

Control Tolerant la Defecte pentru Roboți Mobili

Universitatea „Dunărea de Jos”, Galați
Facultatea de Știință Calculatoarelor
Departament : Automatică și Informatică Industrială



Lucrare de licență

**Temă : Control Tolerant la Defecte pentru
Roboți Mobili**

Domeniu : Ingineria Sistemelor, Robotică

Student : Axenie Cristian

Specializare : Automatică și Informatică Aplicată

Grupa : 2242A

**Profesor coordonator :
Ş.I. Dr. Ing. Stancu Alexandru**

Promoția 2005-2009

Prefață

Lucrarea de față realizează descrierea unei implementări originale a controlului tolerant la defecte pentru roboți mobili. Pornind de la proiectarea și implementarea unei structuri minimale de robot mobil diferențial s-a dezvoltat apoi o aplicație de control în timp real, care se bazează pe sinteza unui controller Sliding Mode pentru operarea robotului în regim de trajectory tracking. Toleranța la defecte a fost implementată sub forma unui banc de filtre Kalman extinse care prin proprietățile lor specifice au dat rezultate bune în ceea ce privește detecția și identificarea defectelor, marcate prin variația parametrilor sistemului. Motivația utilizării acestor instrumente se bazează pe caracteristicile de robustețe, eficiență și complexitate relativ redusă la implementare. Pe parcursul lucrării vor fi surprinse aspecte descriptive formale și analize ale rezultatelor obținute în diferite contexte de operare ale robotului.

„Am văzut mai departe decât alții pentru că m-am ridicat pe umeri de giganți” și pentru că am avut alături oameni extraordinari care să mă susțină și să mă înțeleagă. Mulțumesc familiei mele, prietenilor și colegilor mei și nu în ultimul rând profesorilor mei.

Iulie 2009

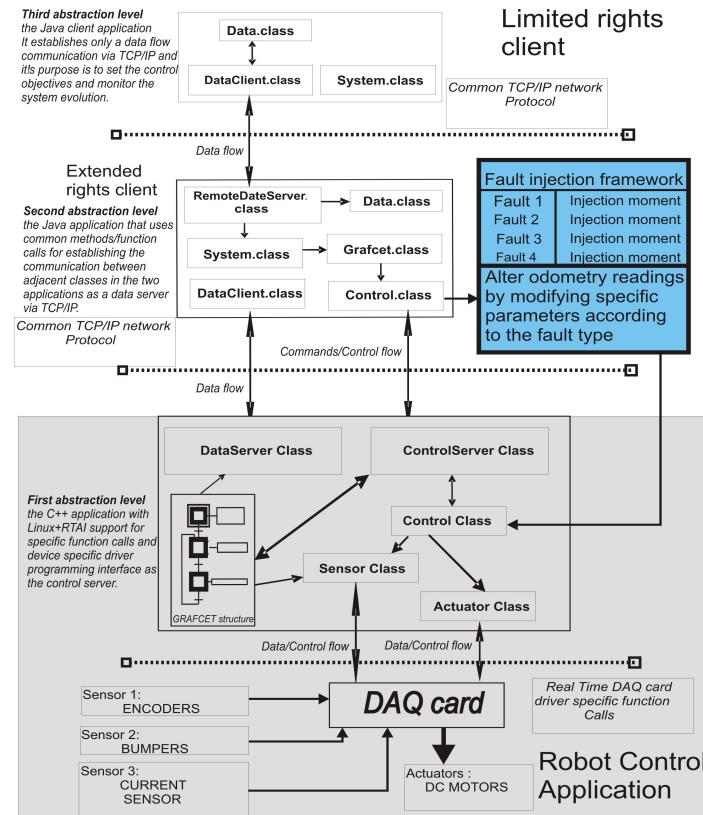
Bachelor thesis report

Mobile Robot Fault Tolerant Control. A fault tolerant real time mobile robot control software application.

Today's trends in control engineering and robotics are blending gradually into a slightly challenging area, the development of fault tolerant real-time applications. Hence, applications should timely deliver synchronized data-sets, minimize latency in their response and meet their performance specifications in the presence of disturbances. The fault tolerant behavior in mobile robots refers to the possibility to autonomously detect and identify faults as well as the capability to continue operating after a fault occurred. This paper introduces a real-time distributed control application with fault tolerance capabilities for differential wheeled mobile robots.

Introduction. System and application overview

By accepting the challenge to ensure fault tolerant real-time control of a self made wheeled differential mobile robot I have been able to develop a hierarchical software application that minimizes costs and supports extensibility. Analytical software redundancy is implemented rather than hardware redundancy for fault tolerance. Following a brief application architecture overview is given so that one can get a proper image on the implementation specific. Note that during this presentation I shall focus only on the specific levels of the application directly involved in the fault tolerance aspects and neglecting the description of the other levels. The main application is designed on a distributed, client-server model, based on TCP/IP wireless communication. Next, the main layers of the server application are described with direct connection to the mobile robot's features. The application's lowest level is based on the interface with the actuators and sensors found in the robot's structure. The next level is the control and fault tolerance level, to the extent that the control algorithm is implemented here and loops concurrently with the monitoring and fault tolerance task. The Grafcet was chosen to be the tool to support various control algorithm implementations and ensures proper serial / parallel execution of the control and monitoring tasks. The fault tolerant module is based on an Extended Kalman Filter bank used to estimate the current robot position and is using a residual computation to determine if a fault appeared in the system. The fault tolerant module is comprised of a fault detection sub-module, a fault identification sub-module and a control reconfiguration sub-module (future work). The third level is a responsible with the communication task, to the extent that it implements a data server and a control server to link over a wireless network to the other nodes in the distributed architecture. Although not presented here, the client application was designed to meet some specific requirements. The first type of client is responsible of interacting with the robot operation, basic start, stop, pause actions of the robot, but also with monitoring the status of the robot by receiving specific packets with sensor and actuator data and also hosts the fault injection framework. The second type of client is a limited only to access data logs from the robot, currently not used. In the next figure one can get a proper overview over the whole distributed application architecture represented at a higher conceptual level.



Next an in depth description of the application is given with focus on the fault tolerant control implementation.

Base level description

As mentioned earlier the base level handles the real time I/O operations issued by the core application to the robot's sensors and actuators. The real time capabilities were implemented by installing the Real Time Application Interface (RTAI), over a standard Linux kernel (currently a RedHat9, 2.4.24 kernel). The RTAI kernel uses an enhanced process scheduler that has an alternate priority system and by using specific algorithms minimizes the latency during I/O operations. The main I/O system is a PCI based DAQ card, namely the NI-6024E that interfaces with the RTOS by using a specific device driver interface, Control and Measurement Data Acquisition Interface (COMEDI), that fits over the real time kernel and manages the accesses to the robot's hardware. By ensuring real time I/O the application minimizes the jitter between the moment the robot controller computes the control signal and the moment the signal is issued by the data acquisition board. Currently the DAQ board outputs a continuous voltage signal to the H-bridge power driver, who transforms it into a PWM signal and send it to the two robot's actuators, two 12V DC motors. The feedback is ensured by a set of sensors. As mentioned earlier the robot was designed with a minimal structure, so the sensor network on the robot is comprised of two incremental encoders with 500 PPR, two bumpers (front, back) and two current sensors for DC motor monitoring. The robot's current processing unit is an Intel Celeron mini-ITX based board. The specific RTAI and Comedi function calls are integrated throughout the application's upper level functions to gain direct access to the base level hardware. Next the control and fault tolerant module is presented.

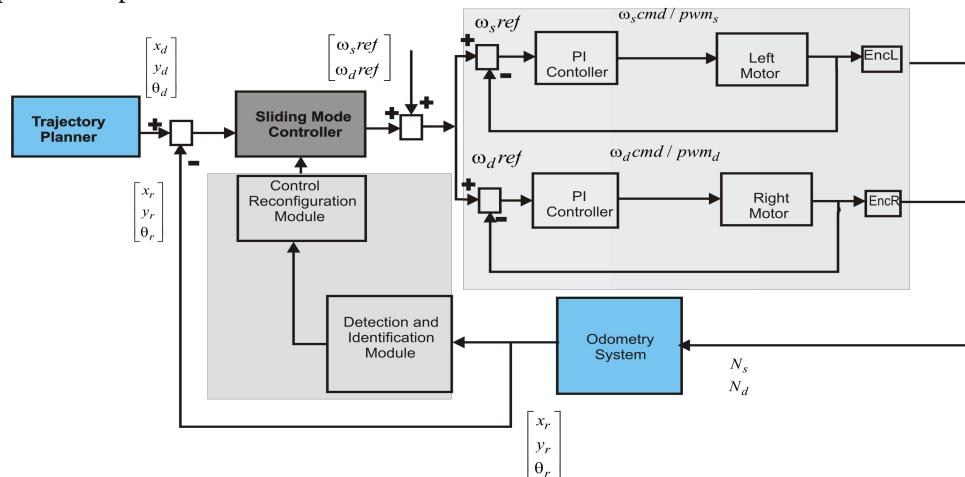
Description of the robot controller and fault tolerant module level

The core of the application is the control and diagnosis level and throughout this short presentation I shall focus only on this aspect. By using the Grafset, as a specific tool that offers support for parallel or serial execution of both control and fault detection and identification, the application gained flexibility and extensibility. The current implementation uses a cascade control loop, to the extent that there are 2 inner PI (Proportional Integral) control loops for the two DC motors running at 20Hz, and an external loop that uses a Sliding Mode controller, with the kinematic model running at 5Hz in order to control the robot's position in trajectory tracking operation. Next some specific details about the design of the Sliding Mode controller are given to mark its particular features in mobile robot trajectory tracking. I've considered the kinematic model equations for the differential mobile robot by accounting that the geometric centre and the rotation centre are not identical. The error vector for trajectory tracking can be obtained by knowing the virtual robot position given by the trajectory planner. It is supposed that the desired trajectory for the mobile robot is pre-specified by a trajectory planner, comprised of speed and acceleration references for the robot. The problem was to design a robust controller so that the robot tracks the desired trajectory under disturbances. The selected sliding surfaces that are marking in fact the desired dynamics are ensuring longitudinal and lateral and heading error convergence.

