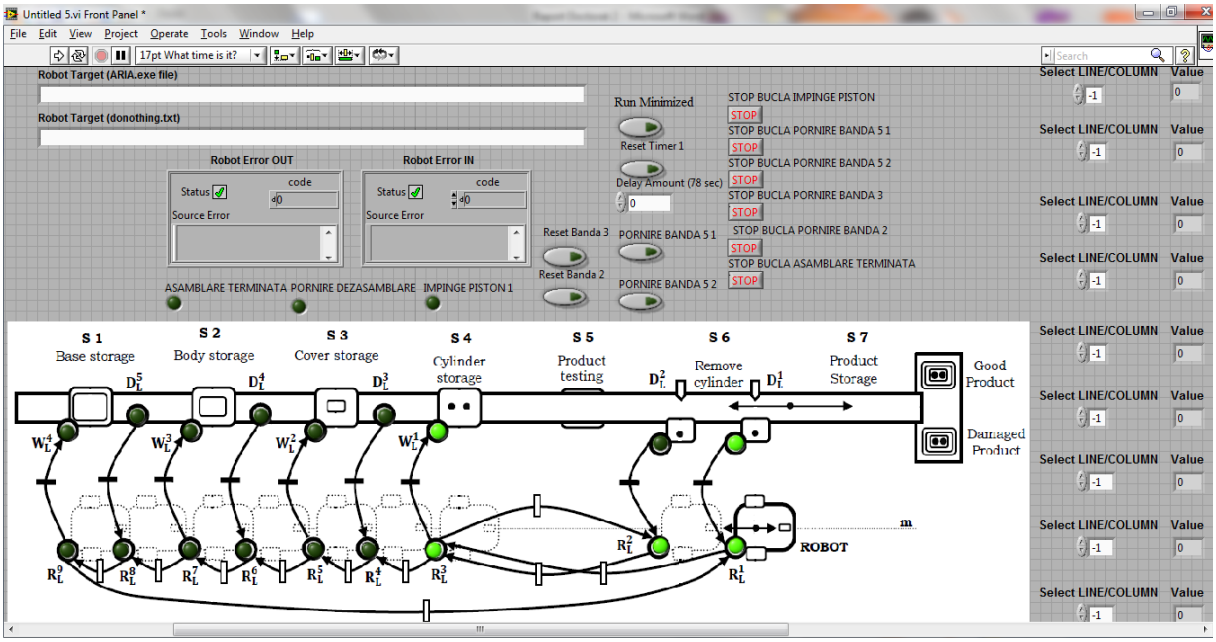


Fig.5.3. Interfața grafică utilizator, în LabView, pentru conducerea A/DML deservită de WMR echipat cu RM

Se poate observa în partea de sus a interfeţei două câmpuri unde se vor scrie, înainte de lansarea aplicaţiei, căile unde sunt salvate executabilul si fişierul text, necesare blocului care face pornirea automată a execuţiei ciclului robotului. Chenarul care are deasupra eticheta cu numele “Robot Error OUT”, din Fig.5.5, afişează un mesaj atunci când ale loc o eroare la execuţia, fie a fişierului text, fie a fişierului executabil. Similar chenarului descris mai sus, chenarul de sub eticheta “Robot Error IN” are rolul de a afişa un mesaj de eroare atunci când la calea specificată nu se găsesc fişierele indicate. Lângă chenarul “Robot Error IN” se poate observa un comutator denumit “Run Minimized”, care trebuie acţionat înainte de lansarea în

execuţie a interfeţei, având rol de a împiedica lansarea în execuţie a *n* fişiere text atunci când fişierul executabil nu este gata de execuţie. La interfaţa utilizator din LabView se mai poate observa sub eticheta “Select Line/Column” un indicator ce are rolul de a impune direcţia în care se citeşte din fişierul text, primul rând în jos în acest caz. Sub eticheta “Value” este un bloc ce arată valoarea citită din fişierul text. Lângă acest bloc denumit “Value” se află un indicator de tip LED ce are rolul de a arăta momentul când se produce acţiunea în cauză, în exemplul de faţă apucarea primului cilindru de către gripper. Pentru a permite utilizatorului posibilitatea de a comanda direct benzile liniei de mecatronică, s-a optat pentru adăugarea de comutatoare, acestea find în număr de 5 şi având numele benzii pe care o controlează. Interfaţa utilizator mai dispune de încă doi indicatori care arată când se petrec două din momentele cheie ale sincronizării liniei de mecatronică cu platforma mobilă, şi anume terminarea asamblării şi acționarea primului piston pentru dezasamblare cilindru 1.

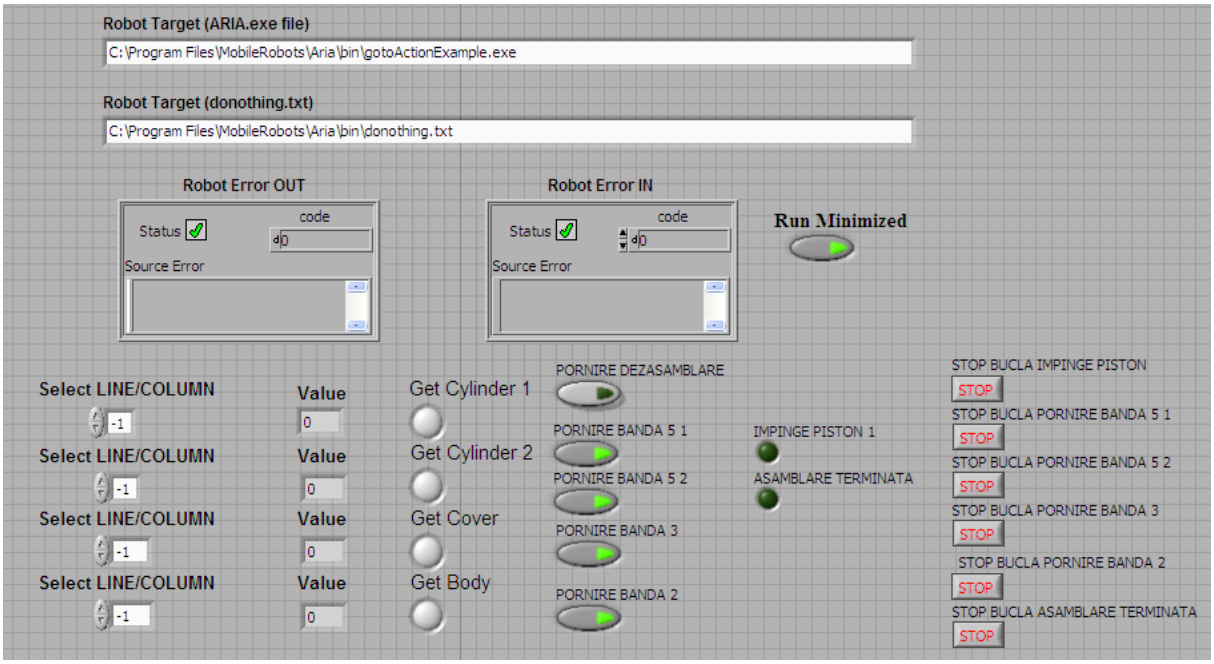


Fig.5.4. Interfața de monitorizare și comandă

5.1.4. Testarea structurii de conducere în MobileSIM

S-a testat conducerea A/DML deservită de WMR echipat cu RM, mai întâi, prin simulare în MobileSim, [94]. Algoritmul de conducere a platformei mobile a fost sliding-mode cu timp continuu, iar structura de conducere, hardware şi software a fost cea din Fig.5.1 şi 5.2. S-au apelat funcţiile din ARIA, [95], iar programul de conducere, atât în regim de simulare, cât şi în timp real, este elaborat în Visual C++. În cadrul simulării s-au impus anumite condiții de parcurgere a traiectoriei de către platforma mobilă Pioneer P3-DX în conducere sliding-mode [71]-[77], [90], [91]. Traiectoria simulată, Fig.5.5, corespunde procesului real de deservire a A/DML HERA&HORSTMANN în cadrul procesului de dezasamblare deservit de WMR echipat cu RM. Astfel, s-au impus următoarele condiții de simulare:

- Distanța totală parcusă de platforma mobilă este de 7960 mm;
- Viteza WMR a fost de 94 mm/s;
- În cadrul simulării nu s-au luat în considerare timpii de staționare ai WMR pentru efectuarea operațiilor de preluare sau eliberare piesă de către RM;

- Traiectoria parcursă a fost în linie dreaptă, deoarece în procesul real, prin construcția liniei flexibile, nu au existat locuri în care platforma mobilă era nevoită să efecteze o curbă sau o întoarcere.

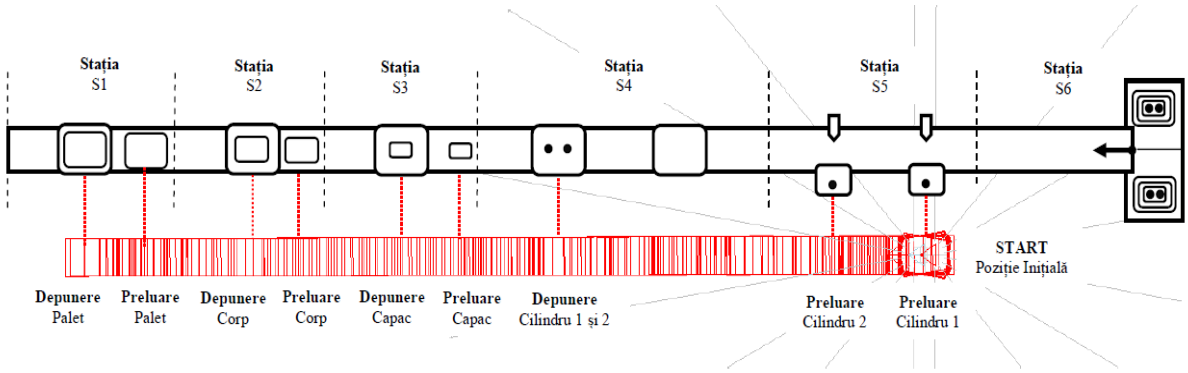


Fig.5. 5. Simularea în MobileSim a traiectoriei parcursă de WMR deservind A/DML

5.1.5. Conducerea în timp real

Pentru conducerea în timp real a tehnologiei hibride de fabricație flexibilă pe FMML, A/DML HERA&HORSTMANN, deservită de WMR, Pioneer 3-DX, echipat cu RM, Pioneer 5-DOF Arm, au fost programate în LabView şi Visual C++ următoarele acţiuni:

- Emitere un semnal digital la momentul depozitării în magazia stației S6 a unei piese cu cilindri de materiale diferite (stabilită prin convenție că este o piesă care nu a trecut testul de calitate). La recepţia semnalului de stare la intrarea digitală a plăcii de achiziție pe pinul 28, Fig.2.15, se va trimite un semnal la ieşirea digitală a plăcii de achizitie pe pinul 17;
- Recepție semnal la intrarea digitală I4.0 a modulului de I/O al PLC-ului și declanșează ”START proces dezasamblare”, Fig.5.6;
- Transport piesă pentru dezasamblare cu liftul stației S6 și pe benzile trasportoare ale stației nodale și stației S 5, până în dreptul primei locații de dezasamblare, pistonul D_L^1, Fig.5.3;
- STOP A/DML, banda transportoare stație S5, START dezasamblare cilindru 1, scriere semnal de stare pe ieşirea digitală Q0.0 a PLC-ului. Semnalul este achiziţionat de placă de achiziţie pe intrarea digitală aferentă pinului 28. Acest semnal de stare declanșează acțiunea de ”*START (ciclu continuu) WMR echipat cu RM”,* Fig.5.7;

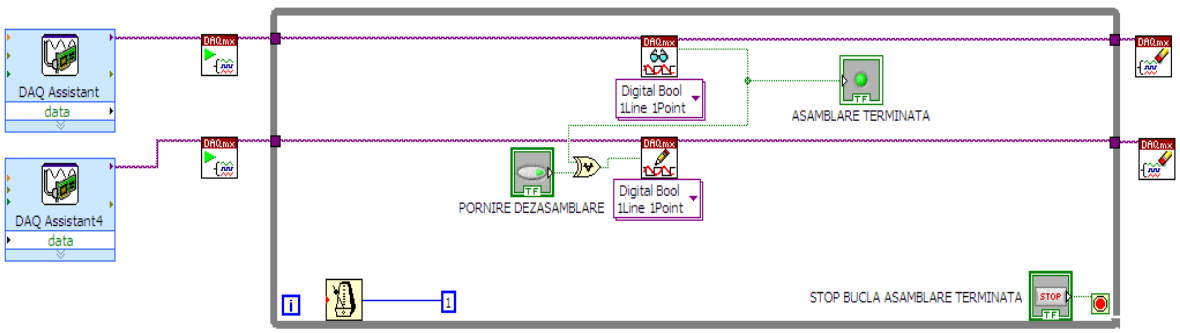


Fig.5.6. Implementarea în LabVIEW, “START dezasamblare”

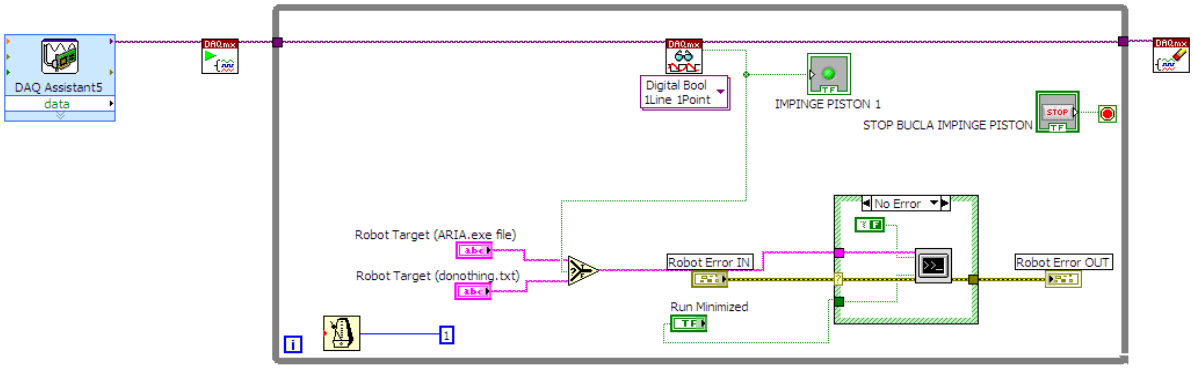


Fig.5.7. Implementarea în LabView, “START WMR echipat cu RM”

- Poziționare RM, prindere ”cilindru 1” prin închidere gripper și scriere în fişier text un caracter numeric care reprezintă o variabilă de stare, fişier text, deplasare WMR, poziționare RM locație magazie cilindri, eliberare ”cilindru 1”, deplasare WMR la următoare locație de dezasamblare;
- Deschidere fișier text de către programul LabVIEW pentru a scrie valoarea variabilei la ieşirea digitală a plăcii de achiziție de la pinul 21. Semnalul ajunge la intrarea digitală I2.0 a PLC-ului care declanşează START bandă transportoare stație S5, transport piesă la următoarea locație de dezasamblare, piston D_L^2 , Fig.5.8;
- STOP banda transportoare S5 și START dezasamblare ”cilindru 2”;

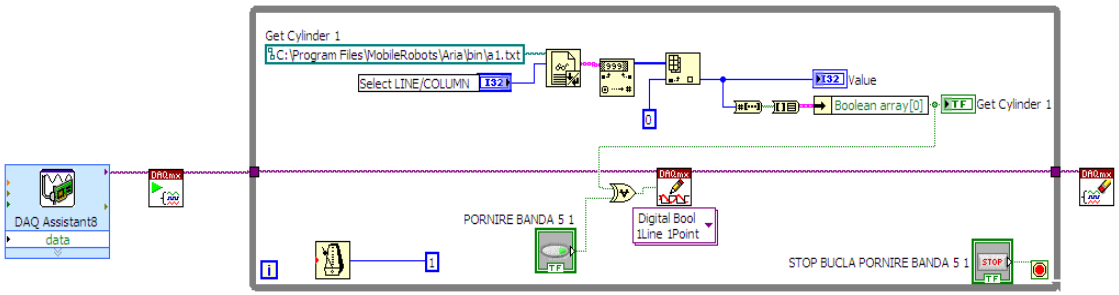


Fig.5.8. Implementarea în LabVIEW, “START bandă stație S5”

- Scriere în fişier text variabilă de stare, cifra 1, de către programul de conducere WMR echipat cu RM, după închidere gripper, prindere ”cilindru 2”. Programul LabVIEW trimite la ieşirea digitală a plăcii de achiziție de la pinul 19 un semnal care va declanșa START banda transportoare pentru a deplasa produsul la următoarea locație de dezasamblare, Fig.5.9;

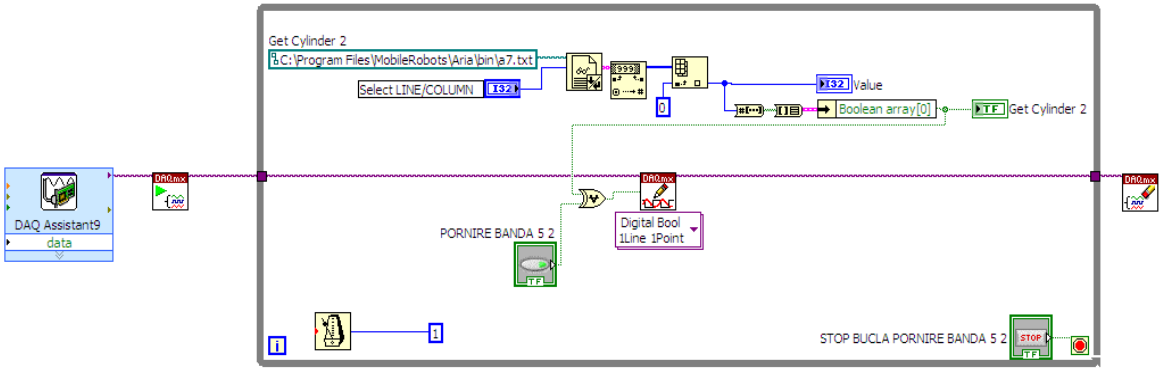


Fig.5.9. Implementarea în LabVIEW, “START bandă stație S4”

Fig.5.10. Prindere/Eliberare “cilindru”

- Deplasare WMR, poziționare RM locație magazie cilindri, eliberare ”cilindru 2”, deplasare WMR de la locația magaziei de cilindri, la stația S3, în dreptul senzorului E15.5, unde se află următoarea locație de dezasamblare;
- STOP bandă S4, START dezasamblare ”capac;
- Prindere capac, închidere gripper, scriere în fisierul text variabilă de stare. Programul preia valoarea variabilei din fişier şi activează ieşirea digitală numărul 17 a plăcii de achiziție, care va trimite un semnal la intrarea digitală I0.0 a PLC-ului care produce START banda S3, transport piesă pană la senzorul E13.5, Fig.5.11, Fig.5.12;
- Deplasare WMR, poziționare RM locație magazie , eliberare ”capac”, deplasare WMR de la stația S3, locația magaziei de capace, la stația S2, în dreptul următoarei locații de dezasamblare, locație dezasamblare ”corp”;

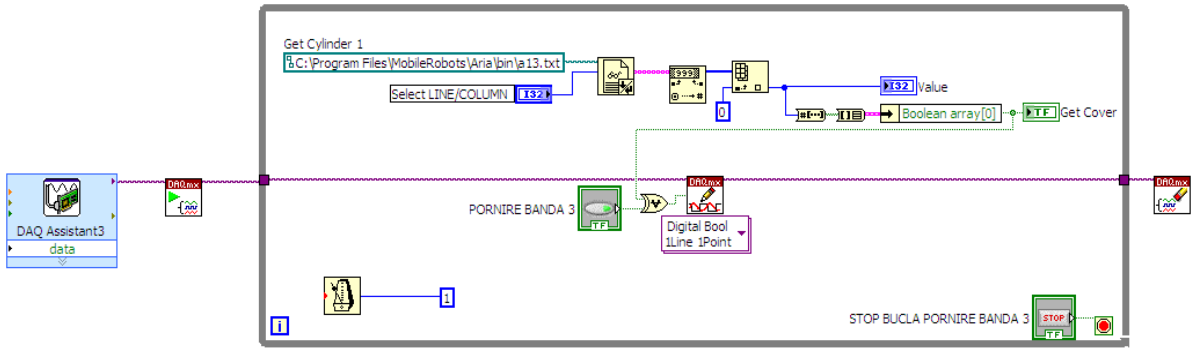


Fig.5.11. Implementarea LabView, “START bandă stație S3”

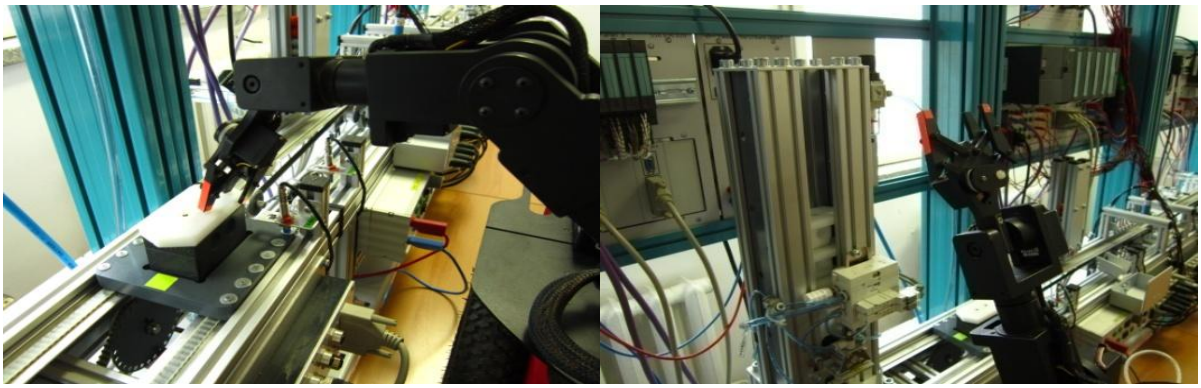


Fig.5.12. Prindere/Eliberare “capac”

- La dezasamblare ”corp” se efectuează aceeași procedură de sincronizare a deplasării WMR echipat cu RM până la senzorul E13.5, prindere ”corp”, START bandă stație S2, deplasare palet până în dreptul senzorului E11.5 al stației S1, unde este efectuată ultima operație de dezasamblare, Fig.5.13, Fig 5.14.