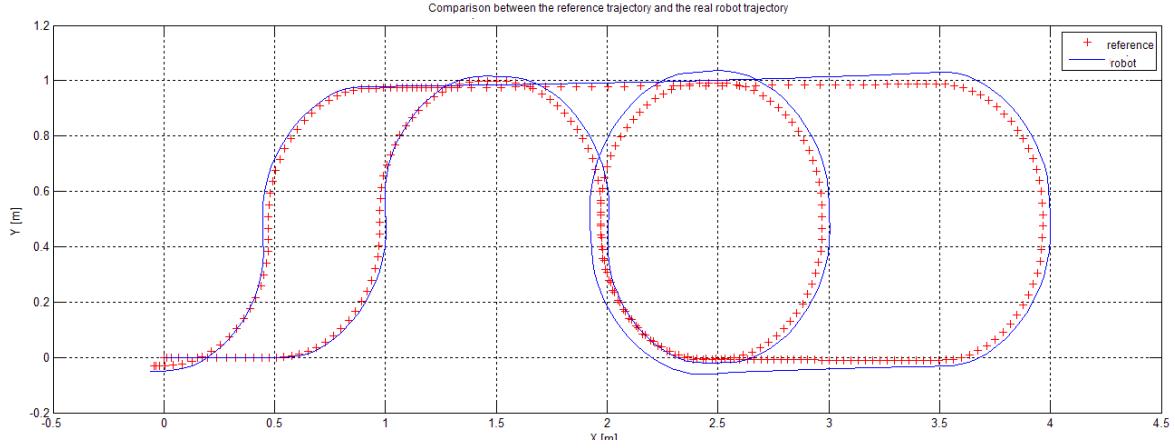
The implemented control law is:

$$\begin{aligned} \dot{v}_r &= \frac{-Q_1 s_1 - P_1 \operatorname{sgn}(s_1) - k_1 \dot{x}_e - \dot{\omega}_d y_e - \omega_d \dot{y}_e + v_r \dot{\theta}_e \sin \theta_e + \dot{v}_d}{\cos \theta_e} \quad \text{and} \\ \omega_r &= \frac{-Q_2 s_2 - P_2 \operatorname{sgn}(s_2) - k_2 \dot{y}_e - \dot{v}_r \sin \theta_e + \dot{\omega}_d x_e + \omega_d \dot{x}_e + \omega_d}{v_r \cos \theta_e + k_0 \operatorname{sgn}(y_e)} . \end{aligned}$$

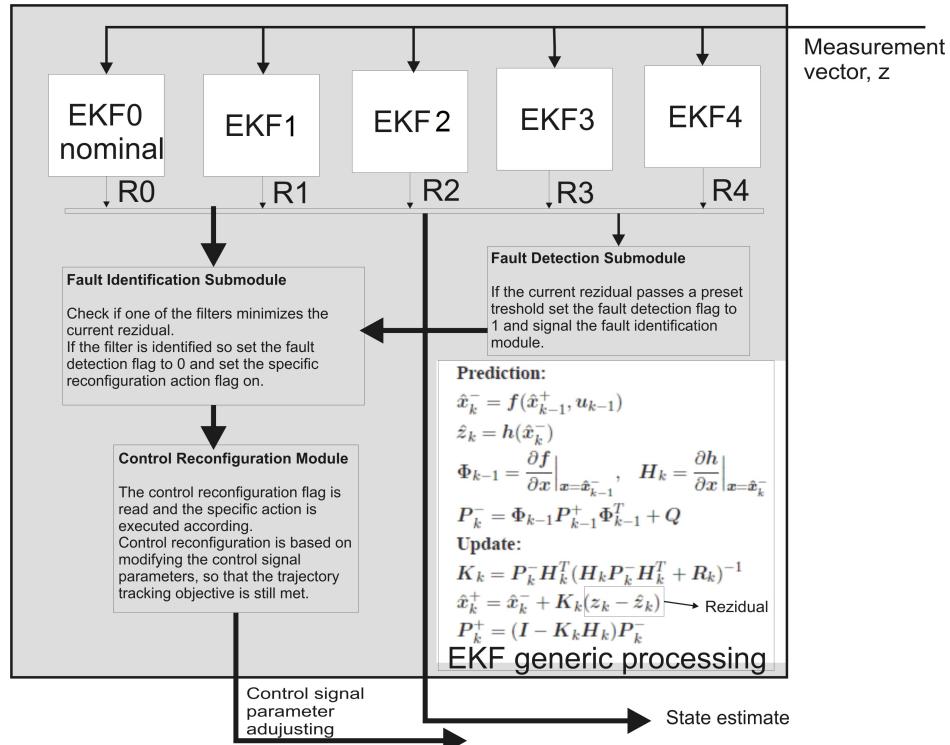
The Q_i and P_i terms were determined such that they ensure good convergence of the state trajectory to the surfaces. The main control loop is next depicted.



The demo trajectory is now depicted and the comparison between the reference and real robot trajectories are emphasized.

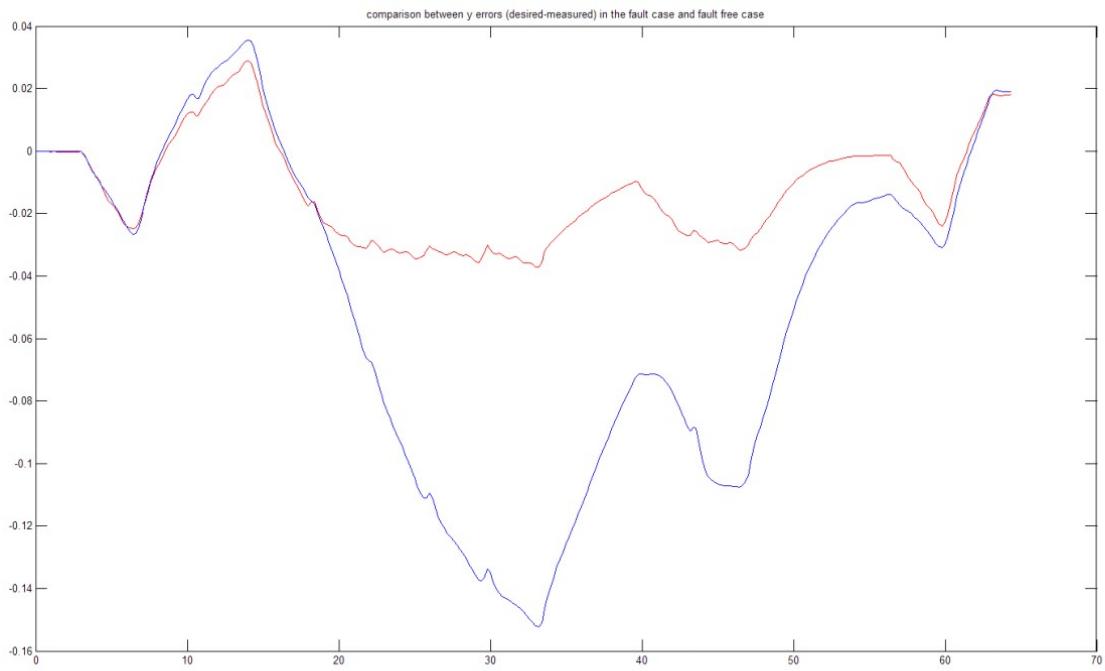
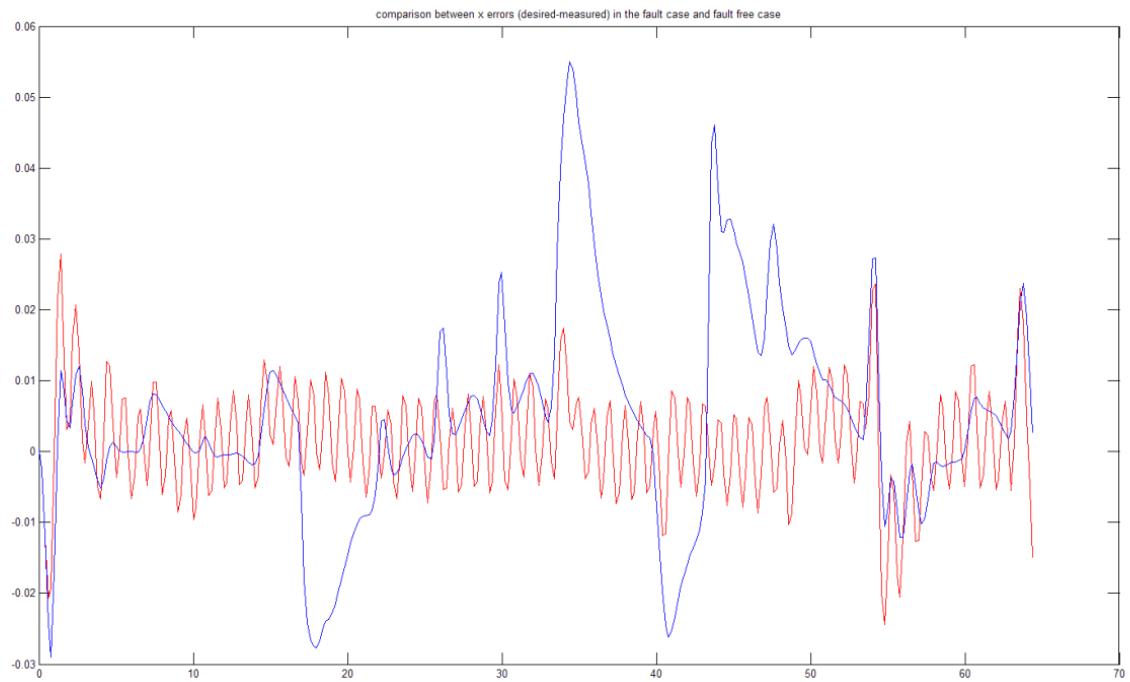


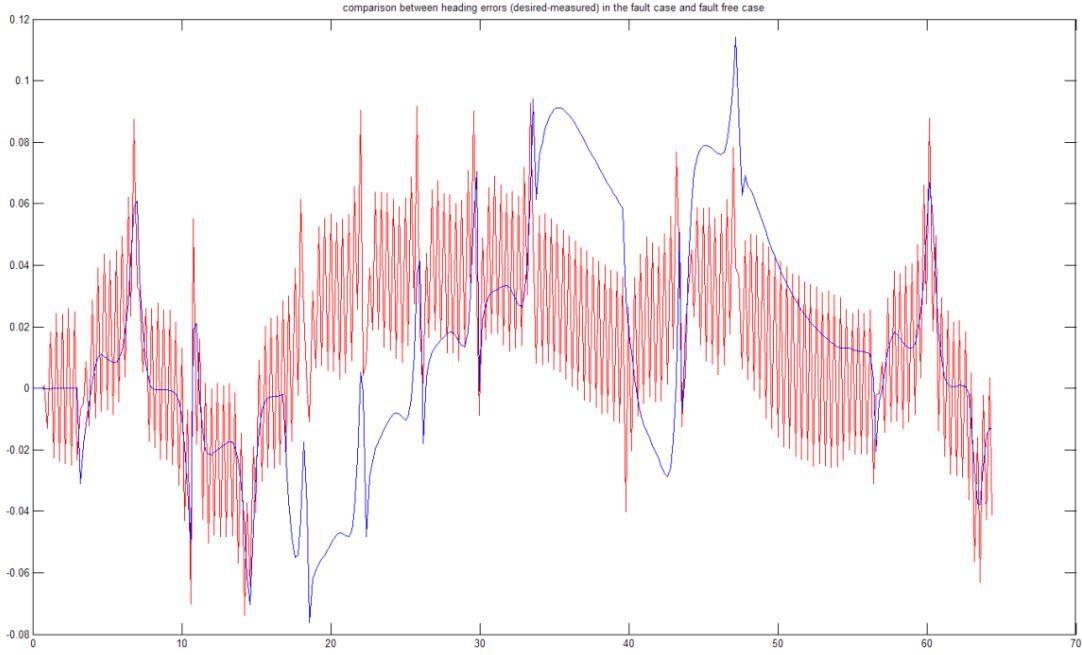
A typical operating context is now described. At each sampling moment (every 50ms) each of the DC motor PI controllers are controlling the motors' speed by tracking the reference set by the Sliding Mode controller (running at 200ms) which is responsible for the accuracy in robot positioning. The odometry system is responsible to output the current robot position by using only the encoder data. Also the odometry system feeds in data for the Fault Detection and Identification Module which is using 5 Extended Kalman Filters used to detect and discriminate between several faults and then issue specific signals to the Control Reconfiguration Module. Each of the 5 Kalman filters embeds in his structure a similar kinematic model of the robot, but with different parameters, to the extent that it shall give a state vector estimate, $[x_{est}, y_{est}, \theta_{est}]^T$ in the context of a specific fault and by comparing it to the measured state vector $[x_m, y_m, \theta_m]^T$, it can decide if a specific fault occurred. The implemented Kalman filters work for a considered fault from the developed benchmark, like the robot wheel radius variation fault and the wheel periodic bump fault. As one can see each EKF computes its own residual by considering the presence of the fault. At each sampling time a current residual is computed by subtracting from the current measurement vector, z , each estimated measurement vector \hat{z} for each filter, and the one minimizing the residual determines the identification of a specific fault. In fact the detection is based on the analysis of the statistical properties of the residual, to the extent that each filter computes a standard sequence of the residual and by using a thresholding method it can effectively decide if a fault occurred. Next the Fault Tolerance subsystem is depicted.



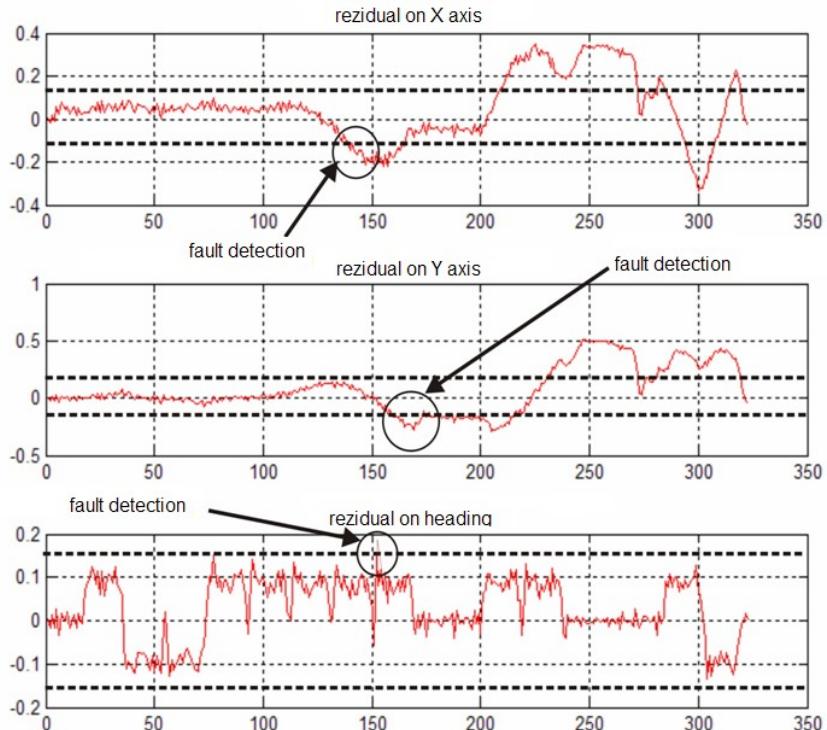
The EKF0 filter embeds a copy of the kinematical model with no modifications. It predicts the nominal behavior of the robot (fault free case). The EKF1 filter embeds a copy of the kinematical model with altered parameters for the first fault (smaller right wheel radius), it shall predict the behavior in the presence of this specific fault and so it generates a specific residual. In a similar way the EKF3, EKF4 filters are embedding the model with altered parameters for the wheel radii to predict the behavior in the presence of a periodical bump of the wheel. After a fault is detected all residuals are compared and the smallest is selected to be the one that is representative for that fault. A minimal residual is obtained in the context that the

measurement vector gives a specific behavior in the presence of the fault and the filter should predict a slightly equal value for the measurement vector based on the altered model. Next a short analysis on the behavior of the system is described in the first fault injected context. So the error vector components (X_e , Y_e , heading error) are described next to emphasize the nominal (fault free) behavior and the faulty one for the demo trajectory.





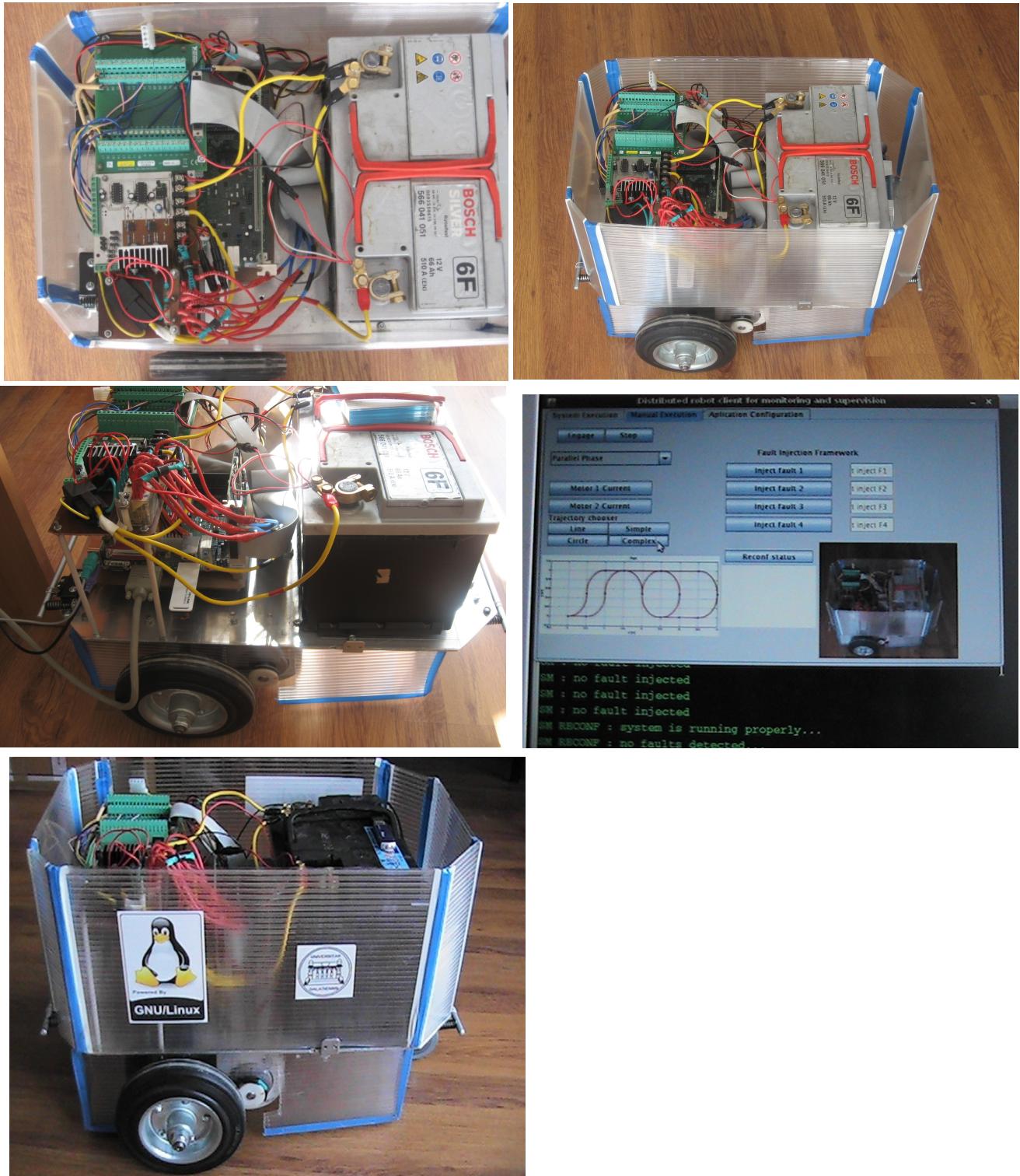
As one can see after injecting the fault the behavior of the robot is altered and it can easily be observed that is relatively easy to discriminate a fault by analyzing the residual. Relatively to the normal behavior the error gets higher values now that the fault was injected and the fault detection module has to signal the appearance of a fault. The residual (generated by the EKF0) is next depicted and as one can see at sample 100 the fault is injected and the residual gets over the preset thresholds and that is the moment when the fault is detected.