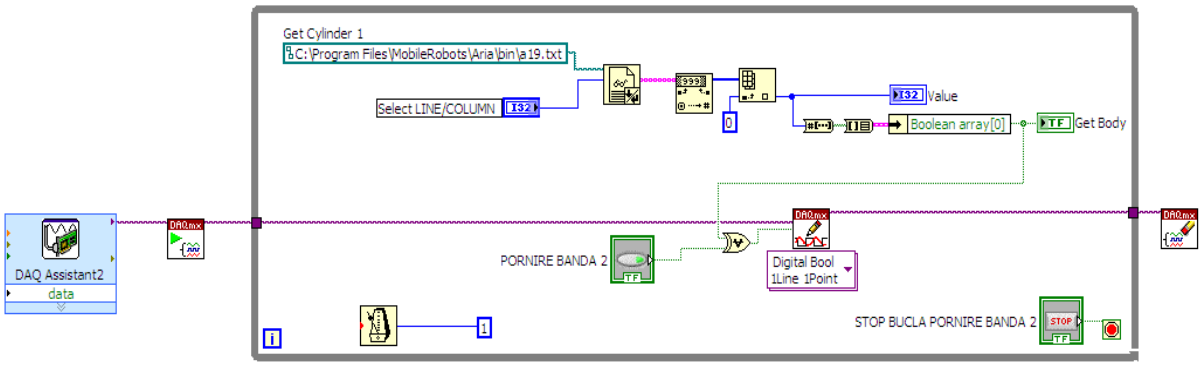


Fig.5.13. Implementarea în LabView “START bandă stație S2”

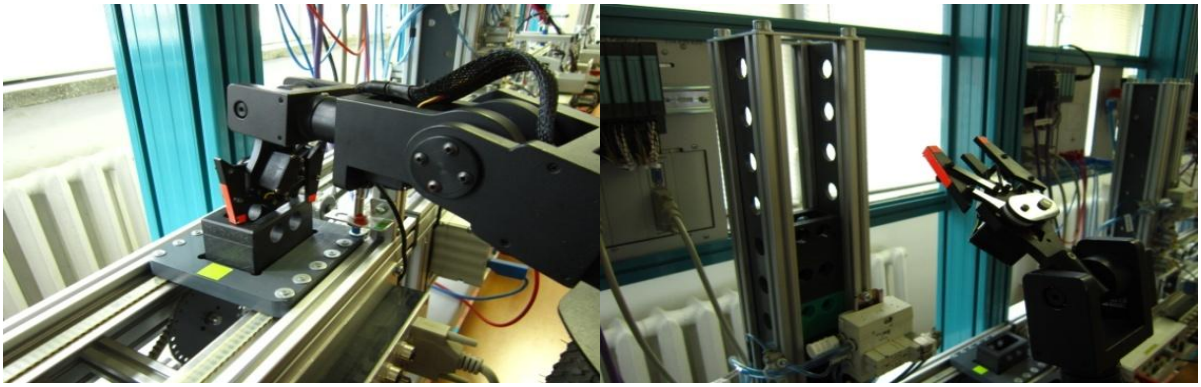


Fig.5.14. Prindere/Eliberare “corp”

- La ultimul ciclu elementar de dezasamblare se execută: ”palet” din dreptul senzorului E11.5, închidere gripper, deplasare WMR echipat cu RM până la locația magazine 1 paleți, deschidere gripper, eliberare palet, deplasare WMR cu RM până la locația primului punct de dezasamblare, dezasamblare ”cilidru 1”, parcare RM, Fig.5.15.

Astfel, WMR echipat cu RM intră în aşteptarea declanșării unui nou proces de dezasamblare. S-au prevăzut două elemente de control destinate preluării de la utilizator a path-urilor: elemente de tip executabil și elemente de tip text. Prin lansarea aplicației de conducere se vor lansa în mod automat aceste două elemente. Elementul de tip executabil este aferent conducerii robotului mobil, acesta este inițializat și executat în mod automat de către programul LabVIEW la fiecare declanșare a operației de dezasamblare. În cadrul elementului de tip text sunt salvate variabile de stare din procesul de dezasamblare. Variabile care sunt citite prin intermediul plăcii de achiziție și salvate în LabVIEW. Prin aceste două tipuri de variabile se execută sincronizarea procesului de dezasamblare a liniei flexibile HERA&HORSTMANN cu robotul mobil Pioneer P3-DX echipat cu manipulator.

Fig.5.15. Prindere/Eliberare “palet”

În Fig.3.8 sunt date distanțele (Rd_1 ... Rd_8) pe care le parcurge robotul mobil echipat cu manipulator în cadrul unui ciclu complet de dezasamblare a unui produs.

- $Rd_1 + Rd_2 + Rd_2 = 300 + 730 + 730 = 1760\,mm$: reprezintă distanța parcursă de WMR echipat cu RM în primul ciclu elementar, dezasamblare “cilindru 1”;
- $Rd_2 + Rd_3 = 730 + 370 = 1100\,mm$: reprezintă distanța parcursă WMR echipat cu RM în al doilea ciclu elementar, dezasamblare “cilindru 2”;
- $Rd_4 + Rd_5 = 310 + 450 = 760\,mm$: reprezintă distanța parcursă WMR echipat cu RM în al treilea ciclu elementar, dezasamblare “capac”;
- $Rd_6 + Rd_7 = 330 + 420 = 750\,mm$: reprezintă distanța parcursă WMR echipat cu RM în al patrulea ciclu elementar, dezasamblare “corp”;
- $Rd_8 + Rd_5 = 340\,mm$: reprezintă distanța parcursă WMR echipat cu RM în ultimul ciclu elementar, dezasamblare “palet”;
- $Rd_8 + Rd_7 + Rd_6 + Rd_5 + Rd_4 + Rd_3 + Rd_2 + Rd_1 = 340 + 420 + 330 + 450 + 310 + 370 + 730 + 300 = 3250\,mm$: reprezintă distanța parcursă WMR echipat cu RM la întoarcerea în poziția inițială;
- viteza robotului mobil echipat cu manipulator este de $94\,mm/s$.

În cadrul procesului de testare în timp real din mediul de programare grafică LabView se poate observa în Fig 5.16 evoluția stărilor continue ale WMR și a stărilor discrete ale RM într-un ciclu complet de dezasamblare.

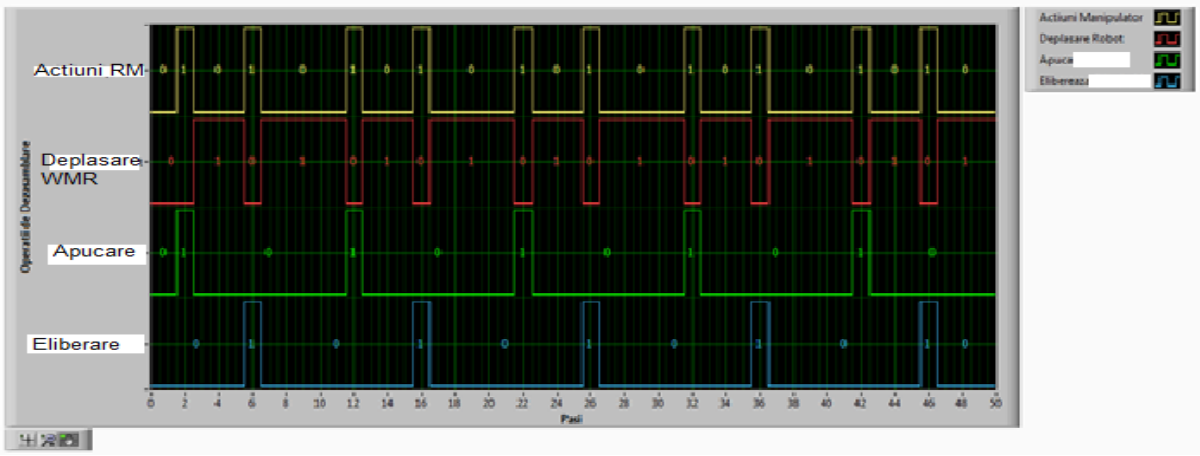


Fig.5.16. Deplasare WMR, poziționare/prindere/eliberare RM, într-un ciclu complet

În Fig.5.17 sunt prezentate acțiunile RM gripper, prindere/eliberare într-un ciclu complet de dezasamblare.

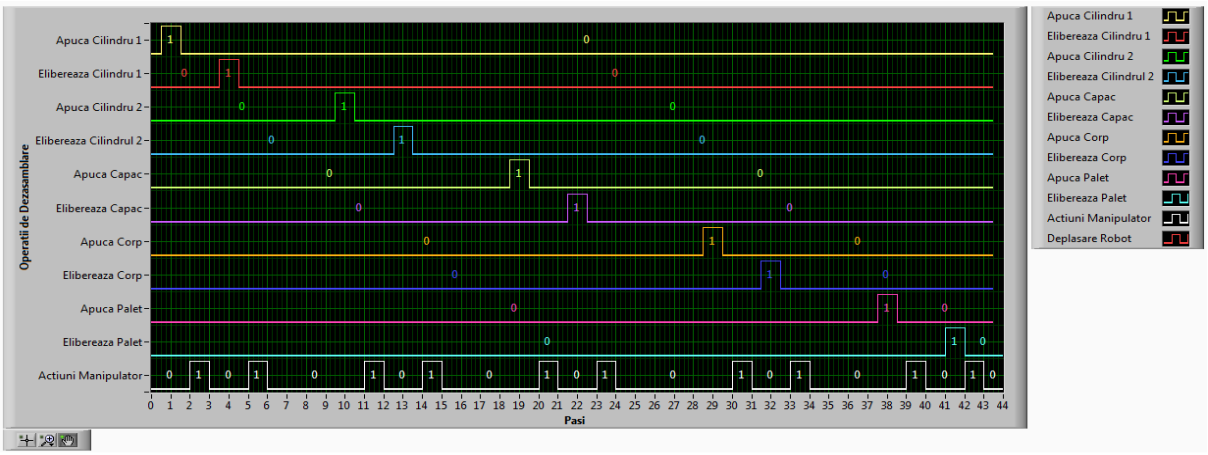


Fig.5.17. Acțiuni RM gripper într-un ciclu complet de dezasamblare

În Fig.5.18 sunt prezentate acțiunile de deplasare WMR şi manipulare RM în cadrul procesului de dezasamblare.

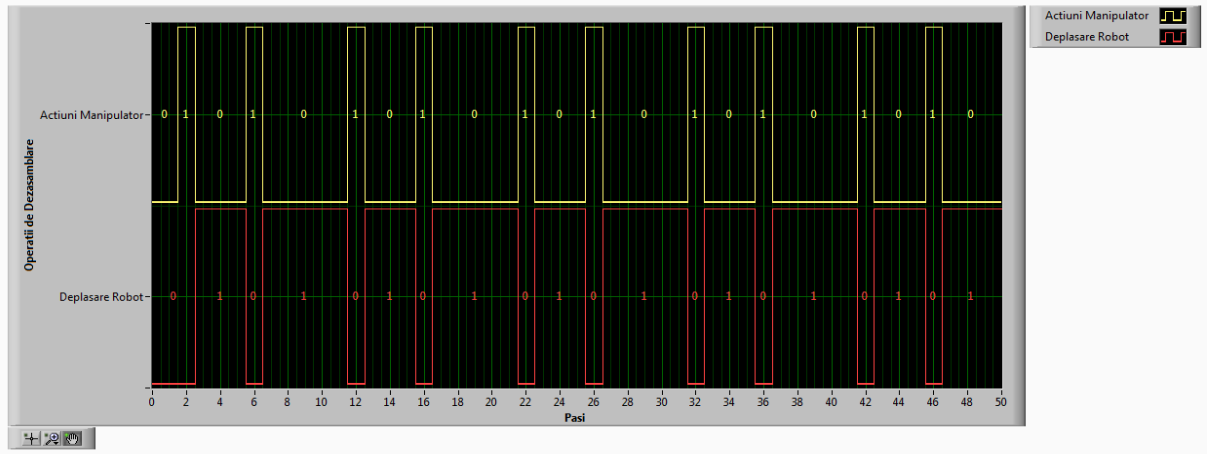


Fig.5.18. Acțiuni WMR echipat cu RM într-un ciclu complet de dezasamblare

În urma execuției programului de conducere a WMR echipat cu RM, program scris în Visual Studio C++, este generat un fişier text care conţine o matrice de 21 de coloane şi 1200 de linii. În această matrice sunt salvate valorile următoare: robot.getX()/1000, robot.getY()/1000, robot.getTh()*(3.14/180), ref_xr, ref_yr, ref_Th, x_e, y_e, theta_e, x_e_der, y_e_der, theta_e_der, (robot.getVel()/1000), (robot.getRotVel()*3.14/180), v, w, v_c, w_c, s_1, s_2, t.

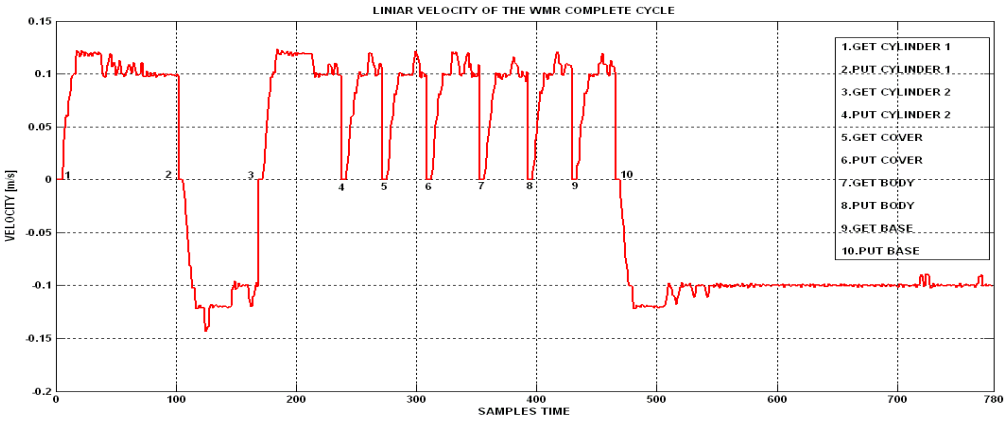


Fig.5.19. Viteza liniară WMR într-un ciclu complet de dezasamblare

În Fig.5.19, se poate urmări viteza liniară reală pe întreg ciclul robotului. Deoarece, în secvenţele pe care le execută, în tandem cu linia de mecatronică, sunt momente de timp când robotul efectuează simulări ale prinderii/eliberării componentelor produsului finit supus dezasamblării, cu RM, Pioneer 5-DOF Arm, pe grafic se poate observa că viteza liniară este nulă. Tot în grafic se poate constata că viteza liniară de deplasare a robotului mobil are şi valori negative, viteză care corespunde mersului înapoi al WMR, adică la preluarea cilindru şi la revenirea în poziţia iniţială robotul se întoarce cu spatele dinspre magazia de bolţuri înspre cutia cu al doilea bolţ scos şi atunci când robotul vine în poziţia iniţială imediat după ce a simulat depozitarea paletului la magazia corespunzătoare. În Fig.5.20 se observă graficul radicalului sumei pătratelor erorilor de poziţie, x_e şi y_e.

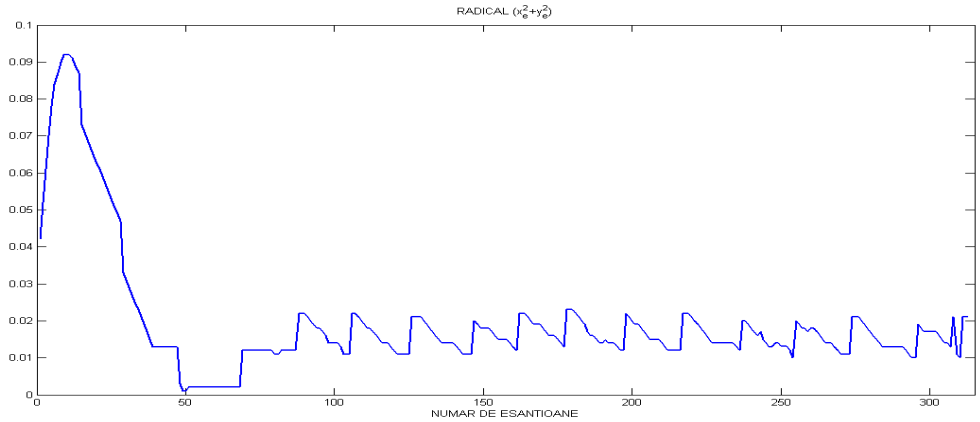


Fig.5.20. Eroarea longitudinală şi eroarea laterală

Traiectoria impusă robotului mobil este una liniară, deoarece se doreşte ca acesta să se deplaseze paralel de-a lungul A/DML. În Fig.5.21 sunt prezentate tranziţiile de stare ale componentelor detașate în urma operaţiilor de dezasamblare.

5.2. Conducerea A/DML HERA&HORSTMANN deservită de doi WMRs operând în paralel, sincron

A/DML este deservită de doi WMRs, dintre care unul este echipat cu RM, utilizat pentru manipulare, iar al doilea fară RM, este utilizat pentru transport. Ambele platforme robotice deservesc A/DML în procesul de dezasamblare. Cele două sisteme robotice operează în paralel, sincron. WMR-ul (WMR1) echipat cu manipulator (RM1) preia componenta de la locația unde se produce dezasamblare și o depune pe platforma (WMR2). Apoi WMR1 și WMR2 se deplaseaza simultan la locația de depozitare, unde RM1 preia componenta de pe platforma WMR2 și o depune în magazie. Aplicația de supervizare, programată in Visual C++ (Anexa 2), este implementată și rulează pe un laptop, aflat la distanță care monitorizează procesele de A/D, execuția și sincronizarea taskurilor pe cele trei subsisteme, A/DML, WMR1 echipat cu RM1 și WMR2. Supervizorul comunică cu cele două sisteme robotice prin protocol TCP/IP, iar cu A/DML prin intermediul unei plăci de achiziție, NI-USB-8008 conectata la I/O digitale ale PLC-ului. Supervizorul asignează taskurile sitemelor robotice, în ceea ce privește, deplasarea între locațiile de dezasamblare și magaziile de depozitare, poziționare RM1, cât și închiderea/deschiderea gripperului. De asemenea, supervizorul comandă oprirea/pornirea benzilor transportoare ale stațiilor de lucru și sincronizarea acțiunilor A/DML cu acțiunile WMR1, RM1 și WMR2. În Fig.5. 22 este prezentată structura software, pentru conducerea în timp real, iar în Fig.5. 23, structura hardware.

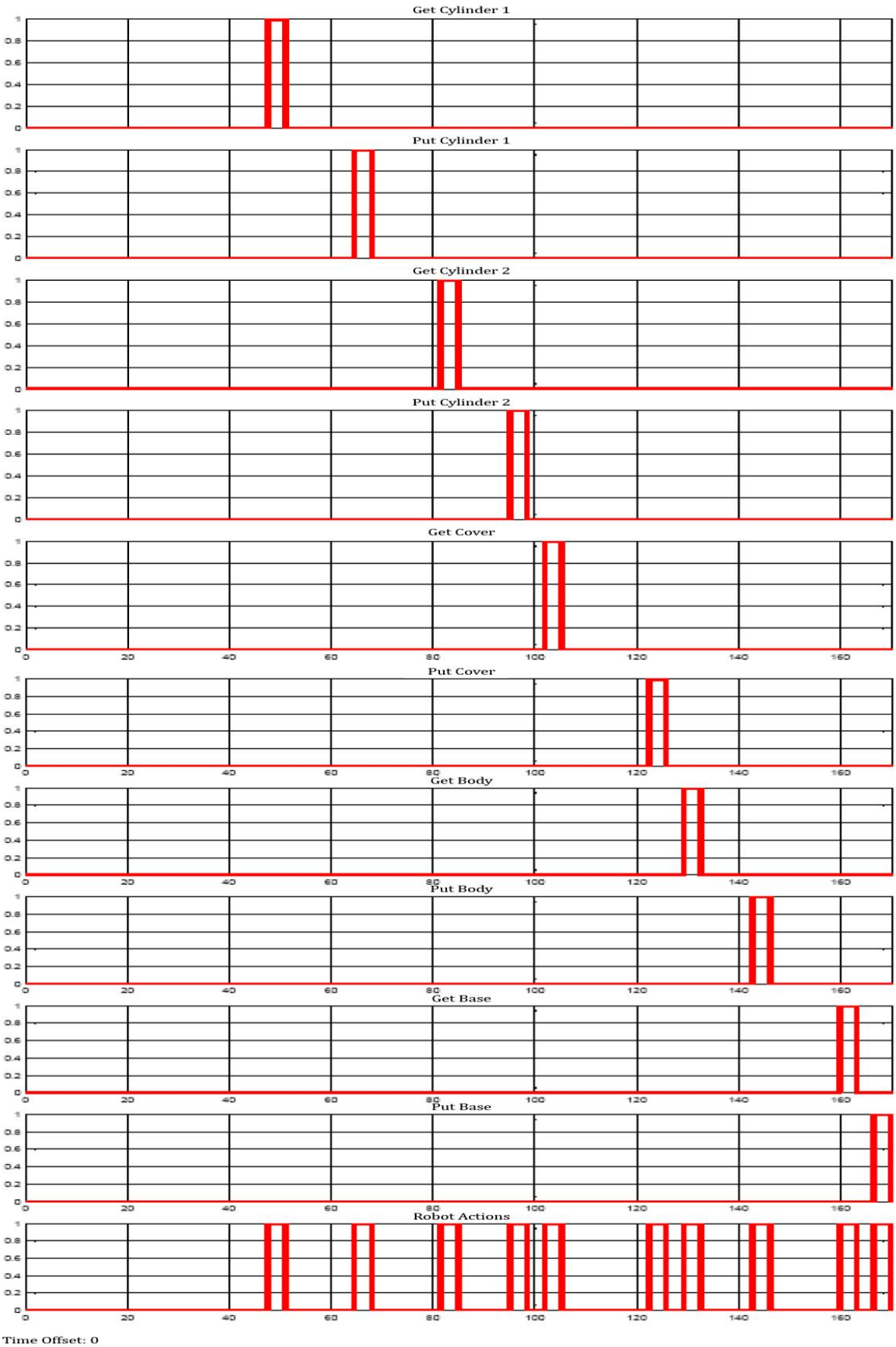


Fig.5.21. Tranzițiile de stare ale componentelor dezasamblate şi transportul lor de la locațiile de dezasamblare la locațiile de depozitare

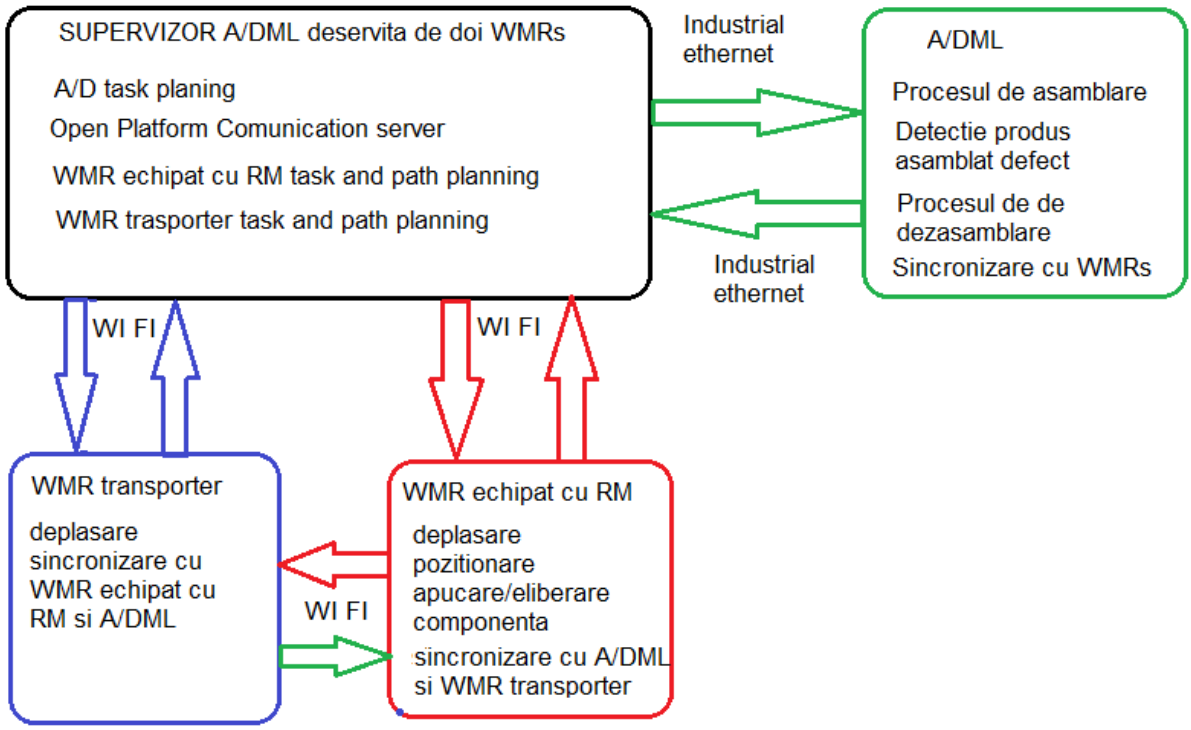


Fig.5.22. Structura software pentru conducerea A/DML deservită de doi WMRs

5.3. Conducerea FMML, P/RML FESTO MPS-200 deservită de un WMR echipat cu RM

5.3.1. Testarea conducerii în MobileSim

S-a testat conducerea P/RML deservit de WMR echipat cu RM, mai întâi, prin simulare în MobileSim. Algoritmul de conducere a platformei mobile a fost sliding-mode cu timp continuu, iar structura de conducere, hardware și software a fost cea din Fig.1.2. S-au apelat funcţiile din ARIA, iar programul de conducere, atât în regim de simulare, cât şi în timp real, este elaborat în Visual C++. În cadrul simulării s-au impus anumite condiții de parcurgere a traiectoriei de către platforma mobilă Pioneer P3-DX în conducere sliding mode [71]-[77]. Traiectoria simulată, Fig.5.24, corespunde procesului real de deservire a P/RML FESTO MPS-200 în cadrul proceselor de P/R deservit de o platformă mobilă echipată cu manipulator. Softul ARIA realizează conectarea automată la simulatorul MobileSim în cazul în care nu este detectat niciun WMR conectat la portul COM1. Simulatorul are implementate modelele cinematice ale roboților, funcții pentru simularea sonarelor și laserelor, care sunt folosite pentru a simula comportamentul unui robot real. Programul scris în C++ apelează funcțiile ARIA în cazul în care se dorește trimiterea unor comenzi către simulator sau citirea datelor simulate. Pentru simularea în MobileSim, s-au utilizat ecuațiile 1.17 și 1.18 la conducerea WMR, Pioneer P3-DX, în SM-TT, în cadrul procesului de deservire a liniei flexibile FESTO MPS-200.