The nominal filter generated residual was depicted to extract the proper features of the implemented method. After detecting the fault this residual is compared with the other filters residuals and as it was mentioned earlier the smallest residual in fact emphasizes the fault that occurred because that specific filter predicts a measurement vector that has close values to the real measurement vector affected by the fault. The control reconfiguration is a future work idea and the current application offers full support to implement this new feature. The injection framework is based on simple signaling some events to the extent that for demo purposes the Java client application user can select an injection moment in the robot operation and inject the fault at that specific moment. To overcome the fact that I couldn't implement the faults in hardware I've selected to

implement them in software, and also as presented the fault detection and identification method is based on analytical redundancy. So a specific fault is injected by altering specific odometry information with the modified parameters (wheels radii) in computing linear and angular speed used to determine the robots' pose. So the measurement vector is now reflecting the injected fault behavior as it was if the real (physical) fault occurred.

The presented approach is based on an extensive literature survey regarding Sliding mode implementations for trajectory tracking mobile robot control and fault diagnosis using EKF filters mainly used in flight control or chemical/batch processes. The innovative aspect in this approach is the regarding the flexibility of the software application and the specific mechanisms implemented to gain proper results in controlling and diagnosing the self made differential mobile robot. Some representative images of the robot in all the building stages are next depicted.



Cuprins

1 . Introducere în problematica diagnozei și controlului tolerant la defecte. Enunțarea problemei diagnozei și toleranței la defecte pentru sisteme dinamice.	1.....23
1.1 Defecte și toleranță la defecte. Elemente specifice ale unui sistem tolerant la defecte și arhitecturi pentru sisteme de control tolerant la defecte2
1.2 Un review al abordărilor existente pentru diagnoză și controlul tolerant la defecte6
2. Descrierea sistemului dezvoltat: robot mobil diferențial cu 2 roți motoare și o roată directoare	25.....51
2.1 Aspecte privind proiectarea și realizarea robotului mobil. Subsisteme specifice ale structurii generice de robot mobil diferențial cu 2 roți motoare și o roată directoare25
2.1.1 Subsistemul de locomoție28
2.1.2 Subsistemul de percepție30
2.1.3 Subsistemul de prelucrare a datelor și comunicație32
2.2 Modelarea matematică a robotului mobil cu 2 roți motoare și directoare34
2.2.1 Modelul cinematic al robotului34
2.2.2 Modelul dinamic al robotului39
2.3 Abordări și aspecte legate de metodele de control și operarea roboților mobili43
3. Descrierea aplicației software de control tolerant la defecte	53.....72
3.1 Conducere în timp real. Instrumente utilizate, facilități specifice și analiza performanțelor soluției utilizate53
3.1.1 Utilizarea unui sistem de operare real time bazat pe LinuxOS. Utilizarea Linux-RTAI pe platforma bazată pe Intel P454
3.1.2 Utilizarea setului de drivere pentru operații I/O real time, COMEDI60
3.2 Prezentarea nivelelor aplicației de control tolerant la defecte. Aspecte generale de proiectare a aplicației63

3.2.1 Nivelul de bază, de interfață cu hardware-ul	64
3.2.2 Nivelul de implementare a algoritmilor de conducere, diagnoza și toleranță a defectelor	66
3.2.3 Nivelul superior al comunicației între nodurile sistemului distribuit	70
4. Aspecte specifice ale implementării controlului tolerant la defecte în timp real a robotului mobil	73.....	121
4.1 Nivelul task-ului de control al robotului în regim de trajectory Tracking	73
4.1.1 Sinteză buclei interne pentru reglarea turăției actuatorilor utilizând controllere de tip PI. Analiza perfomanțelor	74
4.1.2 Sinteză buclei externe de conducere pentru poziționarea robotului în regim de trajectory tracking. Sistemul odometric	79
4.1.3 Sinteză controllerului de tip sliding mode pentru poziționarea robotului în regim de trajectory tracking	83
4.1.3.1 Aspecte teoretice și de implementare ale controlului sliding mode. Motivarea alegerii controlului sliding mode utilizând modelul cinematic al robotului	84
4.1.3.2 Studiul performanțelor robotului în buclă închisă pentru trajectory tracking	92
4.2 Nivelul task-ului de monitorizare, diagnoză și toleranță la defecte. Analiza și sinteza sistemului de diagnoză și toleranță a defectelor	96
4.2.1 Analiza tipurilor de defecte ale robotilor mobili	96
4.2.2 Implementarea sistemului de diagnoză și toleranță a defectelor bazat pe filtrul Kalman extins	99
4.2.2.1 Descrierea formală a filtrului Kalman extins și motivarea utilizării în contextul problemei abordate	100

4.2.2.2 Detalii de implementare ale metodei diagnozei multi-model bazată pe utilizarea unui banc de filtre Kalman extinse. Definirea tipurilor de defecte ce pot fi discriminate cu metoda utilizată104
4.2.2.3 Deteția defectelor și sinteza modulului de detecție a defectelor. Identificarea defectelor și sinteza modulului de identificare a defectelor. Reconfigurarea controlului pe baza tipului de defect diagnosticat106
5. Concluzii și direcții viitoare de studiu	123.....124
6. Bibliografie	125.....129
7. Anexe	131.....189

Capitolul 1

Introducere în problematica diagnozei și controlului tolerant la defecte. Enunțarea problemei diagnozei și toleranței la defecte pentru sisteme dinamice.

Introducere în problematica diagnozei și controlului tolerant la defecte.

Enunțarea problemei diagnozei și toleranței la defecte pentru sistemele dinamice

În marea lor majoritate sistemele tehnice sunt supuse defectelor. Defectele în actuatori determină o reducere a performanțelor sistemului de control și pot determina în anumite cazuri chiar căderea sistemului. Citirile eronate de la senzori determină utilizarea unor puncte de operare departe de cele nominale. Uzura reduce eficiența și calitatea operațiilor executate. În multe cazuri în care apar defecte operarea sistemului este sistată pentru a evita distrugerea instalațiilor și pentru a menține siguranța oamenilor. Astfel detectia și managementul defectelor are un rol din ce în ce mai important în sistemele și liniile tehnologice moderne, unde componente autonome interacționează într-un mod complex și unde un defect al unei singure componente ar putea genera o evoluție eronată a întregului sistem.[1]

Societatea modernă depinde de operarea corectă și disponibilitatea sistemelor tehnologice complexe acest lucru reflectându-se de fapt în asigurarea eficienței în producție și eficienței economice. Într-o accepție generală un defect este un eveniment care generează o schimbare în comportamentul și evoluția unui sistem, în sensul că acesta nu va mai urma evoluția impusă pentru atingerea scopului. Defectul se poate concretiza sub forma unui eveniment intern în sistem sau un eveniment extern care să altereze structura sistemului sau parametrii acestuia în sensul că nivelul performanțelor scade și sistemul poate chiar să-și piardă funcționalitatea. Pentru a elmina posibilitatea apariției daunelor în sistem sau chiar periclitarea activității umane defectele trebuie detectate rapid și luarea deciziei trebuie făcută într-un timp mai scurt decât timpul de propagare al efectului defectului prin sistem. Altfel spus sistemul trebuie să devină tolerant la defecte. Sistemic vorbind toleranța la defecte se manifestă prin interacțiunea dintre un anumit sistem și un controller, utilizând o terminologie generică. Controllerul generic nu generează doar legea de comandă ci are rolul de a analiza comportamentul procesului pentru a detecta și identifica defectele și de a ajusta legea de comandă pentru a menține sistemul în buclă închisă într-o regiune de operare cu performanțe acceptabile.[2] Controlul tolerant la defecte presupune că sistemul este afectat de un defect care împiedică atingerea indicilor de performanță impuși la proiectare. Controllerul tolerant la defecte are posibilitatea de a reacționa la apariția unui defect și de a-și ajusta operarea la

evoluția cu defecte a sistemului condus. Într-o viziune generală sinteza unui sistem tolerant la defecte presupune diagnoza defectelor, ce implică detecția și identificarea defectelor precum și reconfigurarea legii de comandă, pentru a asigura funcționarea sistemului spre atingerea indicilor de performanță. De regulă acești pași nu sunt realizați de un controller ușor ci de către un sistem de supervizare care prescrie structura legii de comandă și selectează algoritmul și parametrii controllerului. Metodele tradiționale pentru diagnoza defectelor presupun că pentru detecția unui defect este necesar ca un semnal măsurat să depășească un anumit prag sau să-și modifice proprietățile spectrale. În sistemele complexe însă acestă abordare nu este mereu cea mai potrivită deoarece nu putem măsura toate mărurile din sistem sau măsura este o opțiune foarte scumpă. Această abordare se bazează de fapt pe redundanță fizică, care presupune că acele componente importante ale sistemului sunt dublate. A doua abordare, pe care se bazează și lucrarea de față se referă la redundanță analitică, care utilizează elemente de modelare matematică și relații analitice împreună cu informație redusă de măsură pentru a îndeplini detecția și identificarea defectelor.[1] De-a lungul lucrării se va considera doar analiza și specificul unei implementări a redundanței analitice pentru controlul tolerant la defecte bazat pe model.