În cadrul simulării s-au impus următoarele condiții:

- Distanța totală parcusă de WMR echipat cu RM este de 5634 mm. Această traiectorie este parcursă în două etape de preluare/eliberare și întoarcere la poziția inițială. Fiecare etapă poate fi împărțită în 3 secțiuni (fiecare etapă are o distanță de 2817 mm): în prima secțiune se parcurge distanța de 700 mm în linie dreaptă (din poziția de *START)* după

care platforma mobilă execută o întoarcere de 90^o apoi parcurge în linie dreaptă distanța de 2097 mm care corespunde celei de-a doua secțiuni, execută cea de-a doua întoarcere de 90^o și parcurge cea de-a treia secțiune (până la poziția de *STOP*) cu o distanță de 200 mm; După parcurgerea celor 3 secțiuni în care se transportă o piesă, WMR echipat cu RM revine în poziția inițială parcurgând aceeași traiectorie;

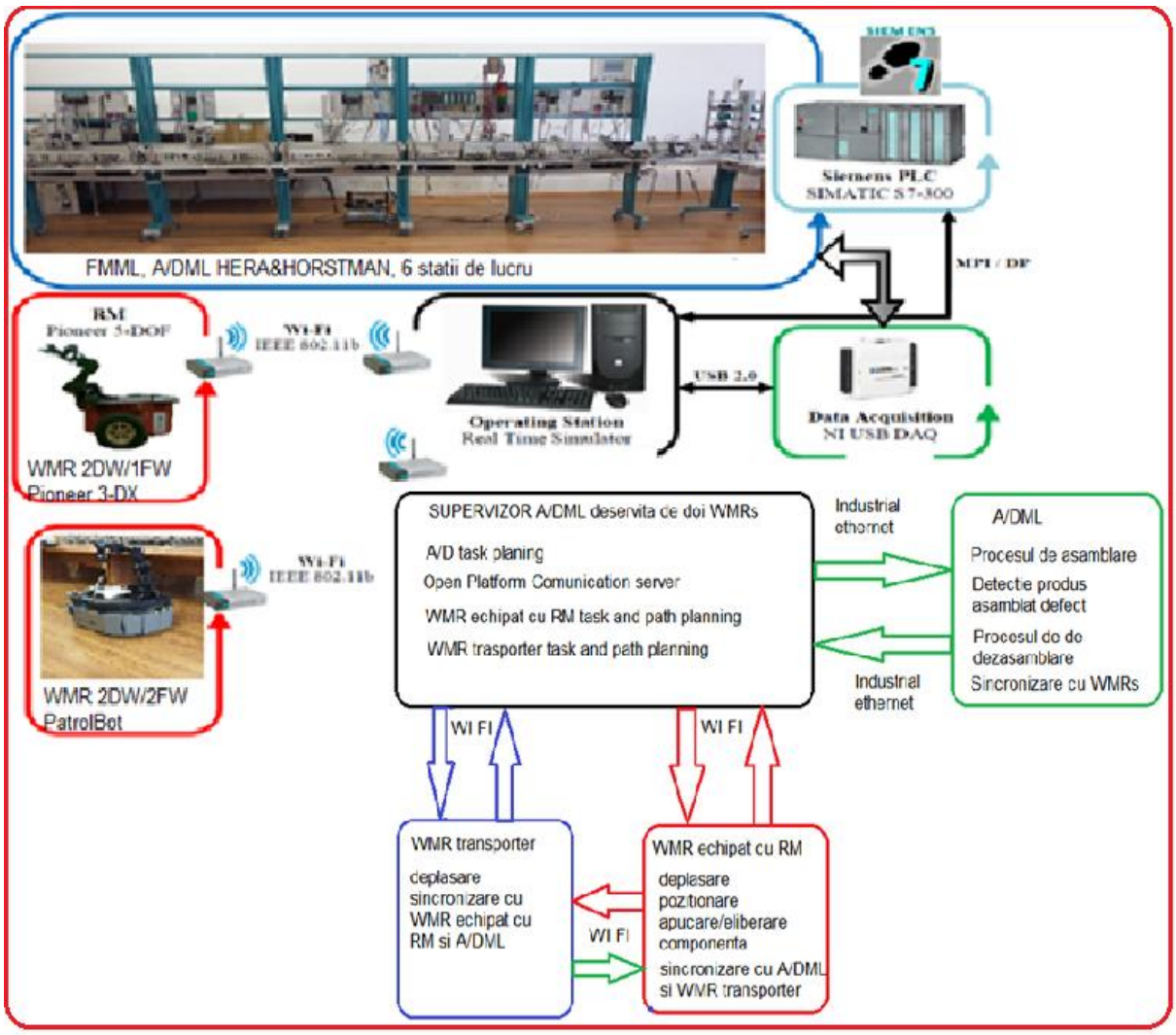


Fig.5.23. Structura hardware și software pentru conducerea A/DML deservită de doi WMRs

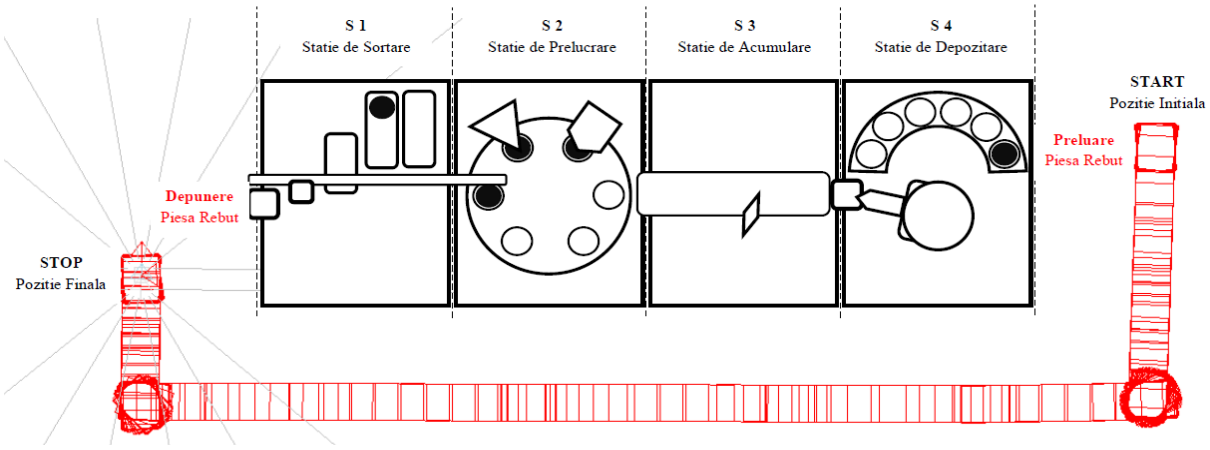


Fig.5.24. Simularea traiectoriei în MobileSim la conducerea SM-TT a WMR echipat cu RM

- Viteza platformei mobile a fost de 94 mm/s;
- În cadrul simulării nu s-au luat în considerare timpi de staționare WMR pentru efectuarea operațiilor de preluare/eliberare piesă de către RM;
- În cadrul simulării, WMR echipat cu RM preia o singură piesă care nu satisfice cerințele de calitate și o readuce pe linia de prelucrare după care WMR se întoarce la poziția inițială.

5.3.2. Conducerea în timp real cu sistem servoing vizual

Programul de conducere a întregului proces de P/R este împărțit în mai multe bucle locale de conducere după cum urmează:

- prima buclă de conducere are ca obiectiv controlul procesului de procesare a liniei flexibile FESTO MPS-200. Această buclă de conducere este implementată în PLC SIEMENS S7-300 cu procesor 313C-2 DP, ea fiind programată în mediul de programare STEP 7;
- a doua buclă de conducere locală o reprezintă conducerea WMR, Pioneer P3-DX echipat cu RM, Pioneer 5-DOF Arm.

Comunicația dintre aceste două bucle locale de conducere se realizează printr-o interfațare dintre linia flexibilă și un calculator de proces, realizată prin intermediul a două camere web de înaltă rezoluție/apertură și o comunicație wireless pe un protocol TCP/IP dintre robotul mobil și calculatorul de proces. Camerele web sunt fixe, localizate, prima deasupra stației de depozitare/sortare, cu vedere asupra depozitului de piese de reprocesat, iar cea de-a doua deasupra intrării în linia de procesare, ambele conectate la calculatorul de proces. Pe acest calculator de proces se găsește programul de detecție a piesei de reprocesat, program care are rolul de a sincroniza în timp real cele două bucle de conducere în cadrul procesului. Schema bloc a comunicației dintre echipamentele utilizate în conducerea liniei flexibile FESTO MPS-200 deservită de un WMR, Pioneer P3-DX echipat cu RM, Pioneer 5-DOF Arm poate fi vizualizată în Fig 5.25.

Aplicația de timp real este realizată pe o platformă MATLAB și este bazată pe modelul SHPN din Fig.4.16, care interfațează modelul HPN cu semnalele de sincronizare preluate de la video camere. După sincronizare, WMR echipat cu RM va prelua piesa de pe stația de depozitare și o va transporta pe staţia de manipulare. Gripperul este poziționat pentru apucare sau eliberare piesă de către sistemul servoing vizual. Pe gripper-ul manipulatorului robotic există 2 markere care ajută în detecția video a poziției manipulatorului. Programul Matlab, prin intermediul prelucrării de imagine, detectează prezența piesei și îi determină coordonatele în imagine. În funcție de poziționarea RM, determinată prin calcularea coordonatelor în imagine a celor 2 markere de pe gripper, aplicația scrie un mesaj într-un fișier text. Conținutul fișierului text este în permanență citit de programul de conducere al robotului. Acesta, împreună cu programul în Matlab rulează în paralel pe calculatorul de proces. Robotul este așezat în poziția inițială de așteptare și inițializat cu brațul pregătit de preluarea piesei. Acesta părăsește poziția inițială doar în momentul detecției piesei de către camera video și se poziționează pentru extragerea piesei, în funcție de comenzile primite. Odată preluată piesa, programul de conducere a robotului rulează algoritmul de conducere SM pentru aducerea piesei la începutul liniei de procesare. În Fig. 5.26, se prezintă momentul detecției și utilizării camerei web pentru poziționarea în vederea recuperării piesei de reprocesat. Viteza liniară a WMR pe durata unui ciclu de transport complet este prezentată în Fig.5.27.

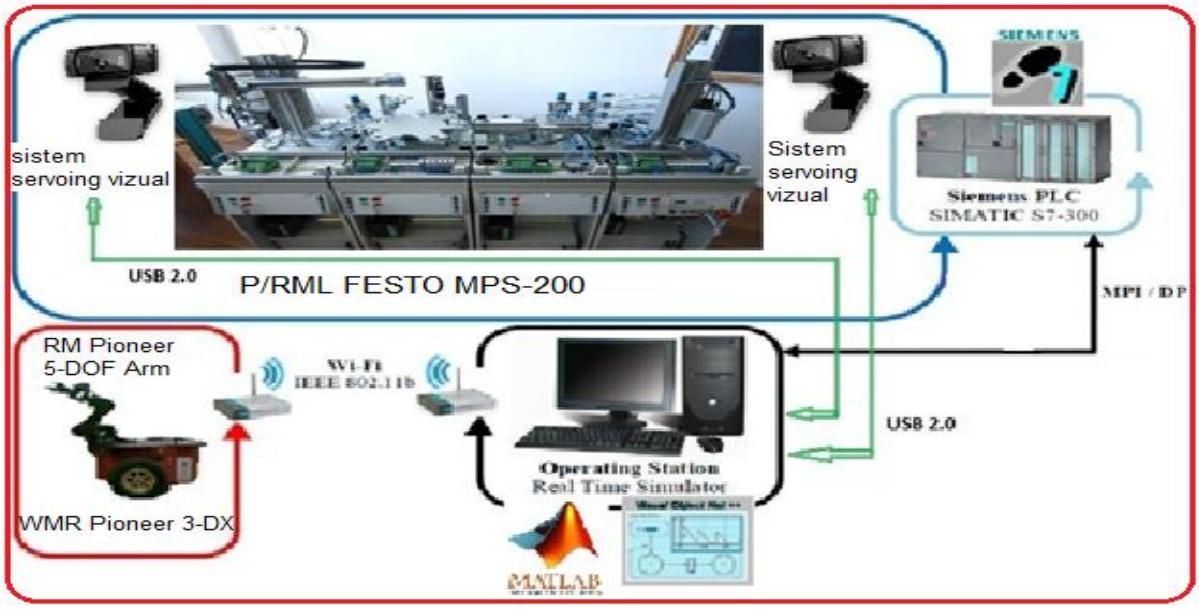


Fig.5.25. Schema bloc a comunicațiilor sistemului de procesare

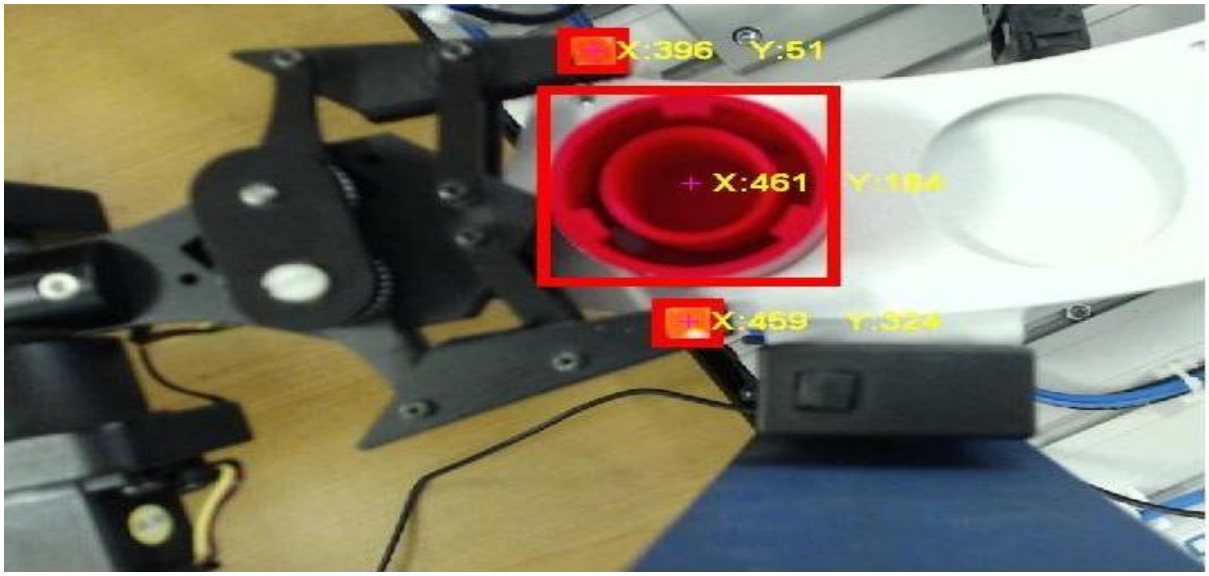


Fig.5.26. Utilizarea camerei web pentru sincronizare și poziționare gripper

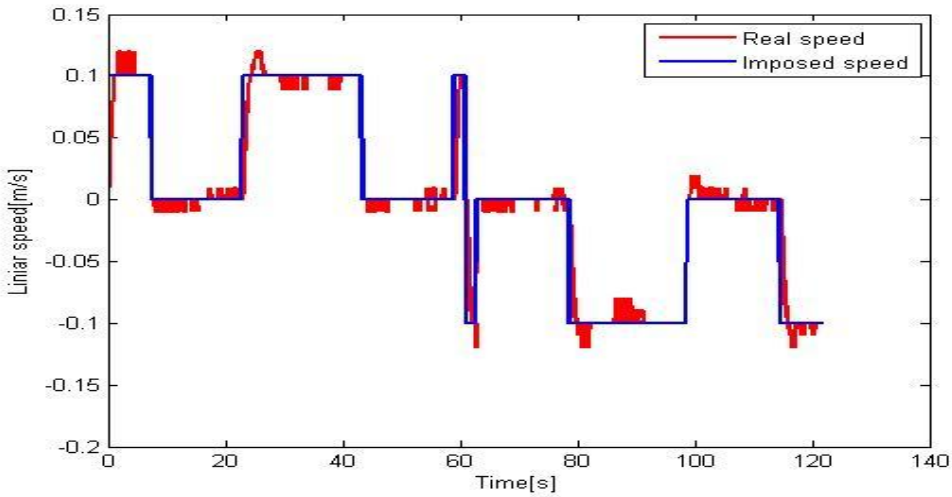


Fig.5. 27. Viteza liniară, impusă și reală, la conducerea WMR integrat în P/RML

5.4. Concluzii

Aplicațiile de acționare și conducere prezentate în acest capitol au vizat obiectivele enunțate în introducere, adică, implemetarea în laborator a unor tehnologii hibride, complet automatizate pe FMMLs deservite de sisteme robotice integrate, pentru un volum de producție dat. Astfel se pot revendica drept contribuții, modalitățile prin care s-a trecut de la modelele SHPN la aplicațiile de timp real, în ceea ce privește sincronizarea, compatibilizarea hardware/software dintre liniile de mecatronică și sistemele robotice, utilizarea unor platforme de programare și interfațare grafică care au permis monitorizarea proceselor în cauză de către utilizator.

Principalele contribuţii rezultate în urma implementării aplicațiilor de conducere în timp real au vizat următoarele:

- Testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe A/DML, HERA&HORSTMANN, deservită de un WMR echipat cu RM;
- Testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe A/DML, HERA&HORSTMANN, deservită de doi roboți mobili colaborativi, lucrând în paralel, sincron, unul dintre ei, Pioneer 3-DX, echipat cu un RM, Pioneer 6-DOF Arm, utilizat pentru manipulare, și un al doilea, PatrolBot, utilizat pentru transport;
- Testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe P/RML, FESTO-MPS 200, deservită de un WMR echipat cu RM.

Capitolul 6

Concluzii finale, contribuţii, direcţii de cercetare viitoare, diseminarea rezultatelor

Abordarea care se propune în teză răspunde noilor concepte de planificare și conducere a proceselor de fabricație flexibilă, de asamblare/dezasamblare, și de prelucrare/reprelucrare, pe sisteme de laborator compuse din linii de mecatronică deservite de platforme robotice mobile echipate cu manipulatoare. Aceste structuri de laborator au corespondent în industria reală, mai ales la procesele de asamblare și prelucrare din industria de automobile, la asamblarea caroseriei, a cutiei de viteze și a blocului motor. Actualmente, liniile de fabricaţie flexibilă sunt deservite de manipulatoare robotice care, în general, au o poziţie fixă. Prin cercetarea propusă în teză, s-a dorit o extindere la integrarea în liniile de fabricaţie flexibilă a roboţilor mobili echipaţi cu manipulatoare. În aceiași măsură, abordarea din teză raspunde și noilor cerințe, de recuperare subansamble sau componente pentru vânzare, reutilizare și reintroducere în fluxurile de fabricație a produselor care după ce au parcurs un proces tehnologic pe o linie de producție, nu corespund calitativ.

6.1. Concluzii finale

Consider ca prin cele 5 capitole la care se adaugă și Introducerea, teza răspunde integral obiectivelor formulate, obiective prin care mi-am propus să finalizez până la stadiul de implementare în timp real tehnologii de fabricație flexibilă pe linii de mecatronică deservite de sisteme robotice mobile. Am abordat aspecte ce țin de acţionarea roboților mobili și a liniilor de mecatronică, insistând pe compatibilizarea hardware a celor două entități, astfel încât sistemele robotice să poată fi integrate într-o măsură cât mai mare în procesele de fabricație pe două tipuri de linii de mecatronică, de asamblare/dezamblare și prelucrare/reprelucrare. Am utilizat instrumente de modelare specifice liniilor de fabricație flexibilă deservite de sisteme robotice: asignare și planificare taskuri, echilibrarea liniilor prin optimizare. Aș revedinca, ca fiind o premieră, abordare hibridă pentru modelare, prin utilizarea instrumentului SHPN, ţinând cont de natura discretă în funcționarea liniilor de mecatronică și natura continuă în deplasarea roboților mobili împreună cu poziționarea manipulatoarelor robotice. În cadrul formalismului SHPN, am prezentat conceptele în forma generală, prezentare urmată de particularizări la cazuistica pe echipamentele pe care le-am avut la dispoziție. Pentru a susține demersul din teză am avut la dispoziție echipamente specifice, A/DML HERA&HORSTMANN, P/RML FESTO MPS-200, WMRs: Pioneer 3-DX, PatrolBot, PeopleBot, PowerBot, RMs: Pioneer 5-DOF Arm, Cyton 1500, camere video de înaltă rezoluție și apertură, PLC-uri, PC-uri, plăci de achiziție, routere, etc. Pentru implementare am apelat la pachete software performante, Sirphyco, Visual C++, MATLAB, LabView, SIMATIC STEP 7, prin intermediul cărora am realizat și programat, simularea modelelor SHPN, comunicația între module și subsisteme, algoritmii de conducere, interfețele grafice utilizator, monitorizarea și sincronizarea. Clipurile video realizate, sunt o confirmare că implementările de timp real funcţionează, implementări bazate pe modelele SHPN prezentate.