1.1 Defecte și toleranță la defecte. Elemente generice specifice ale unui sistem tolerant la defecte și arhitecturi pentru sisteme de control tolerant la defecte

Defectul într-un sistem dinamic este o deviație a structurii sistemului sau parametrilor sistemului de la valorile nominale. Schimbările structurale se pot manifesta sub forma blocajelor actuatorilor, pierderea informației de la senzori sau deconectarea unei componente a sistemului, context ce determină o modificare a interacțiunii componentelor sistemului și modificarea interfeței între sistem și controller. Toate tipurile de defecte ce pot apărea într-un sistem determină deviații ale proprietăților de I/E dinamice ale sistemului de la cele nominale și deci alterează performanțele în buclă închisă ale sistemului. În continuare este analizat impactul defectelor ce pot apărea într-un sistem evidențiind astfel specificul sistemelor tolerate la defecte. Se va realiza o descriere formală pentru a putea distinge elementele de interes utile mai târziu în descrierea implementării propuse pentru sistemul robotic dezvoltat. Considerăm în figura 1 structura generică a unui sistem tolerant la defecte, subliniind la nivel formal modul de evoluție al sistemului în prezența defectelor.[2]

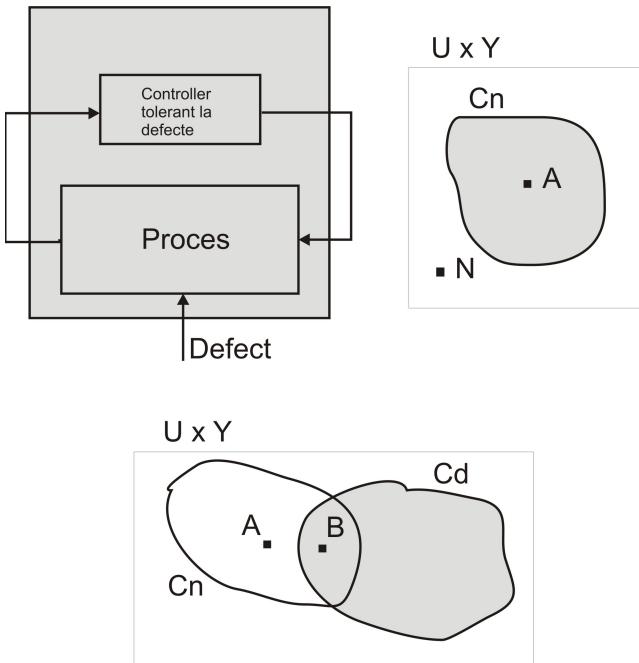


Figura 1 Structura generică a unui sistem tolerant la defecte

Se analizează impactul apariției unui defect în sistem din punctul de vedere al controllerului. Defectele ce apar în structură pot fi separate în seturi, pe care le vom nota D , care pun în evidență de fapt acele evenimente care vor genera alterarea evoluției sistemului pentru atingerea scopurilor impuse. Pentru analiza performanțelor sistemului este importantă analiza perechilor intrare ieșire (u,y) , unde $u(t)$ - vectorul de intrare, $y(t)$ - vectorul de ieșire, care definesc comportamentul sistemului, notat C_n . În general comportamentul sistemului este un subset al spațiului $U \times Y$, al combinațiilor posibile ale semnalelor de intrare și ieșire. Punctul A din interiorul domeniului C_n descrie acele perechi I/E care sunt acceptate în evoluția sistemului, iar punctul N marchează acele perechi care nu sunt consistente cu dinamica sistemului. Apariția unui defect generează o modificare în comportamentul sistemului, după cum se poate observa, comportamentul sistemului înaintează către domeniul de operare cu defecte, notat C_d . Astfel dacă se aplică o intrare u sistemului fără defect și sistemului cu defect vom obține două ieșiri y_A și y_B care se află în domenii separate, făcând detecția și izolarea defectului posibilă, excludând cazul în care perechea de I/E se află la intersecția domeniilor C_n și C_d . Pentru controlul tolerant la defecte modelele dinamice trebuie să descrie sistemul la apariția defectelor. Modelele reprezintă de fapt constrângeri asupra semnalelor de intrare U și a celor de ieșire Y care apar în evoluția sistemului. Depinzând de tipul sistemului

considerat constrângerile pot lua forma unor ecuații algebrice, ecuații diferențiale sau cu diferențe sau automate finite și constituie de fapt un model care poate fi utilizat ca generator al comportamentului sistemului. Astfel pentru o anumită intrare u sistemul răspunde cu ieșirea y , la apariția unui defect se va putea pune în evidență cum este y afectat. Un aspect important în diagnoză îl reprezintă faptul că aceste constrângerile pot fi utilizate pentru a verifica consistența perechilor I/E măsurate. Comportamentul sistemului mai poate fi modificat și prin acțiunea perturbațiilor și a incertitudinilor de model. Astfel putem avea defecte aditive, care se manifestă printr-o reprezentare ca intrare necunoscută adunată termenilor din modelul sistemului. Defectele multiplicative se manifestă printr-o dependență a parametrilor sistemului față de amplitudinea defectului. În continuare este redată abordarea pentru o separare a defectelor de perturbații și incertitudinile de model. Defectele sunt acele situații care trebuie detectate și a căror efecte trebuie contracarurate prin măsuri de remediere explicite. Perturbațiile și incertitudinile de model sunt reprezentate prin zgomote, a căror existență este cunoscută și care sunt eliminate prin filtrări sau o proiectare robustă. Defectele pot apărea la fiecare nivel al buclei sistemului de conducere, defecte ale procesului (care determină modificarea proprietăților dinamice de I/E ale sistemului), defecte ale senzorilor (care nu determină alterarea caracteristicilor sistemului, însă citirile de la senzori au erori foarte mari) și defecte ale actuatorilor (care nu determină o alterare a caracteristicilor sistemului însă determină întreruperea sau modificarea influenței controllerului asupra sistemului). Înainte de a trece la studiul elementelor specifice ale unui sistem tolerant la defecte sunt amintite câteva dintre caracteristicile care ar trebui luate în considerare la proiectarea unui sistem tolerant la defecte. Astfel siguranța, fiabilitatea, disponibilitatea și dependabilitatea toate asigură reușita în cazul cazul proiectării unui sistem tolerant la defecte menținând și aspectul performanței în prim-plan. În figura următoare este redată structura concretă a unui sistem tolerant la defecte.[1]

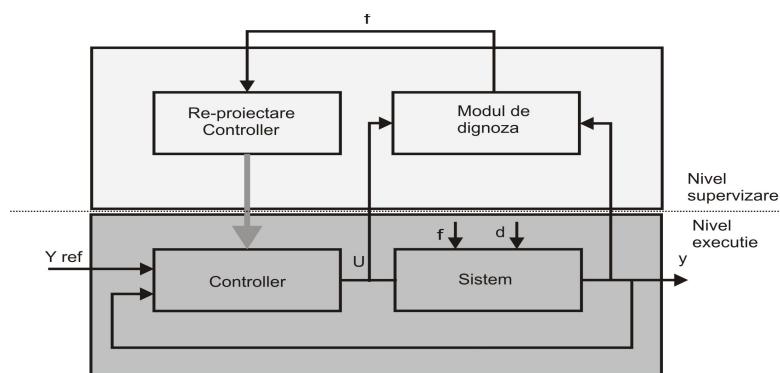


Figura 2 Structura concretă a unui sistem tolerant la defecte

În arhitectura concretă blocul de diagnostic are rolul de a testa consistența măsurătorilor I/E cu modelul procesului. Ieșirea modulului de diagnoză este o caracterizare a defectului ce conține suficientă informație pentru a realiza re-proiectarea controllerului. Blocul de reproiectare a controllerului utilizează informația primită și ajustează controllerul pentru contextul de operare în prezența defectului. Din figură putem observa că în cazul controlului tolerant la defecte se realizează o extensie a controllerului tipic pentru bucla de reglare prin adăugarea unui nivel de supervizare ce include blocurile de diagnoză și re-proiectare a controllerului. În cazul operării fără defecte controllerul nominal reiectează perturbațiile și asigură urmărirea referinței, precum și alte cerințe ale sistemului în buclă închisă. Acțiunile principale de control au loc la nivelul execuție, la nivelul de supervizare blocul de diagnoză recunoaște comportamentul sistemului în buclă închisă ca fiind fără defecte și astfel nu este necesară nici o modificare a legii de comandă. Dacă apare un defect f, blocul de diagnostic detectează defectul, îl identifică și blocul de re-proiectare a controllerului ajustează controllerul pentru noul context de operare. Un aspect interesant care a fost observat și în cazul dezvoltării propriului sistem constă în faptul că până la un anumit nivel toleranța la defecte poate fi atinsă și fără o structură superioară de supervizare, doar prin alegerea unor metode de control, la nivelul execuție, adecvate, întradevăr doar pentru o clasă redusă de defecte. O primă soluție vine din direcția controlului robust, care presupune sinteza unui controller care tolerează schimbările în dinamica procesului, acesta atingându-și toate cerințele de performanță impuse în prezența defectului. Astfel, toleranța la defecte este obținută fără a modifica parametrii controllerului, abordarea fiind generic intitulată toleranță pasivă la defecte. În cazul în care parametrii controllerului sunt ajustați pentru a se adapta variațiilor parametrilor procesului, datorate unui defect, avem abordarea generic intitulată toleranță activă a defectelor, fiind concret implementată prin structurile de control adaptiv. În cazul controlului tolerant la defecte localizarea și amplitudinea defectului trebuie determinate, conturându-se astfel niște etape generice ale diagnozei. Astfel prima etapă constă în detecția defectului, se decide dacă a apărut sau nu un defect și putem determina astfel momentul de timp la care sistemul este suspus unui defect. A doua etapă este izolarea defectului, care presupune determinarea componentei în care a apărut defectul, realizându-se astfel o localizare a defectului. A treia etapă este identificarea defectului care presupune determinarea tipului de defect și amplitudinea lui. Ultima etapă este reconfigurarea controlului, în sensul ajustării legii de comandă pentru a asigura operarea sistemului în prezența defectului. În mod uzual etapele prezentate anterior se întrepătrund sau se combină aparând doar etapele de

detectie, identificare și reconfigurare. Înainte de a trece la prezentarea metodelor cunoscute pentru control tolerant la defecte se impune prezentarea unor aspecte practice utile pentru dezvoltarea sistemelor tolerante la defecte. Toleranța la defecte necesită redundanță, fie ea analitică, fie fizică, fiind necesar un studiu prealabil pentru a decide alocarea optimă a structurii senzorilor și numarul lor pentru a minimiza costurile. Degradarea performanțelor este inevitabilă într-un sistem în care apar defecte însă se face un compromis în sensul că cu cât sunt mai scăzute cerințele de performanță cu atât crește capacitatea de tolerare a mai multor defecte. [3]

În continuare sunt prezentate câteva arhitecturi specifice de sisteme de control tolerant la defecte. Arhitectura unui sistem de control tolerant la defecte descrie în genere care componente ale sistemului condus, controllerului și sistemului de diagnoză interacționează și care este informația schimbată între aceste componente. Astfel puteam avea diagnoză distribuită, în care sistemul de diagnostic este proiectat ca o entitate unică și algoritmul de diagnostic este distribuit între diferite componente pentru a distribui efortul computațional. Diagnoza descentralizată presupune descompunerea problemei diagnozei în subprobleme care se referă la subsisteme ale sistemului considerat, problemele fiind rezolvate independent. Diagnoza coordonată presupune la rândul ei o descompunere, însă soluțiile obținute independent sunt combinate de un coordonator pentru a asigura consistența. În cele ce urmează este prezentat cadrul general de studiu al problemei abordate în lucrarea de față precum și nivelul actual al rezultatelor obținute.