6.2. Contribuții

Contribuţiile revendicate la acţionarea şi conducerea FMMLs deservite de roboţi mobili echipaţi cu manipulatoare, diseminate şi certificate prin lucrările publicate, la care am fost prim autoare sau co-autoare, sunt următoarele:

- Modelarea cinematică a WMRs: 2DW/2FW, PatrolBot şi 2DW/1FW, Pioneer 3-DX cât și acționarea și comanda RMs, Pioneer 5-DOF Arm și 7-DOF Cyton 1500. Pentru determinarea modelului cinematic al WMR am considerat variabilele generalizate ale sistemului, rotirea roților robotului fără alunecare și aplicat constrângerile nonholomice specifice acestui caz. Modelul rezultat astfel are cinci variabile: două variabile reprezintă centrul geometric al WMR, o variabilă reprezentând unghiul de direcție al robotului și alte două variabile reprezentând unghiurile fiecărei roți. Aceste variabile depind de vitezele de rotire ale roților. S-a simplificat acest model pentru că am dorit doar calculul coordonatelor carteziene ale centrului geometric și unghiul de direcție, înlocuind vitezele celor două roți cu viteza liniară și viteza unghiulară a robotului, [F1], [F2], [F16], [F17], [F18], [F21];
- Conducerea sliding-mode cu timp continuu și conducerea sliding-mode cu timp discret, conduceri bazate pe modelul cinematic al WMRs cu 2DW/2FW și 2DW/1FW, modelul erorilor de urmărire și dinamica erorilor de urmărire. S-au calculat comenzile pentru viteza liniară și viteza unghiulară a WMR, [F3], [F4], [F5], [F6], [F7], [F20];
- Sistemul de comunicație la distanță și de comandă a WMR cu 2DW/2FW și 2DW/1FW în protocolul client server. Considerând modelul cinematic al WMR, 2DW/2FW, PatrolBbot, erorile de urmărire, dinamica erorilor de urmărire, suprafețele de comutație și legea de conducere, toate în timp discret, s-au calculat comenzile pentru viteza liniară și viteza unghiulară a WMR 2DW/2FW, PatrolBot. Utilizând MobileSim s-a testat conducerea în urmărirea unei traiectorii (trajectory tracking) în formă de curbă deschisă (“S”), [F3], [F4], [F5], [F6], [F20, [F21];
- Utilizând MobileSim s-a testat comparativ conducerea sliding-mode, “trajectory tracking”, în regim simulat pentru trei traiectorii, cerc, pătrat și curba deschisă a WMR 2DW/1FW, Pioneer 3-DX şi 2DW/2FW, PatrolBot, [F3], [F4], [F5], [F6], [F20, [F21];
- Schema bloc de acționare a roților motoare și de conducere în buclă închisă cu structură de conducere sliding-mode, cu timp discret, echivalent discret cu extrapolator de ordin zero, [F9], [F16];
- Analiza FMMLs cu referire la două tipuri de procese industriale, de fabricaţie flexibilă, asamblare şi prelucrare şi corespondențele acestor două sisteme industriale, de fabricaţie flexibilă cu două FMMLs, A/DML şi P/RML, [F10];
- Obținerea unor structuri optimizate de fabricație flexibilă care permit cu aceleași staţii efectuarea de operaţii de A/D şi P/R, [F11], [F14];
- Ipotezele de lucru şi de funcţionare aferente FMML, A/DML cu N staţii de lucru deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare, [F11];
- Ipotezele de lucru şi de funcţionare aferente FMML, P/RML deservită de un sistem robotic mobil echipat cu manipulator, [F12], [F13], [F14], [23];
- Asignarea şi planificarea taskurilor pentru operaţiile derulate pe A/DML, cu N staţii de lucru, deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare, [F10], [F19], [F24];

- Echilibrarea A/DML cu *N* staţii de lucru, deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare, prin rezolvarea unei probleme de optimizare cu restricţii, [F11], [F14], [F18];
- Asignarea si planificarea taskurilor pentru operaţiile derulate pe A/DML, HERA&HORSTMANN, deservită de unul sau două sisteme robotice mobile echipate cu manipulatoare, [F11], [F14], [F19];
- Asignarea si planificarea taskurilor pentru operaţiile derulate pe P/RML FESTO MPS-200, deservită de un sistem robotic mobil echipat cu manipulator, [F12], [F13], [F14], [23];
- Structura, modelul SHPN şi formalismul general, cu evidenţierea submodelelor repetitive, pentru A/DML deservită de un WMR echipat cu RM, [F15], [F22];
- Structura, modelul SHPN şi formalismul particularizat, cu evidenţierea submodelelor repetitive, pentru A/DML HERA&HORSTMANN deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, [F15], [F22];
- Simularea în pachetul Sirphyco a modelului SHPN pentru A/DML HERA&HORSTMANN deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, primul şi ultimul ciclu elementar de dezasamblare, [F11], [F15];
- Structura, modelul SHPN şi formalismul particularizat, cu evidenţierea submodelelor repetitive, pentru A/DML HERA&HORSTMANN deservită de doi WMRs, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm şi PatrolBot, [F19], [F24];
- Simularea în pachetul Sirphyco a modelului SHPN pentru A/DML HERA&HORSTMANN deservită de doi WMRs, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm şi PatrolBot, primul ciclu elementar de dezasamblare, [F19], [F24];
- Structura, modelul SHPN şi formalismul particularizat pentru P/RML, FESTO MPS-200 deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, [F12], [F13], [F14], [F23];
- Simularea în pachetul Sirphyco a modelului SHPN pentru P/RML, FESTO MPS-200 deservită de WMR, Pioneer 3-DX echipat cu RM, Pioneer 5-DOF Arm, [F12], [F13], [F14], [F23];
- Implementare în LabView, testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe A/DML, HERA&HORSTMANN, deservită de un WMR echipat cu RM, [10], [F11], [F15], [F22];
- Implementare în Vizual C++, testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe A/DML, HERA&HORSTMANN, deservită de doi roboți mobili colaborativi, lucrând în paralel, unul dintre ei, Pioneer 3-DX, echipat cu un RM, Pioneer 6-DOF Arm, utilizat pentru manipulare, și un al doilea, PatrolBot, utilizat pentru transport, [F19], [F24];
- Implementare în Matlab şi Vizual C++, testarea și validarea în laborator a tehnologiei hibride de fabricație flexibilă pe P/RML, FESTO-MPS 200, deservită de un WMR echipat cu RM, [F14].

6.3. Direcţii de cercetare viitoare

Cercetările şi rezultatele obţinute, materializate şi prezentate în această teză de doctorat, mă determină să jalonez câteva direcţii de investigaţie viitoare:

- Acţionarea şi conducerea FMMLs deservite de sisteme robotice, cu luarea în consideraţie a unor concepte de robusteţe la incertitudini parametrice și de model,

atât privitoare la linia de fabricaţie flexibilă, cât şi privitoare la sistemele robotice integrate;
- Implementarea de metode de evitare obstacole pentru sistemele robotice integrate în FMMLs;
- Având la bază implementările pe linii de mecatronică de laborator, se poate face extinderea tehnologiilor hibride de fabricaţie flexibilă deservite de roboţi mobili echipaţi cu manipulatoare la procese industriale reale, în diverse sectoare, din industria auto, metalurgie, industria ceramicii şi a sticlei.

6.4. Diseminarea rezultatelor

Rezultatele obţinute ca urmare a activităţii de cercetare au fost publicate sau sunt acceptate spre publicare, după cum urmează:

- 19 lucrări la conferinţe internaţionale, indexate în următoarele BDI: SCOPUS (19), IEEE Xplore (12), Science direct-IFAC-PapersOnLine (1);
- 10 din cele 19 lucrări sunt indexate ISI proceedings Web of Science (WoS);
- 5 lucrări în reviste, dintre care una în Springer, una indexată ISI Web of Science (factor de impact=0.449) şi 3 indexate în Cambridge Scientific Abstracts.

Mai jos, este prezentată lista cu lucrările prin intermediul cărora au fost diseminate rezultatele.

6.4.1. Lucrări publicate în proceedinguri (indexate ISI, SCOPUS, IFAC-PapersOnLine și/sau BDI) la conferințe internaționale

[F1]. Filipescu, A., Susnea, I., **Filipescu, A.**, **Jr.,** Stamatescu, G., Control of Mobile platforms as Robotic Assistants for Elderly ,Procaeedings of the 7th Asian Control Conference, Hong Kong, China, August 27-29, 2009, IEEE Catalog Number CFP09832, ISBN:978-89-956056-9-1, pp:1456-1461 (indexed WoS).

[F2]. Filipescu, A.**,** Susnea, I., **Filipescu, A., Jr.**, Stamatescu, G., Distributed System of Mobile Platform Obstacle Avoidance and Control as Robotic Assistant for Disabled and Elderly,Proceedings of2009, IEEE International Conference on Control and Automation Christchurch, New Zealand, December 9-11, 2009,IEEE Catalog Number CFP09537,ISBN: 978-1-4244-4707-7, pp: 1886-1891(indexed WoS).

[F3]. Mihalcea I., Radjabov S., **Filipescu, A., Jr.**, IntelligentTrajectory Tracking in Sliding Mode Based Wheeled Mobile Robot Control, Proceeding of 14th IEEE International Conference in Syatem Theory and Control, pp: 556-561, 17-19 Oct., Sinaia, Romania, 2010, ISSN: 2068-0465.

[F4]. Dumitrascu B., Filipescu A., Radaschin A., **Filipescu A. Jr.,** Minca E.,-Discrete-Time Sliding Mode Control of Wheeled MobileRobots, Proceedings of the 8th Asian Control Conference, pp: 771 – 776, Kaohsiung, Taiwan, China, May 16-18, 2011, ISBN: 978-1-61284-487-9(indexed WoS).

[F5]. Filipescu, A., Minzu, V., Dumitrascu, B., **Filipescu, A, Jr.,** Minca, E., Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots, 2011 IEEE International Conference on Information and Automation, pp: 27-32, e-ISBN: 978-1-4577-0269-3, Print ISBN: 978-1-4577-0268-6, DOI: 10.1109/ICINFA.2011.5948958, 6-8 June 2011, Shenzhen, China.

[F6]. Dumitrascu, B., Filipescu, A.**,** Vasilache, C., Minca, E., **Filipescu, A. Jr.,** Discrete-time sliding-mode control of four driving/steering wheels mobile platform, The Proceedings of 19th IEEE Mediterranean Conference on Control & Automation, pp: 1076–1081, ISBN: 978-1-4577-0124-5, DOI: 10.1109/MED.2011.5983167, 20-23 June 2011, Corfu, Greece (indexed WoS).

[F7]. Dumitrascu, B., Filipescu**,** A., Minzu, V., and **Filipescu, A., Jr.,** -Backstepping Control System Theory, Control and Computing, pp: 206-211, 14-16 Oct., Sinaia, Romania, 2011, ISBN: 978-973-621-323-6;

[F8]. Radaschin**, A.,** Filipescu**, A.,** Minzu, V., Minca E., and **Filipescu. A., Jr.,**-Adaptive disassembly sequence control by using mobile robots and system information, Proceeding of 15th IEEE International Conference in System Theory, Control and Computing, pp: 499-505, 14-16 Oct., Sinaia, Romania, 2011, ISBN: 978-973-621-323-6.

[F9]. Filipescu, A., Susnea, I., Minzu, V., Minca, E., Serbencu, A., **Filipescu Jr, A.,-**Path Following Fuzzy Control and Bubble Rebound Obstacle Avoidance Method of a WMR mobile platform, 18th International Conference on Control Systems and Computer Science, proceedings, CSCS 18, 24-27 May, 2011, Bucharest, ed. Politehnica press, ISSN 2066-4451, pp.404-409.

[F10]. Minca, E., Stefan, V., Filipescu, A., Serbencu, A., **Filipescu, A., Jr,** Two Approaches in Modeling of Assembly/Disassembly Line with Integrated Manipulator Mounted on Mobile Platform, International Conference on System Theory, Control and Computing, Joint Conference SINTES 16, SACCS 12, SIMSIS 16, Proceedings of the 16th IEEE, Intrenational Conference on System Theory, Control and Computing, ICSTCC2012 12-14, Oct. Sinaia, 2012, ISBN 978-606-834-848-3, IEEE Catalog Number CFP1236P-CDR.

[F11]. Minca, E., Voda, A., Filipescu, A., and **Filipescu, A., Jr.**: Hybrid Model Based Control of a Mechatronics Line Served by Mobile Robot with Manipulator, In Proceedings of the 8th IEEE International Conference on Industrial and Electronic Application (ICIEA2013), pp: 1296-1301, ISBN: 978-1-4673-6322-8, 19-21, June, 2013, Melbourne, Australia (indexed WoS).

[F12]. Petrea, G., Filipescu, A., Minca, E., Voda, A., **Filipescu, A., Jr.,** Serbencu, A., Hybrid Modelling Based Control of a Processing/Reprocessing Line Served by an Autonomous Robotic System, International Conference on System Theory, Control and Computing, Joint Conference SINTES 16, SACCS 12, SIMSIS 16, Proceedings of the 16th IEEE, International Conference on System Theory, Control and Computing, ICSTCC2013 11-13, Oct. Sinaia, 2013, pp: 410-415, ISBN: 978-1-4799-2228-4/13/$31.00 ©2013 IEEE(indexed WoS).

[F13]. **Filipescu, A., Jr.,** Petrea, G., Filipescu, A., Minca, E., Filipescu, S.,-Discrete modelling based control of a processing/reprocessing mechatronics line served by an autonomous robotic system, The 4th International Symposium on Electrical, and Electronics Engineering, ISEEE 2013, 11-13, Oct, Galati, 2013, ISBN: 978-1-4799-2442-4/13/$31.00 ©2013 IEEE(indexed WoS).

[F14]. **Filipescu, A.**, **Jr.**, Petrea, G., Filipescu, A., Filipescu, S.,-Modeling and Control of a Mechatronics System Served by a Mobile Platform Equipped with Manipulator, Proceedings of the 33rd Chinese Control Conference, July 28-30, 2014, Nanjing, China, pp: 6577-6582, ISBN:978-988-15638-4-2, IEEE Catalog number CFP:1441A-CDR(indexed WoS).

[F15]. Filipescu, A., **Filipescu, A.**, **Jr.**, Simulated Hybrid Model of an Autonomous Robotic System Integrated into Assembly/Disassembly Mechatronics Line Preprints of the 19th World Congress The International Federation of Automatic Control, Cape Town, South

Africa. August 24-29, 2014, pp.9223-9228, Copyright © 2014 IFAC, DOI:10.3182/20140824-6-ZA-1003.00556.

[F16]. Solea, R., Filipescu, A., **Filipescu, A., Jr.,** Minca, E., Filipescu, S., Wheelchair Control and Navigation Based on Kinematic Model and Iris Movement, Proceedings of the 2015 7th IEEE International Conference on Robotics, Automation and Mechatronics (CIS&RAM), 15 – 17 July 2015 Angkor Wat, Cambodia, IEEE Catalog Number: CFP15835-CDR, ISBN: 978-1-4673-7336-4, pp:78-83 (indexed WoS).

[F17]. Filipescu, A., Minca E., Voda A., Dumitrascu B., **Filipescu A., Jr.**, Ciubucciu G., Sliding-Mode Control and Sonnar Based Bubble Rebound Obstacle Avoidance for a WMR, Proceedings of the 19th IEEE, International Conference on System Theory, Control and Computing, ICSTCC 2015 14-16, Oct., Cheile Grădiştei, Romania, 2015, pp.105-110, ISBN: 978-1-4799-8481-7©2015 IEEE (indexed WoS).

[F18]. Ciubucciu, G., Filipescu, A., **Filipescu, A., Jr.,** Filipescu, S., Dumitrascu, B., Control and Obstacle Avoidance of a WMR Based on Sliding-Mode, Ultrasounds and Laser; Proceedings or the 12th IEEE International Conference on Control and Automation, Kathmandu, Nepal, June 1-3, 2016, pp.779-784, ISBN: 978-1-5090-1737-9/16/$31.00 ©2016 IEEE(indexed WoS).

[F19]. Filipescu A., **Filipescu A., Jr.,** Minca, E., Voda, A., Hybrid Modeling,Balancing and Control of a Mechatronics Line Served by Two Mobile Robots, Proceedings of the 20th IEEE, International Conference on System Theory, Control and Computing, ICSTCC 2016, ISBN: 978-1-5090-2720-0/16/$31.00 ©2016 IEEE, pp:234-239, 14-16, Oct., Sinaia, Romania.

6.4.2. Lucrări publicate în reviste

[F20]. Filipescu, A., Minzu, V., **Filipescu, A., Jr.**, Minca E., *"Discrete-Time Sliding-Mode Control of a Mobile Platform with Four Driving/Steering Wheels",* Lecture Notes in Electrical Engineering, Book Title: Advances in Automation and Robotics, Vol.1, Series Volume 122, Series ISSN 1876-1100, Publisher Springer Berlin Heidelberg.

[F21]. Susnea I, Filipescu A.**, Filipescu A., Jr.**, Coman G., Wheeled Mobile Robot Control Using Virtual Pheromones, Petroleum-Gas University of Ploiesti Bulletin, Tehnical Series, Vol LXI, no.3/2009, ISSN 1224-8495, pp. 117-126, cod CNCSIS 38.

[F22]. Filipescu**,** A., Minca, E., **Filipescu, A., Jr.,** Petrea, G., Modeling and Control of Assembly/Disassembly Mechatronic Line Served by Robotic Manipulator Mounted on Mobile Platform The Annals of "Dunărea de Jos" University of Galati Fascicle III, Year 2012: Volume 35, Number 1, Electrotechnics, Electronics, Automatic Control, Informatics, ISSN 1221-454X, pp:17-22.

[F23]. Petrea ,G., Filipescu, A., Minca, E., Voda, A., **Filipescu, A., Jr.,** Hybrid Modelling Based Control of a Processing/Reprocessing Line Served by an Autonomous Robotic System, The Annals of "Dunărea de Jos" University of Galati Fascicle III, Year 2013, Volume 36, Number 1, pp:13-18, Electrotechnics, Electronics, Automatic Control, Informatics, ISSN 1221-454X.

[F24]. Minca, E., Filipescu A., Coanda H., G., Fratila, C., Ion, F., **Filipescu, A., Jr.,** Hybrid Models for Simulation of Mechatronics Lines Served by Parallel or Collaborative Mobile Robots, accepted paper Journal of Control Engineering and Applied Informatics, ISSN 1454-8658, ISI Impact Factor of the journal is 0.449.

Bibliografie

[1]. Alla, H. and David, R., “Continuous and Hybrid Petri Nets”. Journal of Circuits, Systems & Computers 8(1): 159–188, 1998.

[2]. Altekin, F.T., Kindlier, L. and Ozdemirel, N., E., “Profit Oriented Disassembly Line Balancing”, *Journal of Production Research*, 46, pp. 2675-2693, 2008.

[3]. Balduzzi, F., Giua, A., Menga., G., *“First-order hybrid Petri nets: A model for optimization and control”*, IEEE Trans. on Robotics and Automation, 16(4), pp: 382-399, 2000.

[4]. Boothroyd, G.and Althing, L., *Design for assembly and disassembly*, *Annals of CIRP* 41(2), pp: 625–636, 1992.

[5]. Cao T. and Sanderson A.C. (1998). AND/OR Net representation for robotic task sequence planning.IEEE *Trans. Syst. Man and Cybernetics-Part C: Application and Review*. Vol. 28, 104-218.

[6]. Chen, F., F., and Adams, E., E., *“The Impact of Flexible Manufacturing Systems on Productivity and Quality"*, IEEE Transactions of Engineering Management, vol. 38, pp: 33-45, 1991.

[7]. Choi, C. K., Zhang, X., F., Ng, T. L. and Lau, W. S. (1998). On the generation of product assembly sequences, *Intl. Journal of Production Research*, pp: 617-633.

[8]. Chryssolouris G.*Manufacturing Systems – Theory and Practice.* New York, NY: Springer Verlag, 2005. 2-nd edition.

[9]. Chu-qing C., Lian-zheng G.and Rui-feng L., Mobile robots target tracking using finite-time convergence sliding mode controller, *2010 8th IEEE International Conference on Control and Automation*, Xiamen, China, June 9-11, pp: 460-464, 2010.

[10]. Chwa, D., Seo J. H., Kim P., and Choi J. Y., Sliding mode tracking control of nonholonomic wheeled mobile robots, *Proceedings of the American Control Conference,* Anchorage, pp: 3991-3996, May, 2002.

[11]. Chwa D., Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates, *IEEE Transactions on Control Systems Technology*, vol. 12, nr. 4, iulie, pag.637-644, 2004.

[12]. Ciubucciu, G., Adrian Filipescu, A., Filipescu, A., Jr., Filipescu, S., Dumitrascu, B., Control and Obstacle Avoidance of a WMR Based on Sliding-Mode, Ultrasounds and Laser; Proceedings or the 12th IEEE International Conference on Control and Automation, Kathmandu, Nepal, June 1-3, 2016, pp.779-784, ISBN: 978-1-5090-1737-9/16/$31.00 ©2016 IEEE.

[13]. Corradini, M. L., Leo, T., Orlando G., Variable structure control of robotic asistance system, *Proceedings of the 5th IEEE Mediterranean Conference on Control and Systems*, http://med.ee.nd.edu/MED5/PAPERS/052/052.PDF.

[14]. Danwei, W. Feng, Qi, Trajectory planning for a four-wheel steering vehicle, *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, pag.3320-3325, 2001.

[15]. David,R., Alla, H., *Discrete, Continuous and Hybrid Petri Nets*, ISBN 978-3-642-10668-2, Springer Verlag, Berlin Heidelberg, 2010.

[16]. Dolgui A, Guschinsky N, Levin G.*"A special case of transfer lines balancing by graph approach"*, European Journal of Operational Research 2006, 168: 732-746.

[17]. Dumitrascu, B., Filipescu, A., Radaschin, A., **Filipescu, A. Jr.,** Minca E.,-Discrete-Time Sliding Mode Control of Wheeled MobileRobots, Proceedings of the 8th Asian Control Conference, pp: 771 – 776, Kaohsiung, Taiwan, China, May 16-18, 2011, ISBN: 978-1-61284-487-9.

[18]. Dumitrascu, B., Filipescu, A**.,** Vasilache, C., Minca, E., **Filipescu, A. Jr.,** Discrete-time sliding-mode control of four driving/steering wheels mobile platform, The Proceedings of 19th IEEE Mediterranean Conference on Control & Automation, pp: 1076–1081, ISBN: 978-1-4577-0124-5, Digital Object Identifier: 10.1109/MED.2011.5983167, 20-23 June 2011, Corfu, Greece.

[19]. Dumitrascu, B., Filipescu**,** A., Minzu V. and **Filipescu, A. Jr.;** -Backstepping Control of Wheeled Mobile Robots, Proceeding of 15th IEEE International Conference in System Theory, Control and Computing, pp: 206-211, 14-16 Oct., Sinaia, Romania, 2011, ISBN: 978-973-621-323-6.