1.2 Un review al abordărilor cunoscute în diagnoză și controlul tolerant la defecte

În continuare este prezentat un review al celor mai cunoscute abordări dezvoltate în diagnoză și controlul tolerant la defecte utilizând o împărțire în trei mari categorii de metode și anume, metode cantitative bazate pe model, metode calitative bazate pe model și strategii de căutare și metode bazate pe istoricul procesului. Înainte de a trece la descrierea particulară a fiecărei metodologii se impune prezentarea clasificării metodelor de diagnoză din figura 3. [1]

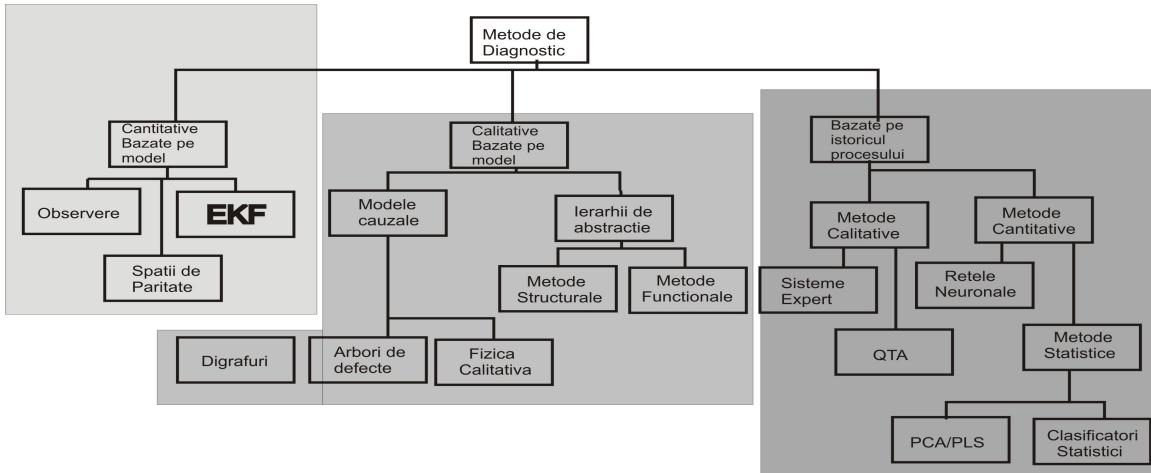


Figura 3 Metodologii pentru diagnoza defectelor

În secțiunea următoare sunt descrise metodele de diagnoză a defectelor bazate pe model. Bazându-se pe un model explicit al sistemului monitorizat metodele cantitative bazate pe model presupun execuția a doi pași.[2] Primul pas determină inconsistențele între comportamentul real și cel dorit, care se concretizează de fapt ca fiind reziduurile (semnale artificiale ce reflectă potențialele defecte ale sistemului). Al doilea pas presupune alegerea regulii de decizie pentru diagnoză. Redundanța analitică este obținută din dependențele funcționale între variabilele procesului și sunt redate de obicei sub forma unui set de relații algebrice sau temporale între stările, intrările și ieșirile sistemului. Redundanța analitică directă este obținută prin utilizarea unor relații algebrice între diferite măsurători de la senzori. Valoarea obținută este comparată cu măsurătoarea de la alt senzor pentru aceeași mărime și o diferență va indica apariția unui defect de senzor. Redundanța analitică temporală este obținută din relații diferențiale sau cu diferențe între diverse ieșiri ale senzorilor și intrări ale actuatorilor. Utilizând informația I/E a procesului redundanța temporală e utilă pentru detecția defectelor în senzori sau actuatori. Ideea de bază a redundanței analitice pentru diagnoză este de a verifica consistența comportamentului actual al sistemului cu modelul sistemului. Un aspect important se referă la faptul că schemele de redundanță analitică pentru diagnoza defectelor sistemelor dinamice sunt de fapt metode de procesare a semnalului care utilizează estimarea stării, estimare parametrică sau filtrarea adaptivă. Fie că se utilizează un model în reprezentare de stare, fie un model intrare – ieșire sistemul poate fi reprezentat prin ecuația 1,

$$y(t) = f(u(t), \omega(t), \theta(t)) \quad (1),$$

unde $y(t)$, $u(t)$ denotă ieșirile și respectiv intrările măsurabile, $x(t)$ și $\omega(t)$ reprezintă mărimile de stare și perturbațiile în general nemăsurabile și θ parametrii procesului. Concret pentru a construi contextul de analiză pornim de la reprezentările pe stare și I/E a unui sistem în timp discret, cu intrarea $u(t)$ -vector m-dimensional, ieșirea $y(t)$ -vector k-dimensional și $x(t)$ vectorul de stare n-dimensional, redată în ecuațiile următoare,

$$\begin{aligned}x(t+1) &= Ax(t) + Bu(t) \\y(t) &= Cx(t) + Du(t)\end{aligned}$$

și modelul I/E $H(z)y(t) = G(z)u(t)$, (2). Matricile $H(z)$ și $G(z)$ sunt matrici polynomiale în z^{-1} având forma $H(z) = I + H_1z^{-1} + H_2z^{-2} + \dots + H_nz^{-n}$, $G(z) = G_0 + G_1z^{-1} + G_2z^{-2} + \dots + G_nz^{-n}$. Cele două reprezentări de mai sus redau procesul în situația ideală unde nu sunt prezente defecte și alte perturbații sau zgromot. Defectele în reprezentarea structurală pot fi descrise prin următoarele ecuații,

$$\begin{aligned}x(t+1) &= Ax(t) + Bu(t) + Ep(t) \\y(t) &= Cx(t) + Du(t) + E'p(t) + q(t)\end{aligned}, (3)$$

în care $u(t)$ și $y(t)$ sunt observabile, $p(t)$ marchează defectele actuatorilor, anumite defecte ale procesului, perturbațiile și defecte de intrare pentru senzori, $q(t)$ reprezintă defectele de ieșire al senzorilor. Pe de altă parte apariția defectelor determină o modificare și în modelul I/E determinată și de tipul de defect (multiplicativ sau aditiv),

$$\begin{aligned}H(z)y(t) &= G(z)u(t) + H(z)q(t) + F(z)p(t), \text{ generic} \\(H(z) + \Delta H(z))y(t) &= (G(z) + \Delta G(z))u(t), \text{ multiplicative} \\H(z)y(t) &= G(z)u(t) + H(z)q(t) + F(z)p(t) + K(z)\omega(t), \text{ additive}\end{aligned}, (4)$$

unde $q(t)$ sunt defectele de senzori, $p(t)$ sunt defectele actuatorilor și $\omega(t)$ vectorul perturbațiilor. Defectele determină în general modificări ale variabilelor de stare sau modificări ale parametrilor procesului. Pornind de la modelul procesului se pot estima valorile pentru semnalele nemăsurabile utilizând mărimile măsurabile utilizând metode de estimare parametrică sau metode de estimare a stării. Filtrele Kalman sau observerele sunt des utilizate pentru estimarea stării. Metoda celor mai mici pătrate este o metodă puternică prin monitorizarea online a estimărilor parametrilor. Cele mai recente tehnici se bazează pe ecuații

de paritate pentru generarea reziduurilor, ecuații obținute prin rearanjarea sau transformarea modelelor I/E, care se pot genera on-line relativ ușor din informații de la proces. În ceea ce privește utilizarea observerelor pentru diagnoza sistemelor dinamice se dorește generarea unui set de reziduuri pentru detecția și identificarea individuală a defectelor din clase diferite. Reziduurile generate trebuie să fie robuste în sensul că deciziile nu trebuie să fie alterate de zgomotul de proces sau de măsură sau de incertitudinile de model.[5] Metoda dezvoltă un set de observare, fiecare dintre ele fiind sensibil la un subset de defecte și insensibil la celelalte subseturi de defecte și la intrări necunoscute. Gradele de libertate care rezultă din măsurători și redundanța modelului asigură condițiile de implementare ale acestei abordări. Ideea de bază este că în cazul fără defecte observerele vor urmări cu precizie procesul, reziduurile având valori mici. La apariția unui defect observerele care sunt insensibile la defectul apărut mențin același reziduu mic, în schimb observerele care sunt sensibile la defect vor devia de la urmărirea procesului și vor genera valori mari ale reziduurilor. Observerele se proiectează astfel încât fiecare reziduu să aibă o alură diferită pentru fiecare defect, acest aspect făcând relativ ușoară etapa de identificare a defectului, fiecare observer radând o semnatură specifică a defectului.[4] Pentru a reda modalitatea de sinteză a unui observer utilizat pentru detecția defectelor se pornește de la un sistem discret în reprezentare structurală, redat în ecuațiile 5

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + Ed(t) + Fp(t) \\ y(t) &= Cx(t) \end{aligned} \quad , (5)$$

unde $d(t)$ este vectorul intrărilor necunoscute. Observerul este un model care poate fi reprezentat prin setul de ecuații 6

$$\begin{aligned} x_0(t) &= Tx(t) \\ x_0(t+1) &= Hx_0(t) + Ju(t) + Gy(t) \end{aligned} \quad , (6)$$

Ideea de bază este de a utiliza un algoritm dinamic pentru a estima variabilele de stare din valorile observate ale intrării și ieșirii, concret sinteza observerului rezumându-se la alegerea matricilor T , H , J și G . [6] Dacă considerăm eroarea de estimare la momentul $t+1$ ca fiind $e(t+1)$ și reziduul fiind $r(t)$ putem scrie ecuațiile, (7)

$$\begin{aligned} e(t+1) &= x_0(t+1) - Tx(t+1) \\ r(t) &= L_1x_0(t) + L_2y(t) \end{aligned} \quad (7)$$