[20]. Dumitrascu B., Contributii la conducerea, navigația și evitarea de obstacole a roboților mobili și vehiculelor autonome, PhD Thesis, "Dunarea de Jos" University of Galati, Galati, 2012.

[21]. Feddema, J. T. and Simon, R. W., *"Visual servoing and CAD-driven microassembly",* IEEE Robotics Automat*.* Mag.5(4), pp:18–24, 1998.

[22]. Feng S. and Song E., (2008). A manufacturing process information model for design and process planning *Journal of Manufac. System*, Vol. 22, 1-16.

[23]. Filipescu, A., Susnea, I., **Filipescu, A.**, **Jr.,** Stamatescu, G., Control of Mobile platforms as Robotic Assistants for Elderly ,Proceedings of the 7th Asian Control Conference, Hong Kong, China, August 27-29, 2009, IEEE Catalog Number CFP09832, ISBN:978-89-956056-9-1, pp:1456-1461.

[24]. Filipescu, A**.,** Susnea, I., **Filipescu, A., Jr.**, Stamatescu, G., Distributed System of Mobile Platform Obstacle Avoidance and Control as Robotic Assistant for Disabled and Elderly,Proceedings of2009, IEEE International Conference on Control and Automation Christchurch, New Zealand, December 9-11, 2009,IEEE Catalog Number CFP09537,ISBN: 978-1-4244-4707-7, pp: 1886-1891, Library of Congress:2009904841.

[25]. Filipescu, A., Susnea, I., Minzu, V., Minca, E., Serbencu, A., **Filipescu, A., Jr., -**Path Following Fuzzy Control and Bubble Rebound Obstacle Avoidance Method of a WMR mobile platform, 18th International Conference on Control Systems and Computer Science, proceedings, CSCS 18, 24-27 May, 2011, Bucharest, ed. Politehnica press, ISSN 2066-4451, pp:404-409.

[26]. Filipescu, A., Minzu, V., Dumitrascu, B., **Filipescu, A, Jr.,** Minca, E., Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots, 2011 IEEE International Conference on Information and Automation, pp: 27-32, e-ISBN : 978-1-4577-0269-3, Print ISBN: 978-1-4577-0268-6, DOI : 10.1109/ICINFA.2011.5948958, 6-8 June 2011, Shenzhen, China.

[27]. Filipescu, A., Minzu, V., **Filipescu, A., Jr.**, Minca E., *"Discrete-Time Sliding-Mode Control of a Mobile Platform with Four Driving/Steering Wheels",* Lecture Notes in Electrical Engineering Book Title: Advances in Automation and Robotics, Vol.1 , Series Volume 122, Series ISSN 1876-1100, Publisher Springer Berlin Heidelberg.

[28]. Filipescu**,** A., Minca, E., **Filipescu, A., Jr.,** Petrea, G., Modeling and Control of Assembly/Disassembly Mechatronic Line Served by Robotic Manipulator Mounted on Mobile Platform, The Annals of "Dunărea De Jos" University of Galati Fascicle III, Year

2012: Volume 35, Number 1electrotechnics, Electronics, Automatic Control, Informatics, ISSN 1221-454X, pp:17-22.

[29]. **Filipescu, A., Jr.,** Petrea, G., Filipescu, A., Minca, E., Filipescu S.,-Discrete modelling based control of a processing/reprocessing mechatronics line served by an autonomous robotic system, The 4th International Symposium on Electrical, and Electronics Engineering, ISEEE 2013, 11-13, Oct, Galati, 2013, ISBN: 978-1-4799-2442-4/13/$31.00 ©2013 IEEE.

[30]. **Filipescu, A., Jr.**, Petrea, G., Filipescu, A.,, Filipescu, S.,-Modeling and Control of a Mechatronics System Served by a Mobile Platform Equipped with Manipulator, Proceedings of the 33rd Chinese Control Conference, July 28-30, 2014, Nanjing, China, pp: 6577-6582, ISBN:978-988-15638-4-2, IEEE Catalog number CFP:1441A-CDR.

[31]. Filipescu, A., **Filipescu, A.**, **Jr.**, Simulated Hybrid Model of an Autonomous Robotic System Integrated into Assembly/Disassembly Mechatronics Line Preprints of the 19th World Congress The International Federation of Automatic Control, Cape Town, South Africa. August 24-29, 2014, pp: 9223-9228, Copyright © 2014 IFAC, DOI: 10.3182/20140824-6-ZA-1003.00556.

[32]. Filipescu, A., Minca, E., Voda, A., Dumitrascu B., **Filipescu A., Jr.,** Ciubucciu G., Sliding-Mode Control and Sonnar Based Bubble Rebound Obstacle Avoidance for a WMR, Proceedings of the 19th IEEE, International Conference on System Theory, Control and Computing, ICSTCC 2015 14-16, Oct. Cheile Gradistei, Romania, 2015, pp: 105-110, ISBN: 978-1-4799-8481-7©2015 IEEE.

[33]. Fukao T., Nakagawa H., Adachi N., Adaptive tracking control of a nonholonomic mobile robot, *IEEE Transactions on Robotics and Automation*, vol. 16, nr.5, pp:. 609-615, 2000.

[34]. Galitsky, C., and Worrell, E., *"Energy Efficiency Improvement and Cost Saving Opportunities for the Vehicle Assembly Industry",* Lawrence Berkeley National Laboratory (LBNL-50939-Revision), 2008.

[35]. Ganget, J., Hattenberger, G., Alami, R. (2005). Task planning and control for multi-UAV system: architecture and algorithms, *IEEE Int. Conf. On Intelligent Robot and System*, Vol.18, pp: 758-768.

[36]. Gao, W., Hung, J., C., Variable structure control of nonlinear systems: A new approach, *IEEE Transactions on Industrial Electronics*, 40(1), pp: 45 - 55, 1993.

[37]. Gao W., Wang Y. and Homaifa A., Discrete-time variable structure control systems, *IEEE Transaction on Industrial Electronics*, vol. 42, pp: 117-122, 1995.

[38]. Garcia, M., A., P., Montiel, O., Castillo, O., Sepulveda, R., Melin, P., Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, *Applied Soft Computing*, vol.9, pp: 1102-1110, 2009.

[39]. Gasparetto A. and Zanotto V., (2007). A new method for smooth trajectory planning of robot manipulators. *Mechanism and Machine Theory*, Vol. 42, pp: 455-471.

[40]. Ge S. S., Lai X.-C., Mamun A. Al., Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints, *Robotics and Autonomous Systems*, vol. 57, pp: 513-526, 2007.

[41]. Ghomri L. and Alla H., (2008*), Modeling and Analysis using Hybrid Petri Nets*, ISBN 978-3-902613-12-7, In Tech.

[42]. Gungor A., and Gupta, S.M., (2002), Disassembly line in product recovery, *International Journal of Production Research*, 40, 2002, pp: 2569-2589.

[43]. Hung J. Y., Gao, W., Hung J. C., Variable structure control: A survey, *IEEE Transactions on Industrial Electronics*, vol. 40(1), pp: 2-22, 1993.

[44]. Inman, R., R., Blumenfeld, D., E., Huang, N., Li, J., *"Production System Design for Quality: Research Opportunities from Automotive Industry Prospective"*, International Journal of Production Research, vol. 41, pp:1953-1971, 2003.
[45]. Jiang, J., P., Nijmeijer, H., Tracking control of mobile robots: a case study in backstepping, *Automatica*, vol. 33, nr. 7, pp: 1393-1399, 1997.
[46]. Kallrath J., Planning and scheduling in the process industry. Springer-Verlag, *Advance Planning and Scheduling Solution in Process Industry*, pp: 201-227, 2003.
[47]. Kanarat A., Motion Planning and robust control for nonholonomic mobile robots under uncertainties, PhD Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2004.
[48]. Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi T., A stable tracking control scheme for an autonomous mobile robot, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp: 384-389, 1990.
[49]. Khalil H.K., Nonlinear Systems, Second Edition, Prentice Hall, New Jersey, 1996.
[50]. Khatib O., Real-time obstacle avoidance for manipulators and mobile robots, *IEEE International Conference on Robotics and Automation*, pp: 500-505, March, 1985.
[51]. Kopacek, B., and Kopacek,*P., "Intelligent Disassembly of Electronic Equipment*", in Proceedings of the 1st IFAC Workshop on „Intelligent Assembly and Disassembly – *IAD'98"*, pp: 87-92, Oxford, UK, 1998.
[52]. Kopacek, B., and Kopacek, P., *"Intelligent Disassembly of Electronic Products"*, in Proceedings of the 2nd Symposium „ Eco-Efficient Concepts for the Electronics Industry Towards Sustainability - CARE INNOVATION`98", pp: 130-136, Vienna, Austria, 1998.
[53]. Kumar U. and Sukavanam N., Backstepping based trajectory tracking control of a four wheeled mobile robot, *International Journal of Advanced Robotic Systems*, Vol.5, No.4, pp: 403-410, 2008.
[54]. LabVIEW User Manual, 2011. http://www.ni.com/pdf/manuals/320999e.pdf.
[55]. Lee H., Chattering suppression in sliding mode control system, PhD Thesis M. S. Ohio State University, 2007.
[56]. Lee J. H, Lin C., Lim H. and Jang Myung Lee, Sliding mode control of mobile robot in the RFID sensor space, *International Journal of Control, Automation and Systems,* pp: 429-435, 2009.
[57]. Li, J., and Huang, N., *"Quality Evaluation in Flexible Manufacturing Systems: A Markovian Approach"*, Mathematical Problems in Engineering, vol. 2007, article ID 57128, 2007.
[58]. Lumelsky V., Skewis T., Incorporating range sensing in the robot navigation function, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no: 5, pp: 1058-1068, 1990.
[59]. Manalov O. B., Adaptive steering control for autonomous mobile robots, *Proceedings of the 10th Mediterranean Conference on Control and Automation- MED2002,* Portugal, 2002.
[60]. Menegatti E., Maeda T., Ishiguro H., Image-based memory for a robot navigation using properties of omnidirectional images, *Robotics and Autonomous Systems*, vol. 47, pp: 251-267, 2004.
[61]. Minca, E., Stefan, V., Filipescu, A., Serbencu, **A., Filipescu, A., Jr,** Two Approaches in Modeling of Assembly/Disassembly Line with Integrated Manipulator Mounted on Mobile Platform, Proceedings of the 16th IEEE, International Conference on System Theory, Control and Computing, ICSTCC2012 12-14, Oct. Sinaia, 2012, ISBN 978-606-834-848-3, IEEE Catalog Number CFP1236P-CDR.
[62]. Minca, E., Voda, A., Filipescu, A., and **Filipescu, A., Jr.**, Hybrid Model Based Control of a Mechatronics Line Served by Mobile Robot with Manipulator, In Proceedings

of the 8th IEEE International Conference on Industrial and Electronic Application (ICIEA2013), pp: 1296-1301, ISBN: 978-1-4673-6322-8, 19-21, June, 2013, Melbourne, Australia.

[63]. Minca, E., Filipescu, A., Voda, A., Modelling and control of an assembly/disassembly mechatronics line, served by mobile robot with manipulator, Control Engineering Practice vol. 31 (2014) pp. 50–62, ISSN: 0967-0661, DOI: 10.1016/j.conengprac.2014.06.005.

[64]. Mnif F., Recursive backstepping stabilization of a wheeled mobile robot*, International Journal of Advanced Robotic Systems*, vol. 1 no. 4, pp: 287–294, 2004.

[65]. Momcilovic O. I., Discrete-time variable structure controller synthesis for third order velocity measurement from low-cost optical encoders, *IMTC 2008- IEEE Instrumentation and Measurement Technology Conference*, Victoria, Canada, May, 2008.

[66]. Nof, S. Y., Wilhelm, W. E., and Warnecke, H. J., *Industrial Assembly*, Chapman & Hall, London, 1997.

[67]. Petrea, G., Filipescu, A., Minca, E., Voda, A., **Filipescu, A., Jr.,** Serbencu, A., Hybrid modelling based control of a processing/reprocessing mechatronics line served by an autonomous robotic system, Proceedings of the 16th IEEE, Intrenational Conference on System Theory, Control and Computing, ICSTCC2013 11-13, Oct. Sinaia, 2013, pp: 410-415, ISBN: 978-1-4799-2228-4/13/$31.00 ©2013 IEEE.

[68]. Petrea ,G., Filipescu, A., Minca, E., Voda, A., **Filipescu , A., Jr.,** Hybrid Modelling Based Control of a Processing/Reprocessing Line Served by an Autonomous Robotic System, The Annals of "Dunărea de Jos" University of Galati Fascicle III, Year 2013, Volume 36, Number 1, pp:13-18, Electrotechnics, Electronics, Automatic Control, Informatics, ISSN 1221-454X.

[69]. Peng, S. S., and Zhou, M. C., (2003), Sensor-Based Stage Petri Net Modeling of PLC Logic Programs for Discrete-Event Control Design, *International Journal of Production Research*, 41(3), pp: 629-644, Feb., 2003.

[70]. Petrella R., Tursini M., Peretti L., Zigliotto M., Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative analysis, *International Aegean Conference on Electrical Machines and Power Electronics*, Bodrum, Turkey, Sep., 2007.

[71]. Radaschin**, A.,** Filipescu**,** A., Minzu, V., Minca, E., and **Filipescu, A., Jr.,** Adaptive disassembly sequence control by using mobile robots and system information, Proceeding of 15th IEEE International Conference in System Theory, Control and Computing, pp: 499-505, 14-16 Oct., Sinaia, Romania, 2011, ISBN: 978-973-621-323-6;

[72]. Radaschin, A., Voda, A., Minca E., and Filipescu, A., (2012) Task Planning Algorithm in Hybrid Assembly/Disassembly Process, In *Proceeding of 14th IFAC Symposium on Information Control Problems in Manufacturing*, May 23-25, 2012, Bucharest, ISSN: 1474-6670; ISBN: 978-3-902661-98-2, pp: 571-576.

[73]. Radaschin A., Contribuții la conducerea inteligentă a roboților mobili utilizați în liniile flexibile de fabricație, PhD Thesis, "Dunarea de Jos" University of Galati, Galati, 2012.

[74]. Rubio J. De Jesu´s and Yu W., A new discrete-time sliding-mode control with time-varying gain and neural identification, *International Journal of Control*, vol. 79, no: 4, pp. 338–348, April, 2006.

[75]. Seliger, G.Grudzien, W. and Zaidi, H. (1999), New methods of product data provision for a simplified disassembly, *In Proceedings of the 6th International Seminar on Life Cycle Engineering*, Kingston, Canada, June 21–23, 1999. pp: 250–259.

[76]. Solea R., Filipeascu, A., Stamatescu G., Trajectory-tracking sliding-mode control for wheeled mobile robots, Energy, *Transport and Envinronment Control Applications-International Workshop*, pp: 121-134, May, 2009.

[77]. Solea, R., Filipescu, A. and Stamatescu, G., Sliding-mode real-time mobile platform control in the presence of uncertainties, *Proceedings of the 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, Shanghai, pp: 7747-7752, 2009.

[78]. Solea R., Filipescu A., Nunes U., Sliding-mode control for trajectory-tracking of a wheeled mobile robot in presence of uncertainties, *Proceedings of the 7th Asian Control Conference*, Hong Kong, pp: 1701-1706, 2009.

[79]. Solea R., Sliding mode control applied in trajectory-tracking of WMRs and autonomous vehicles, PhD Thesis, University of Coimbra, Portugal, 2009.

[80]. Solea, R., Filipescu, A., **Filipescu, A., Jr.,** Minca, E., Filipescu, S., Wheelchair Control and Navigation Based on Kinematic Model and Iris Movement, Proceedings of the 2015 7th IEEE International Conference on Robotics, Automation and Mechatronics (CIS&RAM), 15 – 17 July 2015 Angkor Wat, Cambodia, IEEE Catalog Number: CFP15835-CDR, ISBN: 978-1-4673-7336-4, pp:78-83.

[81]. Susnea, I., Filipescu, A., **Filipescu, A., Jr.,** Coman G., Wheeled Mobile Robot Control Using Virtual Pheromones, Petroleum-Gas University of Ploiesti Bulletin, Technical Series, Vol LXI, no.3/2009, ISSN 1224-8495, pp: 117-126, cod CNCSIS 38.

[82]. Susnea, I., Contribuţii la elaborarea unor soluţii bazate pe structuri încorporate (embedded) pentru conducerea in timp real a sistemelor robotice, PhD Thesis, "Dunarea de Jos" University of Galati, Galati, 2010.

[83]. Tolio D., *Design of Flexible Production Systems – Methodologies and Tools*. Berlin: Springer, 2009. ISBN 978-3-540-85413-5.

[84]. Utkin V.I., Guldner J., and Shi J., Sliding mode control in electromechanical systems, *Taylor & Francis*, London, 1999.

[85]. Utkin V.I., Sliding modes in optimization and control, *Springer-Verlag*, New York, 1992.

[86]. Velagic J., Lacevic B., Perunicic B., A 3-level autonomous mobile robot navigation system designed by using reasoning/ search approaches, *Robotics and Systems*, vol. 54, pp: 989-1004, 2006.

[87]. Vivekanandan C., Prabhakar R., A redefined discrete quasi-sliding mode strategy, *International Journal of Recent Trends*, vol. 1, no. 3, pp: 92-96, 2009.

[88]. Vivekanandan V., Prabhakar R. and Gnanambigai M., A redefined quasi-sliding mode control strategy, *World Academy of Science, Engineering and Technology*, 39, pp: 292-295, 2008.

[89]. Xiao L., Su H., Zhang X., Chu J., A new discrete variable structure control algorithm based on sliding mode prediction, *2005 American Control Conference*, Portland, USA, pp: 4643-4648, 2005.

[90]. Yuan J., Tang G.-Y., Finite-time tracking control algorithms based on variable structure for mobile robots, *Proceedings of the 29th Chinese Control Conference*, Beijing, China, pag.419-423, June, 2010.

[91]. Yuan Z. P., Wang Z. P. and Chen Q. J., Trajectory tracking control of a nonholonomic mobile robot, *2010 8th IEEE International Conference on Control and Automation* , Xiamen, China, pag.2207-2211, June, 2010.

[92]. Zhou1 B., Han J., Dai X., Backstepping Based Global Exponential Stabilization of a Tracked Mobile Robot with Slipping Perturbation, *Journal of Bionic Engineering,* vol. 8, pp: 69–76, 2011.

[93]. Zussman, E. M. and Seliger, G., "Production remanufacturing", in: S. Y. Nof (ed.), *Handbook of Industrial Robotics*, 2nd edn, Wiley, New York, Chapter 54, pp: 1037–1044, 1999.

[94]. http://www.mobilerobots.com/software/mobilesim.aspx.
[95]. MobileRobots Advanced Robotics Interface for Applications (ARIA) Developer's Reference Manual, www.mobilerobots.com.
[96]. www.mathworks.inc.
[97]. http://www.lag.ensieg.inpg.fr/sirphyco.
[98]. https://www.visualstudio.com/vs/cplusplus.

Anexa 1

Program C++ pentru conducera sliding-mode a robotului mobil Pioneer 3-DX şi conducerea în bucla deschisă a manipulatorului robotic Pioneer 5-DOF ARM

```
#include "Aria.h"
#include <stdio.h>
#include <math.h>
#include <conio.h>
//#include "Velocity.h"
static double angleLimit(double angle)
{
  if (angle > 3.14)
    return(angle - 2*3.14);
  else
    if (angle < -3.14)
      return(angle + 2*3.14);
   else
  return(angle);
}
/* ------------- SIGN & SATURATION FUNCTION ------------- */
double sign(double value){
  if (value<0)
    return(-1.00);
  else
    if (value>0)
    return(1.00);
      else
 return(1.00);
}
double satur(double value){
  if (value<-1)
    return (-1.00);
  else if (value > 1)
   return (1.00);
  else
   return (value);
}
int main(int argc, char **argv)
{
  Aria::init();
  ArSimpleConnector con(&argc, argv);
```

```
  ArRobot robot;
  ArP2Arm arm;
ArKeyHandler keyHandler;
Aria::setKeyHandler(&keyHandler);
printf("You may press escape to exit\n");
  if(!Aria::parseArgs())
  {
    Aria::logOptions();
    Aria::shutdown();
    return 1;
  }
  ArLog::log(ArLog::Normal, "armExample: Connecting to the robot.");
  if(!con.connectRobot(&robot))
  {
    ArLog::log(ArLog::Terse, "armExample: Could not connect to the robot. Exiting.");
    Aria::shutdown();
    return 1;
  }
  ArSonarDevice sonar;
  robot.addRangeDevice(&sonar);
  robot.runAsync(true);
  // turn off sonar
 // robot.comInt(28, 0);
  robot.enableMotors();
  // Collision avoidance actions at higher priority
  ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
  //ArActionLimiterForwards limiterFarAction("speed limiter far", 300, 1100, 400);
  ArActionLimiterForwards limiterFarAction("speed limiter far", 600, 2200, 800);
  ArActionLimiterTableSensor tableLimiterAction;
  robot.addAction(&tableLimiterAction, 100);
  robot.addAction(&limiterAction, 95);
  robot.addAction(&limiterFarAction, 90);
  // Set up and initialize the arm
  arm.setRobot(&robot);
  if (arm.init() != ArP2Arm::SUCCESS)
  {
    ArLog::log(ArLog::Terse, "armExample: Error initializing the P2 Arm!");
    return 1;
  }
  // Print out some of the settings
  P2ArmJoint *joint;
  printf("Current joint info:\nJoint   Vel  Home  Center\n");
  for (int i=1; i<=ArP2Arm::NumJoints; i++)
  {
    joint = arm.getJoint(i);
    printf("  %2i:  %5i %5i   %5i\n", i, joint->myVel, joint->myHome, joint->myCenter);
  }
  printf("\n");
  // Put the arm to work
```

```
  printf("Powering on  (takes a couple seconds to stabilize)\n");
  arm.powerOn();
  // Request one status packet and print out the arm's status
  printf("Current arm status:\n");
  arm.requestStatus(ArP2Arm::StatusSingle);
  ArUtil::sleep(200);  // Give time to get the packet
  printf("Arm Status: ");
  if (arm.getStatus() & ArP2Arm::ArmGood)
    printf("Good=1 ");
  else
    printf("Good=0 ");
  if (arm.getStatus() & ArP2Arm::ArmInited)
    printf("Inited=1 ");
  else
    printf("Inited=0 ");
  if (arm.getStatus() & ArP2Arm::ArmPower)
    printf("Power=1 ");
  else
    printf("Power=0 ");
  if (arm.getStatus() & ArP2Arm::ArmHoming)
    printf("Homing=1 ");
  else
    printf("Homing=0 ");
  printf("\n\n");
  double u1,u2,u3;
       // u1,u2,u3 ughiuile articulatiilor 1,2,3
       double c1,c2,c3,s1,s2,s3;

       double a1,a2,a3;
       /*double px[]={ 0, 50, 88};
       double py[]={ 0, 8,  100};
       double pz[]={ 0, 30,  60};
       FILE *date1;
  double v, av, w, aw;                  /* velocity to drive left wheel*/
      //
       /*/
  int i;
int lungime=3;
double px=0;
       double py=8;
       double pz=1;
a1=0;a2=15,a3=15;
  /*for( i=1;i<lungime;i++){

       c3=(px[i]*px[i]+py[i]*py[i]+ pz[i]*pz[i]-a2*a2-a3*a3)/(2*a2*a3);
      s3=sqrt(1-c3*c3);
       s2=((a2+a3*c3)*pz[i]-a3*s3*sqrt(px[i]*px[i]+py[i]*py[i]))/(px[i]*px[i]+py[i]*py[i]+pz[i]*pz[i]);
       c2=((a2+a3*c3)*sqrt(px[i]*px[i]+py[i]*py[i])+a3*s3*pz[i])/(pz[i]*pz[i] +py[i]*py[i]+pz[i]*pz[i]);
       u1=atan2(py[i],px[i]);
```

```
      u2=atan2(s2,c2);
     u3=atan2(s3,c3);
   arm.moveTo(1,u1,40);
  arm.moveTo(3,u2,40);
  arm.moveTo(6,u3,40);
  ArUtil::sleep(9000);
  } */
/*c3=(px*px+py*py+ pz*pz-a2*a2-a3*a3)/(2*a2*a3);
     s3=sqrt(1-c3*c3);
      s2=((a2+a3*c3)*pz-a3*s3*sqrt(px*px+py*py))/(px*px+py*py+pz*pz);
      c2=((a2+a3*c3)*sqrt(px*px+py*py)+a3*s3*pz)/(pz*pz +py*py+pz*pz);
      u1=atan2(py,px);
      u2=atan2(s2,c2);
     u3=atan2(s3,c3);
     */
  arm.moveTo(1,75,20);
  arm.moveTo(2,-40,30);
  arm.moveTo(3,-70,30);
  arm.moveTo(5,10,30);
  ArUtil::sleep(7000);
  arm.moveTo(6,10,20);
  ArUtil::sleep(5000);
  //arm.moveTo(1,75,20);
  arm.moveTo(2,35,20);
  arm.moveTo(3,70,20);
  arm.moveTo(5,10,20);
   arm.moveTo(6,10,220);ArUtil::sleep(5000);
//
  //
   //-----------------------------------
  // double lungime=10;
  FILE *date1;
  double v, av, w, aw;                  /* velocity to drive left wheel*/
 double theta_des_next;        /* theta desired */
  double ref_xr, ref_yr, ref_Th, ref_w, ref_v, x_des_next, y_des_next;  /* desired position */
  double x_e, y_e, theta_e, x_e_der, y_e_der, theta_e_der; /* errors */
  double s_1, s_2, Q1, Q2, P1, P2, gama0, gama_x, gama_y;
  double v_c, v_c_der, w_c, temp1, temp2;
  double Ts, v_l, v_r, timp, alpha;
  int  n;
//   date1=fopen("c:\Test\test_01_erori.txt","w+");
  timp=0.0;
  Ts=0.1;
  Q1 = 0.05;          //0.05        0.5
  Q2 = 0.5;           //0.5         0.75
  P1 = 0.5;       //0.5          1.75
  P2 = 0.75;          //1.75        0.75
  alpha = 0.9;
  // gama0 = 15;    //15, 10,5 100
```

```
  // gama_x = 0.95;     //1.75        0.75
  // gama_y = 3.75;          //10          15
//  gama0 = 30;     //30
  gama_x = 0.75;  //0.75 //pt. cazul III gama_x=5.75
  gama_y = 15;          //15
  ref_xr = 0; //2
  ref_yr = 0; //1
  ref_Th = 0;
  ref_v = 0.00;
  ref_w = 0.00;
  gama0 = gama_y/(3.14/2);
     n = 300;
     ArUtil::sleep(1000);
     robot.lock();
     robot.setVel(0);
     robot.unlock();
     ArUtil::sleep(100);
  for (i=1; i<200; i++){
robot.lock();
     /*   v = velocity[i][0];
          av = velocity[i][1];
          w = velocity[i][2];
          aw = velocity[i][3];
          */
          v=0.27;
          av=0;
          w=0.3;
          aw=0;
          theta_des_next = angleLimit(Ts*w+ref_Th);
          x_des_next = Ts*v*cos(ref_Th) + ref_xr;
          y_des_next = Ts*v*sin(ref_Th) + ref_yr;
          x_e = ((robot.getX()/1000)-ref_xr)*cos(ref_Th)+((robot.getY()/1000)-
ref_yr)*sin(ref_Th);
          y_e = -((robot.getX()/1000)-ref_xr)*sin(ref_Th)+((robot.getY()/1000)-
ref_yr)*cos(ref_Th);
          theta_e = angleLimit((robot.getTh()*(3.14/180))-ref_Th);
     if (y_e < 0)
          {
             gama_y = -abs(gama_y);
          }
           else if( y_e > 0 )
          {
             gama_y = abs( gama_y );
          }
          x_e_der = -v + (robot.getVel()/1000)*cos(theta_e)+y_e*w;
          y_e_der = (robot.getVel()/1000)*sin(theta_e)-x_e*w;
          theta_e_der = (robot.getRotVel()*3.14/180)-w;
          /* SLIDING SURFACE */
           s_1 = x_e_der + gama_x*x_e;
```

```
          s_2 = y_e_der + gama_y*y_e + gama0*sign(y_e)*(theta_e);
         // CAZUL I (A)
          temp1 = -Q1*satur(s_1/0.5) - gama_x*x_e_der - (aw*y_e) - w*y_e_der + av;
          temp2 = -Q2*satur(s_2/0.5) - gama_y*y_e_der + (aw*x_e) + w*x_e_der;
         // CAZUL II (B)
         // temp1 = -Q1*satur(s_1/0.4) - (P1*s_1) - gama_x*x_e_der - (aw*y_e) - w*y_e_der +
av;
         // temp2 = -Q2*satur(s_2/0.4) - (P2*s_2) - gama_y*y_e_der + (aw*x_e) + w*x_e_der;
         // CAZUL III (C)
         // temp1 = -Q1*pow(abs(s_1),alpha)*satur(s_1/0.5) - gama_x*x_e_der - (aw*y_e) -
w*y_e_der + av;
         // temp2 = -Q2*pow(abs(s_2),alpha)*satur(s_2/0.5) - gama_y*y_e_der + (aw*x_e) +
w*x_e_der;
         // CAZUL IV (D)
         // temp1 = -Q1*exp(alpha*abs(s_1))*satur(s_1/0.5) - gama_x*x_e_der - (aw*y_e) -
w*y_e_der + av;
         // temp2 = -Q2*exp(alpha*abs(s_2))*satur(s_2/0.5) - gama_y*y_e_der + (aw*x_e) +
w*x_e_der;
         v_c_der = (temp1 + (robot.getVel()/1000)*theta_e_der*sin(theta_e))/(cos(theta_e));
         v_c = (Ts*v_c_der) + ref_v;
         w_c = (temp2-
v_c_der*sin(theta_e))/((robot.getVel()/1000)*cos(theta_e)+gama0*sign(y_e))+w;
         // ROBOT
              v_l = v_c + 0.19*w_c; //0.19 //0.28 //0.24
              v_r = v_c - 0.19*w_c;
//            v_l = v + 0.19*w;   //0.19 //0.28 //0.24
//            v_r = v - 0.19*w;
              // Simulator
              // v_r = v_c + 0.19*w_c;
              // v_l = v_c - 0.19*w_c;
       // setVel2 (double leftVelocity, double rightVelocity)
              robot.setVel2(1000*v_r, 1000*v_l);
     //    robot.setRotVel(w);
     //    robot.setVel(100*v);
         //printf("\t %4.5f \t %4.5f \t %4.5f %4.5f \t %4.5f \n", robot.getX()/1000,
robot.getY()/1000, robot.getTh()*(3.14/180), v_r, v_l);
         // printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", robot.getVel()/1000,
robot.getRotVel()*3.14/180, v_c, w_c);
     //fprintf(date1," %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t
%4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \t
%4.5f \n", robot.getX()/1000, robot.getY()/1000, robot.getTh()*(3.14/180), ref_xr, ref_yr,
ref_Th, x_e, y_e, theta_e, x_e_der, y_e_der, theta_e_der, (robot.getVel()/1000),
(robot.getRotVel()*3.14/180), v, w, v_c, w_c, s_1, s_2, timp);
         printf("%4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", x_e, y_e, theta_e, v, w);
         //printf("\t %4.5f \t %4.5f \t %4.5f \t %4.5f \t %4.5f \n", ref_xr, ref_yr, ref_Th*(180/3.14),
robot.getRobotDiagonal()/1000, robot.getRobotRadius()/1000);
         ref_xr = x_des_next;
         ref_yr = y_des_next;
         ref_Th = theta_des_next;
```

```
            ref_w = w_c;
            ref_v = v_c;
            timp=timp+Ts;
            robot.unlock();
            ArUtil::sleep(100);
  }
 robot.lock();
  robot.setRotVel(0);
  robot.setVel(0);
  robot.unlock();
  ArUtil::sleep(3000);
  arm.moveTo(2,-35,30);
  arm.moveTo(3,-70,30);
  arm.moveTo(5,-10,30);
  ArUtil::sleep(7000);
  arm.moveTo(6,-90,20);
  ArUtil::sleep(5000);
  for (int i=1; i<=ArP2Arm::NumJoints; i++)
  {
    joint = arm.getJoint(i);
     printf("[%d] %.0f, ", i, arm.getJointPos(i));
  }
  printf("\n");
/*  px=4;py=6;pz=3;
  c3=(px*px+py*py+ pz*pz-a2*a2-a3*a3)/(2*a2*a3);
      s3=sqrt(1-c3*c3);
        s2=((a2+a3*c3)*pz-a3*s3*sqrt(px*px+py*py))/(px*px+py*py+pz*pz);
        c2=((a2+a3*c3)*sqrt(px*px+py*py)+a3*s3*pz)/(pz*pz +py*py+pz*pz);
        u1=atan2(py,px);
        u2=atan2(s2,c2);
      u3=atan2(s3,c3);
  arm.moveTo(2,50,40);
  arm.moveTo(3,45,40);
  ArUtil::sleep(9000);
 for (int i=1; i<=ArP2Arm::NumJoints; i++)
  {
    joint = arm.getJoint(i);
     printf("[%d] %.0f, ", i, arm.getJointPos(i));
  }
  printf("\n");*/
  // Move the arm joints to specific positions
/*  printf("Moving Arm...\n");
  int deploy_offset[] = {0, 0, 50, 0, 0, 0, 0};
  for (int i=1; i<=ArP2Arm::NumJoints; i++)
  {
    joint = arm.getJoint(i);
    arm.moveToTicks(i, joint->myCenter + deploy_offset[i]);
  }
  // Wait for arm to achieve new position, printing joint positions and M for
```

```
  // moving, NM for not moving.
  arm.requestStatus(ArP2Arm::StatusContinuous);
  ArUtil::sleep(300); // wait a moment for arm status packet update with joints moving
  bool moving = true;
  while (moving)
  {
     moving = false;
     printf("Joints: ");
     for (int i=1; i<=ArP2Arm::NumJoints; i++)
     {
       printf("[%d] %.0f, ", i, arm.getJointPos(i));
       if (arm.getMoving(i))
       {
            printf("M;  ");
            moving = true;
          }
       else
       {
         printf("NM; ");
       }
         }
     printf("\r");
  }
  printf("\n\n");
arm.moveTo(1,-45,20);
  arm.moveTo(2,-35,30);
  arm.moveTo(3,-70,30);
  arm.moveTo(5,-10,30);
  ArUtil::sleep(7000);
  arm.moveTo(6,90,20);
  ArUtil::sleep(5000);
  ArUtil::sleep(1000);
  printf("Parking arm.\n");
  arm.park();
  // Wait 5 seconds or until arm shuts off
  for(int i = 5; (i > 0) && (arm.getStatus() & ArP2Arm::ArmPower); i--)
  {
    ArUtil::sleep(1000);
  }
  // Disconnect from robot, etc., and exit.
  printf("Shutting down ARIA and exiting.\n");
  Aria::shutdown();
 return(0);
}
```