și făcând înlocuirile putem rescrie eroarea de estimare a stării sub forma ecuației 8,

$$e(t+1) = Hx_0(t) + (J - TB)u(t) + Gy(t) - TAx(t) - TEd(t) - TFp(t) \quad (8)$$

Pentru a determina urmărirea sistemului de către observer independent de vectorul intrărilor necunoscute, $d(t)$, trebuie să alegem o matrice T , astfel încât $TE = 0$. Urmărirea nu este afectată de vectorul de intrare $u(t)$ dacă matricea J este aleasă astfel încât $J = TB$. Făcând substituțiile necesare obținem ecuația 9,

$$e(t+1) = Hx_0(t) + (GC - TX)x(t) - TFp(t) \quad (9)$$

Se alege matricea G astfel încât $GC - TA = -HT$, unde H este stabilă și $L_1T + L_2C = 0$. Astfel eroarea de estimare și reziduul iau forma descrisă în ecuațiile 10,

$$\begin{aligned} e(t+1) &= He(t) - TFp(t) \\ r(t) &= L_1e(t) \end{aligned} \quad (10)$$

Pentru cazul fără defect eroarea este redată doar de ecuația 11,

$$e(t+1) = He(t) \quad (11).$$

În cazul în care valorile proprii ale lui H sunt mai mici decât 1 atunci $e(t) \rightarrow 0$ pentru $t \rightarrow \infty$ și ca rezultat direct în cazul în care nu avem defect, eroarea de estimare a stării și reziduul vor urmări procesul indiferent de valoarea vectorului de intrări necunoscute, $d(t)$, fapt care îi conferă denumirea de observer cu intrări necunoscute. Astfel la apriția unui defect ieșirea descrisă în ecuația (5) se modifică, la fel și eroarea de estimare și reziduul marcând signatura defectului,

$$\begin{aligned} y(t) &= Cx(t) + q(t) \\ e(t+1) &= He(t) + Gq(t) \\ r(t) &= L_1e(t) + L_2q(t) \end{aligned} \quad ,(12).$$

În secțiunea precedentă a fost prezentată modalitatea de proiectare și sinteză a observerelor pentru diagnostic pentru cazul unui sistem liniar, în continuare fiind redați pașii majori pentru generarea unui observer pentru diagnoză pentru sisteme neliniare. Se utilizează un model liniarizat discret în formă affine, descris de ecuațiile 13,

$$\begin{aligned} x(t+1) &= Ax(t) + B(y(t), u(t)) + Ed(t) + F(x(t))p(t) \\ y(t) &= Cx(t) + K(x(t))p(t) \end{aligned} \quad (13)$$

Pentru a proiecta un observer neliniar se utilizează aceeași metodologie ca și în cazul liniar și astfel vom avea ecuațiile 14, care determină structura observerului,

$$\begin{aligned} x_0(t) &= Tx(t) \\ x_0(t+1) &= Hx_0(t) + J(y(t), u(t)) + Gy(t) \\ r(t) &= L_1x_0(t) + L_2y(t) \end{aligned} \quad (14)$$

Condițiile care se impun sunt $TA - HT = GC$, $TE = 0$ și $L_1T + L_2C = 0$. Înainte de a trece la descrierea unei a doua metode cantitative bazată pe model se impune prezentarea câtorva aspecte legate de evaluarea reziduurilor. Principala prelucrare a unui observer este concentrată în generarea reziduurilor cu proprietăți de decuplare satisfăcătoare. Pentru a realiza evaluarea reziduurilor se pot folosi metode bazate pe thresholding (utilizarea unor praguri superioare și inferioare pentru analiza variației semnalului), clasificatoare statistice sau rețele neuronale. [5]

În continuare este prezentată metoda relațiilor de paritate pentru diagnoza defectelor în sistemele dinamice. Ecuațiile de paritate sunt la bază de fapt o transformare a ecuațiilor modelelor în reprezentare de stare sau I/E. Ideea de bază este de a testa paritatea (consistența) modelelor procesului cu măsurătorile de la senzori și intrările cunoscute ale procesului. În condițiile operării în regimul staționar ideal, reziduul sau valoarea ecuațiilor de paritate este 0.[8] În realitate reziduurile nu sunt 0 datorită faptului că avem zgromot de măsură și de proces, incertitudini de model, erori în senzori și actuatori și defecte în proces. Ideea este de a rearanja structura modelului pentru a obține cea mai bună izolare a defectelor. În cadrul acestei metode redundanța oferă libertatea de a alege modul de proiectare al ecuațiilor pentru generarea reziduurilor pentru a putea obține ulterior izolarea defectelor. Izolarea defectelor presupune posibilitatea de a genera reziduuri care sunt ortogonale cu cele pentru alte defecte.[9] Formal, metoda ecuațiilor de paritate poate fi descrisă pornind de la ecuația de ieșire al modelului în reprezentare de stare, considerând y vectorul n-dimensional al măsurătorilor, x vectorul m-dimensional al valorilor reale ale variabilelor de stare. Redundanța există dacă $n > m$. În condițiile operării fără defecte avem, $y(t) = Cx(t)$ și în cazul în care apare un defect modelul devine $y(t) = Cx(t) + \Delta y(t)$. Se alege matricea de

proiecție V (n-m)xn-dimensională ce satisfacă $VC = 0$ și $V^T V = I_n - C(C^T C)^{-1} C^T$. Dacă combinăm observațiile y într-un vector de paritate p obținem,

$$p(t) = Vy(t) = VCx(t) + V\Delta y(t) = V\Delta y(t) \quad , (15)$$

$p(t) = Vy(t)$ este setul de ecuații de paritate a cărui reziduuri redau signatura pentru defectele de măsură, în cazul în care nu apar defecte $p = 0$. Procedura prezentată până acum se bazează pe redundanță directă. În continuare este redată abordarea pentru cazul redundanței directe cât și redundanței temporale. Se pornește de la modelul în reprezentare de stare, radat în continuare,

$$\begin{aligned} y(t+1) &= CAx(t) + CBu(t) + Du(t+1) \\ y(t+s) &= CA^S x(t) + CA^{S-1} Bu(t) + \dots + CBu(t+s-1) + Du(t+s), s > 0 \end{aligned} \quad (16).$$

Pentru $s = 0, 1, \dots, n-1 < n$ putem scrie în formă compactă $Y(t) = Qx(t-n) + RU(t)$. Pentru a putea fi transformată în ecuație de paritate se impune înmulțirea cu un vector w^T care determină obținerea unei ecuații scalare și punem condiția $w^T Q = 0$. Se poate demonstra că după selectarea obiectivelor de proiectare abordările cu ecuații de paritate și cu observare converg către generarea unor reziduuri echivalente sau identice.

Metoda abordată în dezvoltarea proprie și anume metoda de diagnostic utilizând filtrul Kalman extins se încadrează în clasa metodologiilor cantitative bazate pe model și va fi tratată formal ulterior în lucrare subliniind atunci și detaliile de implementare. În continuare vor fi prezentate aspecte privind diagnoza utilizând estimarea parametrică. Diagnoza variațiilor parametrilor nu se poate realiza on-line fiind necesară o metodă de estimare parametrică.[10] Se impune de asemenea utilizarea unor modele parametrice precise ale sistemului, mai ales sub forma EDO sau EDP continue. Pentru descrierea procedurii se pornește de la modelul,

$$y(t) = f(u(t), \theta) \quad , (17)$$

Parametrii modelului θ relaționează cu parametrii fizici φ ai procesului prin $\theta = g(\varphi)$. Astfel modificările care apar în parametrii φ , referite ca $\Delta\varphi$ sunt calculate utilizând relația anterioară

și utilizând o metodă de recunoaștere se pot asocia aceste deviații, $\Delta\phi$, cu defectele din proces. Pentru implementarea acestei abordări se pot utiliza mai multe metode, cum ar fi metoda celor mai mici pătrate, metoda variabilei instrumentale sau estimarea utilizând modele în timp discret.[12] Aceste metode implică disponibilitatea unor modele dinamice precise ale procesului și pot determina un efort computațional mare pentru procese complexe, astfel un aspect important fiind complexitatea. Modelele procesului ar putea fi prezentate sub mai multe forme, bazate pe informațiile de I/E, utilizând modele neliniare sau modele reduse, însă un aspect important de studiat fiind robustețea.[13] Înainte de a trece la următoarea metodă cantitativă des utilizată se impune prezentarea câtorva aspecte legate de redundanță hardware și structurile bazate pe tehnica votului.[15] Tehnica votului este des utilizată în sistemele care posedă un grad înalt de redundanță hardware paralelă. De exemplu, să considerăm 3 senzori identici măsurând aceeași variabilă. Dacă unul din cele trei semnale de la senzori diferă de celelalte două, senzorul în cauză va fi considerat defect (tehnica 2 din 3). Schemele bazate pe tehnica votului sunt ușor de implementat și determină o metodă ușoară de detecție a defectelor. În continuare se trece la un studiu foarte important și anume generarea reziduurilor îmbunătățite. După cum am menționat anterior reziduurile pentru diagnostic reflectă defectele potențiale în sistem. Următorul pas este confirmarea prezenței unui defect și identificarea acestuia. Pentru etapa de izolare a defectului reziduurile trebuie să conțină suficientă informație nu numai pentru detecție cât și pentru izolare, trebuie să fie selective. În acest context generatorul de reziduuri trebuie să producă un set de reziduuri și fiecare să răspundă selectiv la un anumit defect, realizându-se astfel nu numai detecția ci și o clasificare a defectelor. Pornind de la aceste principii s-a încercat dezvoltarea unor generatoare de reziduuri capabile să genereze reziduuri îmbunătățite care să ofere și capacitatea de izolare, dezvoltându-se în această direcție două abordări, reziduurile direcționale și reziduurile structurate.[16] Reziduurile direcționale sunt dedicate unui anumit defect permitând izolarea lor pornind de la localizarea în spațiul multidimensional al reziduurilor. Proiectarea unui generator de reziduuri direcționale pornește de la un sistem liniar și invariant în timp. Considerând următorul model al sistemului,

$$h(z^{-1})y(t) = U(z^{-1})u(t) + V(z^{-1})p(t) + W(z^{-1})\omega(t), t \geq 0, \quad (18)$$