Anexa 2

Program C++ pentru conducerea A/DML HERA&HORSTMANN, deservită de doi WMRs: Pioneer 3-DX echipat cu Pioneer 5-DOF ARM şi PatrolBot

```
#include "stdafx.h"
#include "AsamblareDezasamblare2roboti.h"
#include "AsamblareDezasamblare2robotiDlg.h"
// pentru fire de executie de win32
#include <windows.h>
#include <tchar.h>
#include <strsafe.h>
void ErrorHandler(LPTSTR lpszFunction) ;
#define MAX_THREADS 4
#define BUF_SIZE 255
#define arm1LA 90
#define arm1R2 -90
// // nomove=1 va seta pe 0 viteza in cele doua functii sliding mode. Utilizat pentru
testarea miscarilor bratului fara deplasarea robotilor.
// // nomove trebuie sa fie in mod normal 0
#define nomove 0
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
#define DAQmxErrChk(functionCall) if( DAQmxFailed(error=(functionCall)) ) goto Error;
else
// CAsamblareDezasamblare2robotiApp
BEGIN_MESSAGE_MAP(CAsamblareDezasamblare2robotiApp, CWinApp)
   ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
// CAsamblareDezasamblare2robotiApp construction
CAsamblareDezasamblare2robotiDlg *m_p_dlg;
// Date globale necesare citirii placii de achizitie
int32     error;
TaskHandle taskHandle;
TaskHandle taskHandle_DO;
uInt32    data;
char      errBuff[2048];
int32     read;
int32 index_record_evenimente_IN,index_record_evenimente_OUT;
// variabile in care se inregistreaze evolutia robotilor
float
*x_pos_pioneer,*y_pos_pioneer,*theta_pioneer,*x_pos_patrolBot,*y_pos_patrolBot,*theta_
patrolBot ;
long long *timp_log;
float
*xe_pos_pioneer,*ye_pos_pioneer,*xe_pos_patrolBot,*ye_pos_patrolBot,*theta_e_pioneer,*
theta_e_patrolBot;
float *v__pioneer ,*v__patrolBot;
int *stare_Pioneer,*stare_PatrolBot;
bool *v_start_dezasamblare;
bool *v_bolt1_pe_pozitie;
bool *v_bolt2_pe_pozitie;
```

```
bool *v_capac_pe_pozitie;
bool *v_corp_pe_pozitie;
bool *v_palet_pe_pozitie;
bool *v_bolt1_preluat;
bool *v_bolt2_preluat;
bool *v_capac_preluat;
bool *v_corp_preluat;
bool *v_palet_preluat;
long contor;
long pas_log;
void alocare_mem_variabile();
void eliberare_mem_variabile();
void salvare_date();
bool date_salvate;  // se confirma ca datele au fost salvate
bool salveaza_datele_acum;  // setat la apasarea unei taste
bool start_inregistrare_date; // setat cand robotii primesc inf de la linia
mecatronica ca boltul 1 a fost scos. resetat cand robotii au ajuns inapoi in pozitia
initiala.
DWORD WINAPI  Thread_InregistrareDate(LPVOID lpParam);
// pentru calcularea timpului
time_t startTime, stopTime, curentTime;
long long mStartTime, mStopTime,mCurentTime;
// semnale transmise intre roboti si linia de asamblare
bool start_dezasamblare;
bool declansare_asamblare;
bool bolt1_pe_pozitie;
bool bolt2_pe_pozitie;
bool capac_pe_pozitie;
bool corp_pe_pozitie;
bool palet_pe_pozitie;
bool bolt1_preluat;
bool bolt2_preluat;
bool capac_preluat;
bool corp_preluat;
bool palet_preluat;
//robot -- structura ce contine informatii despre parametri de deplasare  prin
conducere sliding mode pentru fiecare robot
struct robot{
    double v, av, w, aw;        /* velocity to drive left wheel*/
    double theta_des_next;      /* theta desired */
    double ref_xr, ref_yr, ref_Th, ref_w, ref_v, x_des_next, y_des_next; /* desired
position */
    double x_e, y_e, theta_e, x_e_der, y_e_der, theta_e_der;            /* errors */
    double s_1, s_2, Q1, Q2, P1, P2, gama0, gama_x, gama_y;
    double v_c, v_c_der, w_c, temp1, temp2;
    double Ts, v_l, v_r, timp;
    double x,y, theta;
} r1,r2;
P2ArmJoint *joint;
ArRobot robot1;  // Pioner 3dx
ArRobot robot2;  // PatroBot
int stare_robot1;
int stare_robot2;
int semnal_r1_To_r2;
int semnal_r2_To_r1;
int semnal_start;
bool asteapta_r1;
bool asteapta_r2;
bool eroare_exit;
ArSonarDevice sonar;
ArP2Arm arm;
int viteza_arm;
char *argv1[3];
int argc=3;
```

```
char *argv[3];
char ip_pioneer[]="192.168.0.198";
char setare_ip[]="-rh";
uInt32    dlgDI_curent;
DWORD WINAPI  Thread_PatrolBot(LPVOID lpParam);
DWORD WINAPI Thread_Pioner(LPVOID lpParam);
DWORD WINAPI  citire_NIDAQmx(LPVOID lpParam);
void eliberare_pe_PatrolBot();
void ridicare_de_pe_PatrolBot();
double angleLimit(double angle);
double sign(double value);
double satur(double value);
void sliding1(int value, double vel);
void sliding2(int value, double vel);
void ridicare();
void apucareunu();
void apucare_capac();
void ridicare_capac();
void depozitare_bolt();
void depozitare_corp();
int distante[11];
CAsamblareDezasamblare2robotiApp::CAsamblareDezasamblare2robotiApp()
CAsamblareDezasamblare2robotiApp theApp;
// CAsamblareDezasamblare2robotiApp initialization
BOOL CAsamblareDezasamblare2robotiApp::InitInstance()
{
    Aria::init();
    ArUtil::sleep (10); //las aria sa se initializeze
    InitCommonControls();
    CWinApp::InitInstance();
    AfxEnableControlContainer();
    CAsamblareDezasamblare2robotiDlg dlg;
    m_pMainWnd = &dlg;
    m_p_dlg=&dlg;
    mp_dlgDI_curent=&dlgDI_curent;
    eroare_exit= false;
    asteapta_r1=true;
    asteapta_r2=true;
    date_salvate=false;
    start_inregistrare_date=false;
    salveaza_datele_acum=false;
    stare_robot1=0;
    stare_robot2=0;
    semnal_r1_To_r2=0; // r1=pioner r2=PatrolBot
    semnal_r2_To_r1=0;
    index_record_evenimente_IN=0;
    index_record_evenimente_OUT=0;
    alocare_mem_variabile();
    mp_ip_pioneer=&ip_pioneer[0];
    mp_current_signal_pioneer=&semnal_r1_To_r2;
    error=0;
    taskHandle=0;
    taskHandle_DO=0;
    argc=3;
    argv[0]=theApp.m_lpCmdLine;
    argv[0]=(char*)theApp.m_pszExeName;
    argv[0]=(char*)theApp.m_pszHelpFilePath;
    argv[1]=&setare_ip[0];
    argv[2]=&ip_pioneer[0];
    argv1[0]=argv[0];
    argv1[1]=argv[1];
     DWORD   dwThreadIdArray[MAX_THREADS];
     HANDLE  hThreadArray[MAX_THREADS];
int i=0;
```

```
         hThreadArray[i] = ::CreateThread(
             NULL,                    // default security attributes
             2000000,                       // use default stack size
             Thread_PatrolBot,        // thread function name
             NULL,//pDataArray[i],           // argument to thread function
             0,                       // use default creation flags
             &dwThreadIdArray[i]);    // returns the thread identifier
        if (hThreadArray[i] == NULL)
         {
            ErrorHandler(TEXT("ErrorCreateThread-Thread_PatrolBot"));
            ExitProcess(3);
         }
i++;
hThreadArray[i] = ::CreateThread(
             NULL,                    // default security attributes
             2000000,                       // use default stack size
             Thread_Pioner,        // thread function name
             NULL,//pDataArray[i],           // argument to thread function
             0,                       // use default creation flags
             &dwThreadIdArray[i]);    // returns the thread identifier
        if (hThreadArray[i] == NULL)
         {
            ErrorHandler(TEXT("ErrorCreateThread-Thread_Pioner"));
            ExitProcess(3);
         }
i++;
hThreadArray[i] = ::CreateThread(
             NULL,                    // default security attributes
             2000000,                       // use default stack size
             citire_NIDAQmx,        // thread function name
             NULL,//pDataArray[i],           // argument to thread function
             0,                       // use default creation flags
             &dwThreadIdArray[i]);    // returns the thread identifier
        if (hThreadArray[i] == NULL)
         {
            ErrorHandler(TEXT("ErrorCreateThread-citire_NIDAQmx"));
            ExitProcess(3);
         }
i++;
hThreadArray[i] = ::CreateThread(
             NULL,                    // default security attributes
             2000000,                       // use default stack size
             Thread_InregistrareDate,        // thread function name
             NULL,//pDataArray[i],           // argument to thread function
             0,                       // use default creation flags
             &dwThreadIdArray[i]);    // returns the thread identifier
        if (hThreadArray[i] == NULL)
         {
            ErrorHandler(TEXT("ErrorCreateThread-Thread_InregistrareDate"));
            ExitProcess(3);
         }
declansare_asamblare=true;
    INT_PTR nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    else if (nResponse == IDCANCEL)
    salveaza_datele_acum=true;
    while (!date_salvate)
    {
        ArUtil::sleep(100);
    }
robot1.stopRunning(true);
robot2.stopRunning(true);
        ArUtil::sleep(200);
    Aria::shutdown();
```

```
    Aria::exit() ;
TerminateThread(hThreadArray[0],0);
TerminateThread(hThreadArray[1],0);
TerminateThread(hThreadArray[2],0);
   WaitForMultipleObjects(MAX_THREADS, hThreadArray, TRUE, INFINITE);
    for(int i=0; i<MAX_THREADS; i++)
    {
        CloseHandle(hThreadArray[i]);
    }
exit(0);

    return FALSE;
}
double angleLimit(double angle)
{
    if (angle > 3.14)
        angle = angle - 2*3.14;
    else
        if (angle < -3.14)
            angle = angle + 2*3.14;
    return(angle);
}
/* ------------- SIGN & SATURATION FUNCTION -------------- */
double sign(double value){
    if (value<0)
        return(-1.00);
    else
        if (value>0)
            return(1.00);
        else
            return(1.00);
}
double satur(double value){
    if (value<-1)
        return (-1.00);
    else if (value > 1)
        return (1.00);
    else
        return (value);
}

void sliding1(int value, double vel)
{int i=0;
// algoritm configurat special pt deplasare liniara
r1.ref_v = 0.0;
r1.ref_w = 0.0;
robot1.lock();
ArUtil::sleep(100);
robot1.unlock();
r1.v=vel;
r1.av=0;
r1.w=0;
r1.aw=0;
m_p_dlg->xx_curent_patrolbot=0;
float increment_xx=1.0/value;
for (i=1; i<value; i++){
    robot1.lock();
    r1.theta_des_next = angleLimit(r1.Ts*r1.w+r1.ref_Th);
    r1.x_des_next = r1.Ts*r1.v*cos(r1.ref_Th) + r1.ref_xr;
    r1.y_des_next = r1.Ts*r1.v*sin(r1.ref_Th) + r1.ref_yr;
    r1.x_e = ((robot1.getX()/1000)-r1.ref_xr)*cos(r1.ref_Th)+((robot1.getY()/1000)-
r1.ref_yr)*sin(r1.ref_Th);
    r1.y_e = -((robot1.getX()/1000)-r1.ref_xr)*sin(r1.ref_Th)+((robot1.getY()/1000)-
r1.ref_yr)*cos(r1.ref_Th);
```

```
    r1.theta_e = angleLimit((robot1.getTh()*(3.14/180))-r1.ref_Th);
    if (r1.y_e < 0)
    {
        r1.gama_y = -abs(r1.gama_y);
    }
    else if( r1.y_e > 0 )
    {
        r1.gama_y = abs( r1.gama_y );
    }
    r1.x_e_der = -r1.v + (robot1.getVel()/1000)*cos(r1.theta_e)+r1.y_e*r1.w;
    r1.y_e_der = (robot1.getVel()/1000)*sin(r1.theta_e)-r1.x_e*r1.w;
    r1.theta_e_der = (robot1.getRotVel()*3.14/180)-r1.w;
    /* SLIDING SURFACE */
    r1.s_1 = r1.x_e_der + r1.gama_x*r1.x_e;
    r1.s_2 = r1.y_e_der + r1.gama_y*r1.y_e + r1.gama0*sign(r1.y_e)*(r1.theta_e);
    // CAZUL II (B)
    r1.temp1 = -r1.Q1*satur(r1.s_1/0.5) - (r1.P1*r1.s_1) - r1.gama_x*r1.x_e_der -
(r1.aw*r1.y_e) - r1.w*r1.y_e_der + r1.av;
    r1.temp2 = -r1.Q2*satur(r1.s_2/0.5) - (r1.P2*r1.s_2) - r1.gama_y*r1.y_e_der +
(r1.aw*r1.x_e) + r1.w*r1.x_e_der;
    r1.v_c_der = (r1.temp1 +
(robot1.getVel()/1000)*r1.theta_e_der*sin(r1.theta_e))/(cos(r1.theta_e));
    r1.v_c = (r1.Ts*r1.v_c_der) + r1.ref_v;
    r1.w_c = (r1.temp2-
r1.v_c_der*sin(r1.theta_e))/((robot1.getVel()/1000)*cos(r1.theta_e)+r1.gama0*sign(r1.y
_e))+r1.w;
    // ROBOT
    r1.v_l = r1.v_c + 0.19*r1.w_c; //0.19 //0.28 //0.24
    r1.v_r = r1.v_c - 0.19*r1.w_c;
    // setVel2 (double leftVelocity, double rightVelocity)
    if(nomove==0)
    robot1.setVel2(1000*r1.v_r, 1000*r1.v_l);
    else
robot1.setVel2(0,0);
    r1.ref_xr = r1.x_des_next;
    r1.ref_yr = r1.y_des_next;
    r1.ref_Th = r1.theta_des_next;
    r1.ref_w = r1.w_c;
    r1.ref_v = r1.v_c;
    r1.timp=r1.timp+r1.Ts;
    robot1.unlock();
    ArUtil::sleep(100);
    m_p_dlg->xx_curent_patrolbot+=increment_xx;if(m_p_dlg-
>xx_curent_patrolbot>1)m_p_dlg->xx_curent_patrolbot=1; // pozitie de afisare
}
// oprire robot. Daca se doreste o miscare continua fara oprire, randurile urmatoare
se comenteaza
robot1.lock();
robot1.setVel2(0,0);
robot1.unlock();
ArUtil::sleep(100);
}
void sliding2(int value, double vel)
{int i=0;
vel=-vel;
r2.ref_v = 0.0;
r2.ref_w = 0.0;
robot2.lock();
ArUtil::sleep(100);
robot2.unlock();
r2.v=vel;
r2.av=0;
r2.w=0;
r2.aw=0;
```

```
for (i=1; i<value; i++){
    robot2.lock();
    r2.theta_des_next = angleLimit(r2.Ts*r2.w+r2.ref_Th);
    r2.x_des_next = r2.Ts*r2.v*cos(r2.ref_Th) + r2.ref_xr;
    r2.y_des_next = r2.Ts*r2.v*sin(r2.ref_Th) + r2.ref_yr;
    r2.x_e = ((robot2.getX()/1000)-r2.ref_xr)*cos(r2.ref_Th)+((robot2.getY()/1000)-
r2.ref_yr)*sin(r2.ref_Th);
    r2.y_e = -((robot2.getX()/1000)-r2.ref_xr)*sin(r2.ref_Th)+((robot2.getY()/1000)-
r2.ref_yr)*cos(r2.ref_Th);
    r2.theta_e = angleLimit((robot2.getTh()*(3.14/180))-r2.ref_Th);
    if (r2.y_e < 0)
    {
        r2.gama_y = -abs(r2.gama_y);
    }
    else if( r2.y_e > 0 )
    {
        r2.gama_y = abs( r2.gama_y );
    }
    r2.x_e_der = -r2.v + (robot2.getVel()/1000)*cos(r2.theta_e)+r2.y_e*r2.w;
    r2.y_e_der = (robot2.getVel()/1000)*sin(r2.theta_e)-r2.x_e*r2.w;
    r2.theta_e_der = (robot2.getRotVel()*3.14/180)-r2.w;
    /* SLIDING SURFACE */
    r2.s_1 = r2.x_e_der + r2.gama_x*r2.x_e;
    r2.s_2 = r2.y_e_der + r2.gama_y*r2.y_e + r2.gama0*sign(r2.y_e)*(r2.theta_e);
    // CAZUL II (B)
    r2.temp1 = -r2.Q1*satur(r2.s_1/0.5) - (r2.P1*r2.s_1) - r2.gama_x*r2.x_e_der -
(r2.aw*r2.y_e) - r2.w*r2.y_e_der + r2.av;
    r2.temp2 = -r2.Q2*satur(r2.s_2/0.5) - (r2.P2*r2.s_2) - r2.gama_y*r2.y_e_der +
(r2.aw*r2.x_e) + r2.w*r2.x_e_der;
    r2.v_c_der = (r2.temp1 +
(robot2.getVel()/1000)*r2.theta_e_der*sin(r2.theta_e))/(cos(r2.theta_e));
    r2.v_c = (r2.Ts*r2.v_c_der) + r2.ref_v;
    r2.w_c = (r2.temp2-
r2.v_c_der*sin(r2.theta_e))/((robot2.getVel()/1000)*cos(r2.theta_e)+r2.gama0*sign(r2.y
_e))+r2.w;
    // ROBOT
    r2.v_l = r2.v_c + 0.19*r2.w_c; //0.19 //0.28 //0.24
    r2.v_r = r2.v_c - 0.19*r2.w_c;
        if(nomove==0)
    robot2.setVel2(1000*r2.v_r, 1000*r2.v_l);
    else
robot2.setVel2(0,0);
    r2.ref_xr = r2.x_des_next;
    r2.ref_yr = r2.y_des_next;
    r2.ref_Th = r2.theta_des_next;
    r2.ref_w = r2.w_c;
    r2.ref_v = r2.v_c;
    r2.timp=r2.timp+r2.Ts;
    robot2.unlock();
    ArUtil::sleep(100);
}
// oprire robot. Daca se doreste o miscare continua fara oprire, randurile urmatoare
se comenteaza
robot2.lock();
robot2.setVel2(0,0);
robot2.unlock();
ArUtil::sleep(100);
}
void ridicare()
{ int durata=0;
m_p_dlg->pozitie_griper=2;
m_p_dlg->yy_curent_griper=0;
    arm.moveTo(1,arm1LA,viteza_arm);
arm.moveTo(2,59,viteza_arm);
```

```
arm.moveTo(3,-122,viteza_arm);
arm.moveTo(5,60,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,-28,viteza_arm);
arm.moveTo(5,88,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
arm.moveTo(1,0,viteza_arm);
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,-28,viteza_arm);
arm.moveTo(5,88,viteza_arm);
for(durata=0;durata<10;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
}
void apucareunu()
{int durata=0;
m_p_dlg->pozitie_griper=1;
m_p_dlg->yy_curent_griper=0;
arm.moveTo(1,arm1LA,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,-99,viteza_arm);
arm.moveTo(5,84,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
arm.moveTo(2,47,viteza_arm);
arm.moveTo(3,-122,viteza_arm);
arm.moveTo(5,56,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
arm.moveTo(6, 0, viteza_arm);
m_p_dlg->griper_inchis=1;
for(durata=0;durata<20;durata++)//40
ArUtil::sleep (100);
}
void apucare_capac()
{
    int durata=0;
m_p_dlg->pozitie_griper=1;
m_p_dlg->yy_curent_griper=0;
    arm.moveTo(1,arm1LA,viteza_arm);
    arm.moveTo(2,80,viteza_arm);
    arm.moveTo(3,0,viteza_arm);
    arm.moveTo(5,16,viteza_arm);
    for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}

    //  arm.moveTo(1,40,viteza_arm);
    arm.moveTo(2,16,viteza_arm);
    arm.moveTo(3,0,viteza_arm);
    arm.moveTo(5,80,viteza_arm);
    for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
    arm.moveTo(6, 0, viteza_arm);
    m_p_dlg->griper_inchis=1;
    for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
}
void ridicare_capac()
{   int durata=0;
m_p_dlg->pozitie_griper=2;
```

```
m_p_dlg->yy_curent_griper=0;
//  arm.moveTo(1,40,viteza_arm);
arm.moveTo(2,70,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,80,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
}
void depozitare_bolt()
{   int durata=0;
//rotire spre linie
m_p_dlg->pozitie_griper=1;
m_p_dlg->yy_curent_griper=0;
arm.moveTo(1,arm1LA,viteza_arm);
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,28,viteza_arm);
arm.moveTo(5,80,viteza_arm);
for(durata=0;durata<30;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
//pozitionare in dreptul magaziei
arm.moveTo(1,arm1LA,viteza_arm);
arm.moveTo(2,47,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,0,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
//eliberare
arm.moveTo(6, 100, viteza_arm);
m_p_dlg->griper_inchis=0;
for(durata=0;durata<10;durata++)//40
ArUtil::sleep (100);
//sus
//arm.moveTo(1,40,viteza_arm);
m_p_dlg->pozitie_griper=2;
m_p_dlg->yy_curent_griper=0;
arm.moveTo(2,90,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,88,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
//rotire
arm.moveTo(1,0,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
m_p_dlg->pozitie_griper=0;
}
void depozitare_corp()
{      int durata=0;
m_p_dlg->pozitie_griper=1;
m_p_dlg->yy_curent_griper=0;
//rotire spre linie
arm.moveTo(1,arm1LA,viteza_arm);
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,28,viteza_arm);
arm.moveTo(5,80,viteza_arm);
for(durata=0;durata<30;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
//pozitionare in dreptul magaziei
arm.moveTo(2,70,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,0,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
//eliberare
```

```
arm.moveTo(6, 100, viteza_arm);
m_p_dlg->griper_inchis=0;
for(durata=0;durata<10;durata++)//40
ArUtil::sleep (100);
//sus
m_p_dlg->pozitie_griper=2;
m_p_dlg->yy_curent_griper=0;
arm.moveTo(2,90,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,88,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
//rotire
arm.moveTo(1,0,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
m_p_dlg->pozitie_griper=0;
}
void eliberare_pe_PatrolBot()
{
    int durata=0;
m_p_dlg->pozitie_griper=3;
m_p_dlg->yy_curent_griper=0;
    // ne asiguram ca este in pozitie ridicata pentru a nu lovi celalalt robot
    arm.moveTo(1,0,viteza_arm);
    arm.moveTo(2,100,viteza_arm);
    arm.moveTo(3,0,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
        for(durata=0;durata<20;durata++)//40
    {ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
    arm.moveTo(1,arm1R2,viteza_arm);
    arm.moveTo(2,75,viteza_arm);
    arm.moveTo(3,-30,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
        for(durata=0;durata<20;durata++)//40
    {ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
    //pozitionare brat deasupra celui de al doilea robot
    arm.moveTo(1,arm1R2,viteza_arm);
    arm.moveTo(2,53,viteza_arm);
    arm.moveTo(3,-65,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
    for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/60;}
    arm.moveTo(6, 100, viteza_arm);
    m_p_dlg->griper_inchis=0;
    for(durata=0;durata<20;durata++)//40
        ArUtil::sleep (100);
m_p_dlg->pozitie_griper=4;
m_p_dlg->yy_curent_griper=0;
    arm.moveTo(1,arm1R2,viteza_arm);
    arm.moveTo(2,75,viteza_arm);
    arm.moveTo(3,-30,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
    for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
    arm.moveTo(1,0,viteza_arm);
    arm.moveTo(2,100,viteza_arm);
    arm.moveTo(3,0,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
    for(durata=0;durata<30;durata++)//40
        ArUtil::sleep (100);
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/50;}
m_p_dlg->pozitie_griper=0;
}
```

```
DWORD WINAPI  Thread_PatrolBot(LPVOID lpParam)
{
    TRACE("Thread_PatrolBot() called.\n");
    ArUtil::sleep(100);
    int argc1=3;
    char ip_PatrolBot[]="192.168.1.196";
    argv1[2]=ip_PatrolBot;
    ArArgumentParser parser(&argc1, argv1);
    parser.loadDefaultArguments();
    ArSimpleConnector simpleConnector(&parser);
    robot2;
    // ArSonarDevice sonar;
    ArAnalogGyro gyro(&robot2);
    ArKeyHandler keyHandler;
    Aria::setKeyHandler(&keyHandler);
    robot2.attachKeyHandler(&keyHandler);
    TRACE("You may press escape to exit\n");
        ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
    ArActionLimiterForwards limiterFarAction("speed limiter far", 600, 2200, 800);
    ArActionLimiterTableSensor tableLimiterAction;
    robot2.addAction(&tableLimiterAction, 100);
    robot2.addAction(&limiterAction, 95);
    robot2.addAction(&limiterFarAction, 90);
        if (!Aria::parseArgs() || !parser.checkHelpAndWarnUnparsed())
    {    robot1.stopRunning(true);
        Aria::logOptions();
        exit(1);
    }
    // Connect to the robot2
    if (!simpleConnector.connectRobot(&robot2))
    {
        TRACE("Could not connect to robot2... exiting\n");
robot1.stopRunning(true);
        Aria::exit(1);
    }
    robot2.runAsync(true);
    robot2.enableMotors();
    r2.timp=0.0;
    r2.Ts=0.1; // 0.1 secunde=100ms perioada de esantionare a comenzii sliding mode- de
fapt egala cu perioada de citire a datelor de la roboti si de trimitere a oricarei
comenzi
    r2.Q1 = 0.05;    //0.05    0.5
    r2.Q2 = 0.5;        //0.5     0.75
    r2.P1 = 0.5;     //0.5     1.75
    r2.P2 = 0.75;       //1.75    0.75
    r2.gama0 = 30;   //20
    r2.gama_x = 0.75;    //1.25
    r2.gama_y = 25;      //15
    r2.ref_xr = 0.0;
    r2.ref_yr = 0.0;
    r2.ref_Th = 0.0;
    r2.ref_v = 0.00;
    r2.ref_w = 0.00;
    while (true)
    {
        //apuc primul bolt
        // asteptare comanda deplasare de la r1 dupa ce acesta a depus piesa pe robotul
curent r2
        while (semnal_r1_To_r2 != 1)
            ArUtil::sleep (10);
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=1;
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
```

```
        sliding2(98,0.1);
        robot2.setVel2(0,0);
        // confirmare ca s-a ajuns in dreptul magaziei de bolturi
        while (semnal_r1_To_r2 != 2)
            ArUtil::sleep (10);
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=2;
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
        // s-a finalizat depozitarea boltului. se asteapta confirmarea ca r2 este gata
pt deplasare inapoi
        while (semnal_r1_To_r2 != 3)
            ArUtil::sleep (10);
        printf(" r1<-r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=3;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        sliding2(68,-0.1);//65
        robot2.setVel2(0,0);
        ArUtil::sleep (100);
        //se asteapta semnal de la  r2 ca e pe pozitie pt a ne apuca sa ridicam boltul
2
        while (semnal_r1_To_r2 != 4)
            ArUtil::sleep (10);
        printf(" r1<-r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=4;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //sincronizare. Bolt 2 a fost plasat. Start deplasare catre magazie.
        while (semnal_r1_To_r2 != 5)
            ArUtil::sleep (10);
        printf(" r1<-r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=5;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        // deplasare catre magazie bolturi
        sliding2(68,0.1);
        robot2.setVel2(0,0);
        ArUtil::sleep (100);


        // anunt  r2 ca eu(i.e. r1) sunt pe pozitie
        while (semnal_r1_To_r2 != 6)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=6;
        // asteptare confirmare ca si r2 este pe pozitie
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //!sincronizare
        // deplasare pentru ridicare capac
        // anunt  r2 sa se deplaseze spre noua pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 7)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        semnal_r2_To_r1=7;
        // asteptare confirmare ca si r2 este pe pozitie
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        printf("Deplasare catre ridicat capacul\n");
        sliding2(37,0.1);
        robot2.setVel2(0,0);
        printf("Pozitionat pt ridicat capacul\n");
```

```
        //Tranzitie: Astept confirmarea si de la R2
        while (semnal_r1_To_r2 != 8)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat pt capac.
        semnal_r2_To_r1=8;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este pe pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 9)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r2_To_r1=9;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        sliding2(32,0.1);
        robot2.setVel2(0,0);
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r1_To_r2 != 10)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de capace.
        semnal_r2_To_r1=10;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
        //!sincronizare
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 11)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r2_To_r1=11;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
        // deplasare pentru ridicare corp
        printf("Deplasare catre ridicat corp\n");
        sliding2(45,0.1);
        robot2.setVel2(0,0);
        printf("Pozitionat pt ridicat corp\n");
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r1_To_r2 != 12)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul pozitiei de ridicare
corp.
        semnal_r2_To_r1=12;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este incepe sa se deplaseze spre
pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 13)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r2_To_r1=13;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        sliding2(33,0.1);
        robot2.setVel2(0,0);
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
```

```
        while (semnal_r1_To_r2 != 14)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de corpuri.
        semnal_r2_To_r1=14;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //!sincronizare
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 15)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r2_To_r1=15;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
        // deplasare pentru ridicare palet
        printf("Deplasare catre ridicat palet\n");
        sliding2(42,0.1);
        robot2.setVel2(0,0);
        printf("Pozitionare pt ridicat palet\n");
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r1_To_r2 != 16)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul pozitiei de ridicare
palet.
        semnal_r2_To_r1=16;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este incepe sa se deplaseze spre
pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 17)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre noua pozitie/magazie
        semnal_r2_To_r1=17;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        sliding2(34,0.1);
        robot2.setVel2(0,0);


        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r1_To_r2 != 18)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de paleti.
        semnal_r2_To_r1=18;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (100);//1000
        //!sincronizare
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 19)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 sa se deplaseze spre pozitie 0 /referinta
        semnal_r2_To_r1=19;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
```

```
        sliding2(315,-0.1);  // din calcule 326 dar erorile de deplasare se cumuleaza.
trebuie revizuit alg sliding
        robot2.setVel2(0,0);
        //!sincronizare
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r1_To_r2 != 20)
            ArUtil::sleep (10);
        printf(" r1->r2 %d \n",semnal_r1_To_r2);
        // anunt  r2 ca am ajuns la pozitie 0 /referinta
        semnal_r2_To_r1=20;
        printf(" r1->r2 %d \n",semnal_r2_To_r1);
        ArUtil::sleep (10);//1000
    } //
    ArUtil::sleep (100);//1000
    while (semnal_r1_To_r2 != 1)
    {ArUtil::sleep (100);
    }
    semnal_r2_To_r1=1;
    printf(" r2<-r1 %d \n",semnal_r1_To_r2);
    printf(" r2->r1 %d \n",semnal_r2_To_r1);
    {
        sliding2(100,0.1);
        //  p=0;m=0;
    }
    robot2.lock();
    robot2.setVel2(0,0);
    robot2.unlock();
    ArUtil::sleep (100);//1000
    while (semnal_r1_To_r2 != 2)
    {ArUtil::sleep (100);

    }
    semnal_r2_To_r1=2;
    printf(" r2<-r1 %d \n",semnal_r1_To_r2);
    printf(" r2->r1 %d \n",semnal_r2_To_r1);
    {
        sliding2(65,-0.1);
            }
    robot2.lock();
    robot2.setVel2(0,0);
    robot2.unlock();
    ArUtil::sleep(10);
    asteapta_r2 =false;
    while (asteapta_r1)
    {
        ArUtil::sleep(10);
    }
    printf("PatrolBot se deconecteaza() iese.\n");
    return 1;
}
DWORD WINAPI Thread_Pioner(LPVOID lpParam){
    ArUtil::sleep (100);
    errBuff[0]='\0';
    eroare_exit= false;
    asteapta_r1=true;
    asteapta_r2=true;
    stare_robot1=0;
    stare_robot2=0;
    semnal_r1_To_r2=0; // r1=pioner r2=PatrolBot
    semnal_r2_To_r1=0;
    ArArgumentParser parser(&argc, argv);
    parser.loadDefaultArguments();
    ArSimpleConnector simpleConnector(&parser);
```

```
    ArAnalogGyro gyro(&robot1);
    robot1.addRangeDevice(&sonar);
    ArKeyHandler keyHandler;
    Aria::setKeyHandler(&keyHandler);
    robot1.attachKeyHandler(&keyHandler);
    printf("You may press escape to exit\n");
    printf("parametri executie  argc=%d argv[0]=%s  argv[1]=%s argv[2]=%s\n", argc ,
argv[0], argv[1], argv[2]);
    ArUtil::sleep(100);
    ArActionLimiterForwards limiterAction("speed limiter near", 300, 600, 250);
    ArActionLimiterForwards limiterFarAction("speed limiter far", 300, 11000, 400);
    //ArActionLimiterForwards limiterFarAction("speed limiter far", 600, 2200, 800);
    ArActionLimiterTableSensor tableLimiterAction;
    robot1.addAction(&tableLimiterAction, 100);
    robot1.addAction(&limiterAction, 95);
    robot1.addAction(&limiterFarAction, 90);
    ArSimpleConnector con(&argc, argv);
    if(!Aria::parseArgs())
    {
        Aria::logOptions();
        Aria::shutdown();
        return -1;
    }
    //trebuie necomentata cand se lucreaza cu bratul
    ArLog::log(ArLog::Normal, "armExample: Connecting to the robot1.");
    if(!con.connectRobot(&robot1))
    {
        ArLog::log(ArLog::Terse, "armExample: Could not connect to the robot1.
Exiting.");
        Aria::shutdown();
        return -1;
    }
    robot1.runAsync(true);
    ArUtil::sleep(150);
    arm.setRobot(&robot1);
    ArUtil::sleep(150);
    viteza_arm=10;
    if (arm.init() != ArP2Arm::SUCCESS)
    {
        ArLog::log(ArLog::Terse, "armExample: Error initializing the P2 Arm!");
        robot1.stopRunning(true);
        eroare_exit=true;
        return -1;
    }
    robot1.enableMotors();
    // Print out some of the settings

    TRACE("Current joint info:\nJoint   Vel  Home  Center\n");
    for (int i=1; i<=ArP2Arm::NumJoints; i++)
    {
        joint = arm.getJoint(i);
        TRACE("  %2i:  %5i %5i   %5i\n", i, joint->myVel, joint->myHome, joint-
>myCenter);
    }
    TRACE("\n");
    // Put the arm to work
    TRACE("Powering on  (takes a couple seconds to stabilize)\n");
    arm.powerOn();
    // Request one status packet and print out the arm's status
    TRACE("Current arm status:\n");
    arm.requestStatus(ArP2Arm::StatusSingle);
    ArUtil::sleep(200);  // Give time to get the packet
    TRACE("Arm Status: ");
    if (arm.getStatus() & ArP2Arm::ArmGood)
```

```
        TRACE("Good=1 ");
    else
        TRACE("Good=0 ");
    if (arm.getStatus() & ArP2Arm::ArmInited)
        TRACE("Inited=1 ");
    else
        TRACE("Inited=0 ");
    if (arm.getStatus() & ArP2Arm::ArmPower)
        TRACE("Power=1 ");
    else
        TRACE("Power=0 ");
    if (arm.getStatus() & ArP2Arm::ArmHoming)
        TRACE("Homing=1 ");
    else
        TRACE("Homing=0 ");
    TRACE(" \n \n");

    // setare pozitie asteptare
    arm.moveTo(1,0,viteza_arm);
    arm.moveTo(2,95,viteza_arm);
    arm.moveTo(3,0,viteza_arm);
    arm.moveTo(5,0,viteza_arm);
    ArUtil::sleep (1000);//(20500);//3000
    r1.timp=0.0;
    r1.Ts=0.1;
    r1.Q1 = 0.05;     //0.05     0.5
    r1.Q2 = 0.5;         //0.5      0.75
    r1.P1 = 0.5;      //0.5      1.75
    r1.P2 = 0.75;        //1.75     0.75
    r1.gama0 = 30;    //20
    r1.gama_x = 0.75;     //1.25
    r1.gama_y = 25;       //15
    r1.ref_xr = 0.0;
    r1.ref_yr = 0.0;
    r1.ref_Th = 0.0;
    r1.ref_v = 0.00;
    r1.ref_w = 0.00;
    ArUtil::sleep (200);
    {
start_dezasamblare_eticheta:
        start_inregistrare_date=false;
        while (!start_dezasamblare )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("\nstart_dezasamblare\n");
        start_inregistrare_date=true;
        while (!bolt1_pe_pozitie )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("bolt1_pe_pozitie\n");


        //apuc primul bolt
        apucareunu();
        ArUtil::sleep (200);//1000
        bolt1_preluat=true;
        ridicare();
        eliberare_pe_PatrolBot();
        //!! comentate pentru deplasare independenta
        semnal_r1_To_r2=1;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        while (semnal_r2_To_r1 != 1)
            ArUtil::sleep (10);
```

```
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        sliding1(98,0.1);
        robot1.setVel2(0,0);
        // confirmare ca s-a ajuns in dreptul magaziei de bolturi
        semnal_r1_To_r2=2;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        while (semnal_r2_To_r1 != 2)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ridicare_de_pe_PatrolBot();
        depozitare_bolt(); //boltul 1
        // s-a finalizat depozitarea boltului. se asteapta confirmarea ca r2 este gata
pt deplasare inapoi
        semnal_r1_To_r2=3;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        while (semnal_r2_To_r1 != 3)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        sliding1(68,-0.1);//65
        robot1.setVel2(0,0);
        ArUtil::sleep (100);
        //se asteapta semnal de la  r2 ca e pe pozitie pt a ne apuca sa ridicam boltul
2
        semnal_r1_To_r2=4;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        while (semnal_r2_To_r1 != 4)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        while (!bolt2_pe_pozitie )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("bolt2_pe_pozitie\n");
        bolt1_preluat=false;
        //ridicare bolt 2
        apucareunu();
        ArUtil::sleep (200);//1000
        bolt2_preluat=true;
        ridicare();
        eliberare_pe_PatrolBot();
        //sincronizare. Bolt 2 a fost plasat. Start deplasare catre magazie.
        semnal_r1_To_r2=5;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        while (semnal_r2_To_r1 != 5)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);



        // deplasare catre magazie bolturi
        sliding1(68,0.1);
        robot1.setVel2(0,0);
        ArUtil::sleep (100);
        // anunt  r2 ca eu(i.e. r1) sunt pe pozitie
        semnal_r1_To_r2=6;
        // asteptare confirmare ca si r2 este pe pozitie
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        while (semnal_r2_To_r1 != 6)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
```

```
        // activitate
        ridicare_de_pe_PatrolBot();
        depozitare_bolt();
        //!sincronizare
        // deplasare pentru ridicare capac
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r1_To_r2=7;
        // asteptare confirmare ca si r2 este pe pozitie
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 7)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);

        TRACE("Deplasare catre ridicat capacul\n");
        sliding1(37,0.1);
        robot1.setVel2(0,0);
        TRACE("Pozitionat pt ridicat capacul\n");

        //sincronizare. Anunt R2 ca m-am pozitionat pt capac.
        semnal_r1_To_r2=8;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: Astept confirmarea si de la R2
        while (semnal_r2_To_r1 != 8)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        while (!capac_pe_pozitie )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("capac_pe_pozitie\n");
        bolt2_preluat=false;
        //  apucaretrei();
        apucare_capac();
        capac_preluat=true;
        ridicare_capac();
        ArUtil::sleep (100);//1000
        TRACE("S-a ridicat capacul\n");
        eliberare_pe_PatrolBot();
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r1_To_r2=9;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este pe pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 9)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);



        sliding1(32,0.1);
        robot1.setVel2(0,0);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de capace.
        semnal_r1_To_r2=10;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r2_To_r1 != 10)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ridicare_de_pe_PatrolBot();
        depozitare_corp(); // de fapt capac dar operatiile la magazii sunt similare
```

```
        //!sincronizare
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r1_To_r2=11;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 11)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        // deplasare pentru ridicare corp
        TRACE("Deplasare catre ridicat corp\n");
        sliding1(45,0.1);
        robot1.setVel2(0,0);
        TRACE("Pozitionat pt ridicat corp\n");
        //  apucaretrei();
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul pozitiei de ridicare
corp.
        semnal_r1_To_r2=12;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r2_To_r1 != 12)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        while (!corp_pe_pozitie )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("corp_pe_pozitie\n");
        capac_preluat=false;
        apucare_capac(); //defat corp
        corp_preluat=true;
        ridicare_capac();
        ArUtil::sleep (100);//1000
        //ridicare();
        TRACE("S-a ridicat corp\n");
        eliberare_pe_PatrolBot();
        ArUtil::sleep(100);//2000
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r1_To_r2=13;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este incepe sa se deplaseze spre
pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 13)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        sliding1(33,0.1);
        robot1.setVel2(0,0);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de corpuri.
        semnal_r1_To_r2=14;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000

        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r2_To_r1 != 14)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ridicare_de_pe_PatrolBot(); // a corpului
        depozitare_corp();
        //!sincronizare
        // anunt  r2 sa se deplaseze spre noua pozitie
        semnal_r1_To_r2=15;
```

```
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 15)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        // deplasare pentru ridicare palet
        TRACE("Deplasare catre ridicat palet\n");
        sliding1(42,0.1);
        robot1.setVel2(0,0);
        TRACE("Pozitionare pt ridicat palet\n");
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul pozitiei de ridicare
palet.
        semnal_r1_To_r2=16;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r2_To_r1 != 16)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        while (!palet_pe_pozitie )
        {ArUtil::sleep (200);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("corp_pe_pozitie\n");
        corp_preluat=false;
        apucare_capac(); //palet
        palet_preluat=true;
        ridicare_capac();
        ArUtil::sleep (200);//1000
        //  ridicare();
        TRACE("S-a ridicat palet\n");
        eliberare_pe_PatrolBot();
        ArUtil::sleep(200);//2000
        // anunt  r2 sa se deplaseze spre noua pozitie/magazie
        semnal_r1_To_r2=17;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: asteptare confirmare ca si r2 este incepe sa se deplaseze spre
pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 17)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        sliding1(34,0.1);
        robot1.setVel2(0,0);
        //sincronizare. Anunt R2 ca m-am pozitionat in dreptul magaziei de paleti.
        semnal_r1_To_r2=18;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (100);//1000
        //Tranzitie: Astept confirmarea si de la R2 ca e pe pozitie
        while (semnal_r2_To_r1 != 18)
           ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        ridicare_de_pe_PatrolBot();
        depozitare_corp(); // palet


        //!sincronizare
        // anunt  r2 sa se deplaseze spre pozitie 0 /referinta
        semnal_r1_To_r2=19;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
```

```
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 19)
            ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        //sliding1(326,-0.1);
        sliding1(315,-0.1);
        palet_preluat=false;
        //!sincronizare
        // anunt  r2 ca am ajuns la pozitie 0 /referinta
        semnal_r1_To_r2=20;
        TRACE(" r1->r2 %d \n",semnal_r1_To_r2);
        ArUtil::sleep (10);//1000
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        while (semnal_r2_To_r1 != 20)
            ArUtil::sleep (10);
        TRACE(" r1<-r2 %d \n",semnal_r2_To_r1);
        semnal_r1_To_r2=0;
        salvare_date();
        //Tranzitie: asteptare confirmare ca r2 se deplaseaza spre pozitie
        // asteptare pt indeplinire conditii tranzitie
        semnal_start=1;
        while (semnal_start != 1)
        {ArUtil::sleep (100);   arm.moveTo(1,0,viteza_arm);} // trimit comanda bratului
fara sa-l misc doar pentru a nu intra in pozitie de standby
        TRACE("\n\n !!!!  pregatit pentru o noua dezasamblare \n");
        semnal_start=0;
        goto start_dezasamblare_eticheta;
        ArUtil::sleep (100);//3000
    }
    asteapta_r1 =false;
    while (asteapta_r2)
    {
        ArUtil::sleep(100);
    }
    TRACE("Poiner se deconecteaza() iese.\n");
    return 2;
}
// functie ce se ocupa cu comunicatia cu linia de asamblare/dezasamblare
// realizeaza atat citirea cat si scrierea semnalelor
DWORD WINAPI  citire_NIDAQmx(LPVOID lpParam){
    TRACE("S-A CREAT firul ----citire_NIDAQmx ");
    uInt32 DI_curent=0xf;
    uInt32 DI_anterior=0xff;
    uInt8 DO_curent=0;
    int32 esantioane_scrise=0;
    DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
    DAQmxErrChk (DAQmxCreateTask("",&taskHandle_DO));
    DAQmxErrChk
(DAQmxCreateDIChan(taskHandle,"Dev1/port1","",DAQmx_Val_ChanForAllLines));  // intrari
    DAQmxErrChk
(DAQmxCreateDOChan(taskHandle_DO,"Dev1/port0","",DAQmx_Val_ChanForAllLines));
    DAQmxErrChk (DAQmxStartTask(taskHandle));
    int i=0;
    while (true){
        i=(i+1)%40;
Citire:
        esantioane_scrise=0;
        //citire
        {DAQmxErrChk
(DAQmxReadDigitalU32(taskHandle,1,0.1,DAQmx_Val_GroupByChannel,&DI_curent,1,&read,NULL
));
        DI_curent=DI_curent& 0x0000000f; // selectam doar primii 4 biti de interes
        if(i==1)TRACE(" DI_curent=%d  ",DI_curent);
```

```
       if(DI_anterior!=DI_curent) // daca s-a produs o schimbare in valoarea citita
semnalizez celorlalte fire
       {
           DI_anterior=DI_curent;
       if ((DI_curent& 0x1)==0) // B0-activ pe 0
       {
           dlgDI_curent=DI_curent; // il transferam prelucrat catre interfata grafica
           start_dezasamblare=true;
           DI_curent=DI_curent>>1;
           TRACE(" DI_curent>>1 =%d;  ",DI_curent);
           DI_curent=7-DI_curent;
           TRACE(" 7-DI_curent =%d; \n",DI_curent);
           bolt1_pe_pozitie=false;
           bolt2_pe_pozitie=false;
           capac_pe_pozitie=false;
           corp_pe_pozitie=false;
           palet_pe_pozitie=false;
           switch (DI_curent)
           {
           case 0:
               bolt1_pe_pozitie=true;
               printf(" AL->r1 bolt1_pe_pozitie \n");
               break;
           case 1:
               bolt2_pe_pozitie=true;
               printf(" AL->r1 bolt2_pe_pozitie \n");
               break;
           case 2:
               capac_pe_pozitie=true;
               printf(" AL->r1 capac_pe_pozitie \n");
               break;
           case 3:
               corp_pe_pozitie=true;
               printf(" AL->r1 corp_pe_pozitie \n");
               break;
           case 4:
               palet_pe_pozitie=true;
               printf(" AL->r1 palet_pe_pozitie \n");
               break;
           default:
               break;
           }
       }
       else
       {start_dezasamblare=false;
       bolt1_pe_pozitie=false;
       bolt2_pe_pozitie=false;
       capac_pe_pozitie=false;
       corp_pe_pozitie=false;
       palet_pe_pozitie=false;
       }
       index_record_evenimente_IN++;

       }
       }
       // scriere
       {
           DO_curent=0;
           if( bolt1_preluat)DO_curent=1;else
               if( bolt2_preluat)DO_curent=2;else
                   if( capac_preluat)DO_curent=3;else
                       if( corp_preluat)DO_curent=4;else
                           if( palet_preluat)DO_curent=5;
if(declansare_asamblare)
```

```
{DO_curent=DO_curent+8;
declansare_asamblare=false; // cele 50 de ms cat lasam semnalul in 1 pe bitul 4 din
iesirile digitale ar trebui sa fie suficient pentru a fi sesizat de PLC
}
          DO_curent=0xff-DO_curent;
          DO_curent=DO_curent<<1;
          //DO_curent=0xf-(1<<i);
          //i++;
          DAQmxErrChk
(DAQmxWriteDigitalU8(taskHandle_DO,1,1,0.1,DAQmx_Val_GroupByChannel,&DO_curent,&esanti
oane_scrise,NULL));
       }
       ArUtil::sleep(50);
       if (m_p_dlg!=NULL){
       }
   }
Error:
   TRACE("\n DAQmx Error: \n");
   if( DAQmxFailed(error) )
      DAQmxGetExtendedErrorInfo(errBuff,2048);
   if( DAQmxFailed(error) )
      TRACE("\n DAQmx Error: %s\n",errBuff);
   ArUtil::sleep(100);
   goto Citire;
   /*printf("End of program, press Enter key to quit\n");
   getchar();*/
   return 3;
}
void ridicare_de_pe_PatrolBot()
{ int durata=0;
m_p_dlg->pozitie_griper=3;
m_p_dlg->yy_curent_griper=0;
// ne asiguram ca este in pozitie ridicata pentru a nu lovi celalalt robot
arm.moveTo(1,-60,viteza_arm);
arm.moveTo(2,75,viteza_arm);
arm.moveTo(3,-20,viteza_arm);
arm.moveTo(5,0,viteza_arm);
for(durata=0;durata<20;durata++)
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
//pozitionare brat deasupra celui de al doilea robot
arm.moveTo(1,arm1R2,viteza_arm);
arm.moveTo(2,53,viteza_arm);
arm.moveTo(3,-65,viteza_arm);
arm.moveTo(5,0,viteza_arm);
for(durata=0;durata<20;durata++)
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
arm.moveTo(6, 10, viteza_arm);
m_p_dlg->griper_inchis=1;
for(durata=0;durata<20;durata++)//40
ArUtil::sleep (100);
m_p_dlg->pozitie_griper=4;
m_p_dlg->yy_curent_griper=0;
//arm.moveTo(1,40,viteza_arm);
arm.moveTo(2,80,viteza_arm);
arm.moveTo(3,-30,viteza_arm);
//arm.moveTo(5,90,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
arm.moveTo(1,0,viteza_arm);
arm.moveTo(2,90,viteza_arm);
arm.moveTo(3,0,viteza_arm);
arm.moveTo(5,0,viteza_arm);
for(durata=0;durata<20;durata++)//40
{ArUtil::sleep (100); m_p_dlg->yy_curent_griper+=1.0/40;}
```

```
}
void salvare_date(){
    long i;
    struct tm * delta;
        delta = gmtime(&startTime);
    char FileName[255];
    sprintf(FileName,"Hera_dezasamblare_date_%02d_%02d_%04d__%02d_%02d.m", (delta-
>tm_mday),(delta->tm_mon)+1 ,(delta->tm_year) +1900,delta->tm_hour+2, delta->tm_min);
    FILE *outFile = fopen(FileName, "w+");
    pas_log=pas_log-1;
    fprintf(outFile,"nr_esantioane= %d;\n",pas_log);
    // salvare inregistrari
    // x_pos_pioneer*,y_pos_pioneer*,x_pos_patrolBot*,y_pos_patrolBot*,timp*;
    fprintf(outFile,"x_pos_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",x_pos_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"y_pos_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",y_pos_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"x_pos_patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",x_pos_patrolBot[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"y_pos_patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",y_pos_patrolBot[i]);
    fprintf(outFile,"];\n\n");

    fprintf(outFile,"theta_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",theta_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"theta_patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",theta_patrolBot[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"timp_log=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",(float)timp_log[i]*0.001);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"xe_pos_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",xe_pos_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"ye_pos_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",ye_pos_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"xe_pos_patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",xe_pos_patrolBot[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"ye_pos_patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",ye_pos_patrolBot[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"theta_e_pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",theta_e_pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"theta_e_patrolBot=[");
    for(i=0;i<pas_log;i++)
```

```
        fprintf(outFile,"%.3f ",theta_e_patrolBot[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"v__pioneer=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",v__pioneer[i]);
    fprintf(outFile,"];\n\n");
    fprintf(outFile,"v__patrolBot=[");
    for(i=0;i<pas_log;i++)
        fprintf(outFile,"%.3f ",v__patrolBot[i]);
    fprintf(outFile,"];\n\n");
fclose(outFile);
date_salvate=true;
}
void alocare_mem_variabile(){
    x_pos_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    y_pos_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    theta_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    x_pos_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    y_pos_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    theta_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    timp_log = (long long *)malloc(durata_inregistrare_esantioane*sizeof(long long));
    xe_pos_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    ye_pos_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    xe_pos_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    ye_pos_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    theta_e_pioneer = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    theta_e_patrolBot = (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    v__pioneer= (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    v__patrolBot= (float *)malloc(durata_inregistrare_esantioane*sizeof(float));
    stare_Pioneer = (int *)malloc(durata_inregistrare_esantioane*sizeof(int));
    stare_PatrolBot= (int *)malloc(durata_inregistrare_esantioane*sizeof(int));
    v_start_dezasamblare = (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_bolt1_pe_pozitie= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_bolt2_pe_pozitie= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_capac_pe_pozitie= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_corp_pe_pozitie= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_palet_pe_pozitie= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_bolt1_preluat= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_bolt2_preluat= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_capac_preluat= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_corp_preluat= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
    v_palet_preluat= (bool *)malloc(durata_inregistrare_esantioane*sizeof(bool));
}
void eliberare_mem_variabile(){
    free(x_pos_pioneer);
    free(y_pos_pioneer);
    free(theta_pioneer );
    free(x_pos_patrolBot);
    free(y_pos_patrolBot);
    free(theta_patrolBot);
    free(timp_log);
    free(xe_pos_pioneer);
    free(ye_pos_pioneer );
    free(xe_pos_patrolBot);
    free(ye_pos_patrolBot);
    free(theta_e_pioneer);
    free(theta_e_patrolBot);
    free(v__pioneer);
    free(v__patrolBot);
    free(stare_Pioneer);
    free(stare_PatrolBot);
    free(v_start_dezasamblare);
    free(v_bolt1_pe_pozitie);
    free(v_bolt2_pe_pozitie);
```

```
    free(v_capac_pe_pozitie);
    free(v_corp_pe_pozitie);
    free(v_palet_pe_pozitie);
    free(v_bolt1_preluat);
    free(v_bolt2_preluat);
    free(v_capac_preluat);
    free(v_corp_preluat);
    free(v_palet_preluat);
}
DWORD WINAPI  Thread_InregistrareDate(LPVOID lpParam){
    //start timing
    while(!start_inregistrare_date)
        ArUtil::sleep (100);
    startTime = time(NULL);
    mStartTime = mtime();
    pas_log=-1;
    while(pas_log<durata_inregistrare_esantioane  && (!salveaza_datele_acum))
    { pas_log++;
    if(pas_log%100==0) TRACE("\npas_log=%dL",pas_log);
    curentTime = time(NULL);
    mCurentTime = mtime();
    timp_log[pas_log]= mCurentTime-mStartTime;
    x_pos_pioneer[pas_log]=(float)robot1.getX();
    y_pos_pioneer[pas_log]=(float)robot1.getY();
    theta_pioneer[pas_log]=(float)robot1.getTh();
    xe_pos_pioneer[pas_log]=r1.x_e;
    ye_pos_pioneer[pas_log]=r1.y_e;
    theta_e_pioneer[pas_log]=r1.theta_e;
    v__pioneer[pas_log]=r1.v_c ;
    stare_Pioneer[pas_log]=semnal_r1_To_r2;
    x_pos_patrolBot[pas_log]=(float)robot2.getX();
    y_pos_patrolBot[pas_log]=(float)robot2.getY();
    theta_patrolBot[pas_log]=(float)robot2.getTh();
    xe_pos_patrolBot[pas_log]=r2.x_e;
    ye_pos_patrolBot[pas_log]=r2.y_e;
    theta_e_patrolBot[pas_log]=r2.theta_e;
    v__patrolBot[pas_log]=r2.v_c ;
    stare_PatrolBot[pas_log]=semnal_r2_To_r1;
    v_start_dezasamblare[pas_log]=start_dezasamblare;
    v_bolt1_pe_pozitie[pas_log]=bolt1_pe_pozitie;
    v_bolt2_pe_pozitie[pas_log]=bolt2_pe_pozitie;
    v_capac_pe_pozitie[pas_log]=capac_pe_pozitie;
    v_corp_pe_pozitie[pas_log]=corp_pe_pozitie;
    v_palet_pe_pozitie[pas_log]=palet_pe_pozitie;
    v_bolt1_preluat[pas_log]=bolt1_preluat;
    v_bolt2_preluat[pas_log]=bolt2_preluat;
    v_capac_preluat[pas_log]=capac_preluat;
    v_corp_preluat[pas_log]=corp_preluat;
    v_palet_preluat[pas_log]=palet_preluat;
    ArUtil::sleep (100);
    }
    time_t deltaTime;
    struct tm * delta;
        mStopTime = mtime();
    stopTime = time(NULL);
        deltaTime = stopTime-startTime;
    delta = gmtime(&deltaTime);
    salvare_date();
    eliberare_mem_variabile();
    date_salvate=true;
return 4;
}
void ErrorHandler(LPTSTR lpszFunction)
{
```

```
        LPVOID lpMsgBuf;
    LPVOID lpDisplayBuf;
    DWORD dw = GetLastError();
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER |
        FORMAT_MESSAGE_FROM_SYSTEM |
        FORMAT_MESSAGE_IGNORE_INSERTS,
        NULL,
        dw,
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR) &lpMsgBuf,
        0, NULL );
        lpDisplayBuf = (LPVOID)LocalAlloc(LMEM_ZEROINIT,
        (lstrlen((LPCTSTR) lpMsgBuf) + lstrlen((LPCTSTR) lpszFunction) + 40) *
sizeof(TCHAR));
    StringCchPrintf((LPTSTR)lpDisplayBuf,
        LocalSize(lpDisplayBuf) / sizeof(TCHAR),
        TEXT("%s failed with error %d: %s"),
        lpszFunction, dw, lpMsgBuf);
    MessageBox(NULL, (LPCTSTR) lpDisplayBuf, TEXT("Error"), MB_OK);
        LocalFree(lpMsgBuf);
    LocalFree(lpDisplayBuf);
}
```