$p(t)$ este vectorul defectelor și $\omega(t)$ zgromotul. Generatorul de reziduuri este un operator liniar

dinamic ce operează asupra vectorilor $y(t)$ și $u(t)$ având forma $r(t) = G(z^{-1})y(t) + H(z^{-1})u(t)$, unde $H(z^{-1}) = -h^{-1}(z^{-1})G(z^{-1})U(z^{-1})$. Combinând ultimile ecuații putem obține următoarea formă a reziduurilor $r(t) = F(z^{-1})p(t) + L(z^{-1})\omega(t)$.[14] Abordarea bazată pe reziduurile structurate presupune generarea vectorilor reziduu astfel încât fiecare element reziduu să răspundă selectiv la un subset de defecte. În acest caz este necesar ca doar reziduurile specifice pentru un anumit defect să fie nenule ca răspuns la apariția defectului. Astfel se poate implementa un sistem de signaturi binare mai ușor de identificat și analizat. Aceste reziduuri structurate pot fi generate de către ecuațiile de paritate în modele de tip ARMA sau doar MA sau de modele în reprezentare de stare.[17] În continuare este descrisă procedura de generare a reziduurilor structurate. Astfel pentru un sistem liniar vectorul de intrări și cel de ieșire sunt legați de valorile reale astfel ,

$$\begin{aligned} u(t) &= u^0(t) + p(t) \\ y(t) &= y^0(t) + q(t) \end{aligned}, \quad (19)$$

unde $p(t)$ și $q(t)$ redau vectorii ce caracterizează defectele senzorilor și respectiv actuatorilor. Reziduul poate fi definit prin oricare dintre următoarele forme echivalente,

$$\begin{aligned} o(t) &= H(z)y(t) - G(z)u(t), \\ o(t) &= H(z)(y^0(t) + q(t)) - G(z)(u^0(t) + p(t)) \text{ sau} \\ o(t) &= H(z)q(t) - G(z)p(t) \end{aligned}, \quad (20)$$

reziduul pentru diagnoză este obținut prin transformarea $r(t) = W(z)o(t)$. Structurile de reziduuri sunt caracterizate prin matrici de incidență, a căror coloane sunt signaturile defectelor, iar liniile sunt reziduurile. De exemplu pentru un sistem cu 4 defecte posibile $F = [F_1 \ F_2 \ F_3 \ F_4]^T$ matricea de incidență poate fi reprezentată astfel

$$\begin{array}{ccccc} & F1 & F2 & F3 & F4 \\ r1 & \begin{pmatrix} I & I & 0 & I \end{pmatrix} \\ r2 & \begin{pmatrix} 0 & 0 & I & I \end{pmatrix} \\ r3 & \begin{pmatrix} I & I & 0 & I \end{pmatrix} \\ r4 & \begin{pmatrix} 0 & I & 0 & 0 \end{pmatrix} \end{array}, \quad (21)$$

unde elementul *I* indică faptul că reziduul e sensibil la defect în timp ce 0 marchează insensibilitatea la defect. În cazul în care avem două coloane identice cele două defecte nu vor putea fi discriminate. Unul din avantajele majore ale abordării cantitative bazate pe model rezidă în faptul că avem control asupra comportamentului reziduurilor. Totuși există o serie de factori cum ar fi, complexitatea sistemului, dimensiunile mari, neliniaritatea sistemului sau un set inconsistent sau incomplet de informații despre sistem care fac obținerea unui model matematic precis destul de grea. Metodele abordării cantitative pot fi evaluate din mai multe puncte de vedere cum ar fi viteza de detecție a defectului, izolarea și identificarea cu eroare minimă, robustețea la zgromot și incertitudine, adaptabilitatea, efortul de modelare și efortul computațional.[1]

În continuare este prezentată cea de-a doua abordare în diagnoza defectelor și anume abordarea calitativă bazată pe model și strategii de căutare. Această abordare presupune împărțirea activității de diagnostic în două componente importante și anume stabilirea domeniului de cunoștințe apriorice și strategia de căutare.[15] Baza de cunoștințe apriorice este necesară pentru a stabili niște relații preliminare între observații (de fapt simptome) și defecte și se poate implementa de exemplu utilizând tabele de look-up putând fi clasificată calitativ sau cantitativ. Modelul este dezvoltat uzual pe baza înțelegerii proceselor fizice care le implică. În abordarea cu modele cantitative înțelegerea descrisă anterior se referă la relații funcționale matematice între intrările și ieșirile sistemului. În cazul modelelor calitative se utilizează funcții specifice de calitate pentru fiecare unitate a sistemului. Pornind de la clasificarea făcută în figura 3 trebuie menționat că există două abordări diferite pentru căutare în diagnoza defectelor și anume căutarea topografică și căutarea simptomatică. Prima presupune o analiză a defectelor utilizând un şablon al operării normale în timp ce a doua metodă de căutare presupune determinarea unor simptome care să determine localizarea defectului. Pentru etapa de căutare în diagnoza defectelor s-au dezvoltat mai multe metode dintre care descompunerea funcțională, descompunerea structurală (căutare topografică), metoda ipoteză și test în buclă deschisă/inchisă și tabele de look-up (căutare simptomatică). Dezvoltarea sistemelor expert bazate pe cunoștințe a determinat utilizarea unei descrieri formale a acumulării de cunoștințe. Reamintind, un sistem expert este o aplicație software care mimează comportamentul cognitiv uman în rezolvarea anumitor probleme utilizând o bază de cunoștințe prestabilită.[26] Totuși această metodă are dezavantajul că nu poate surprinde elementele de fizica sistemului, acele detalii intime ale dinamicii sistemului. Prima metodă ce descrie cunoașterea calitativă este metoda modelelor cauzale bazate pe digrafuri.

Diagnoza poate fi considerată la un anumit nivel inversa simulării, fiind concentrată pe deducerea structurii din comportament. Astfel relațiile de cauzalitate sau modelele unui sistem pot fi descrise prin intermediul digrafurilor, care sunt de fapt grafuri cu arce orientate între noduri și care pot avea semne atașate arcelor (digrafuri cu semn, SDG).[18] Arcele orientate merg de la nodurile cauză la nodurile efect, fiecare nod fiind de fapt deviația unei variabile de la valoarea nominală. Digrafurile cu semn sunt instrumente foarte utile de reprezentare grafică a metodelor calitative, având cea mai utilizată formă de transpunere a cunoștințelor cauzale pentru diagnoza defectelor. La momentul actual se încearcă o combinare a logicii fuzzy cu metodele de diagnoză calitative crescând astfel granularitatea, utilizând reprezentări vagi ale variabilelor reale. În acest sens abordările hibride constituie un viitor în sensul că se vor putea dezvolta metode care să poată determina calea de propagare a defectelor în sistem menținând un control eficient al preciziei detecției și izolării prin metrici formale. În continuare este prezentată o altă metodă calitativă și anume metoda arborilor de defecte (fault trees). Arborii de defecte sunt utilizati în analiza fiabilității și siguranței unui sistem. Arborele defectelor este un arbore logic care propagă evenimentele primare (defectele) către evenimentul de nivel superior (hazardul) fiind construit pe o structură cu nivele de noduri, la nivelul fiecărui nod având loc operații de tip AND sau OR pentru propagare. Analiza generală a unui arbore de defecte conține 4 etape, definirea sistemului, construirea arborelui defectelor, evaluarea calitativă și evaluarea cantitativă. Arborii de defecte oferă mijloacele computaționale de combinare logică pentru analiza defectelor sistemelor.[21] Pentru a obține o diagnoză validă a defectelor utilizând această metodă se impune ca arborele să surprindă complet relațiile cauzale ale sistemului, deși nu există nici o metodă de verificare formală a preciziei arborelui dezvoltat. În interiorul unui arbore de defecte se pot pune în evidență căile cauzale prin care evenimentele primare (defectele) se pot propaga prin sistem pentru a cauza evenimentul de nivel superior (defecțiunea majoră). Problema de sinteză a unui arbore de defecte poate fi formulată sub forma unei probleme de căutare în spațiul stărilor. Pornind din starea inițială algoritmul aplică operatori de tip AND, OR sau XOR care transformă starea inițială în starea ţintă. Starea inițială este evenimentul de nivel superior, iar starea ţintă este arborele care conectează evenimentul de nivel superior cu evenimentele primare. După sinteza arborelui de defecte informația obținută de la acesta este stocată sub forma unor seturi. Un set conține acele evenimente primare care dacă apar simultan generează apariția evenimentului de nivel superior. O altă metodă din categoria metodelor calitative este metoda fizicii calitative.

Fizica calitativă a fost utilizată în scopul diagnozei defectelor în două direcții. Prima direcție se referă la derivarea ecuațiilor calitative din ecuațiile diferențiale, numite ecuații confluențiale. Reprezentarea ecuațiilor confluențiale se referă de fapt la niște ecuații algebrice care se pot obține fără a necesita multe informații despre sistem.[20] A doua abordare în fizica calitativă se referă la derivarea comportamentului calitativ din EDO pornind de la o descriere a mecanismului fizic al sistemului, se construiește apoi modelul și se utilizează un algoritm pentru determinarea comportamentelor sistemului fără a cunoaște precis parametrii și relațiile funcționale ale sistemului.[19] Un alt rezultat important în fizica calitativă este strategia de modelare compozitională pentru dezvoltarea unui sistem automat de formulare a unui model pentru sistemele de diagnoză, care este bazat pe o descriere a dinamicii sistemului. În continuare este prezentată metoda care se bazează pe ierarhia abstractizată a bazei de cunoștințe despre sistem. Această metodă presupune descompunerea sistemului în subsisteme și analiza comportamentului întregului sistem utilizând doar informația despre comportamentul componentelor sale. În acest mod se poate realiza o analiză fie din punct de vedere structural, fie funcțional. Sistemul poate fi reprezentat în acest context la nivelul operațiilor de I/E ale componentelor sale, descompunerea putându-se realiza la nivele diferite de abstractizare. Ierarhia structurală reprezintă informația de conectivitate a sistemului și componentelor sale. Diagnoza poate fi considerată o căutare de sus în jos de la un nivel înalt de abstractizare, unde se consideră sistemele și elementele de interconectare de nivel superior și până la un nivel de abstractizare jos, unde are loc o analiză a componentelor unui sistem și a funcționalităților acestora. Astfel, bazându-ne pe funcția pe care o îndeplinește subsistemele / componente pot fi clasificate ca : subsisteme funcționale, solicitate, saturate sau necontrolate și subsisteme funcționale sau nefuncționale. Aceste subsisteme sunt reprezentate la nivel jos cum ar fi senzori, actuatori, controllere. Ideea de bază este că defectul manifestat la nivel superior se datorează unei alterări a funcționalității unei componente de nivel jos și descrierea de nivel superior este utilizată pentru a identifica subsistemul care a generat defectiunea. Căutarea funcțională este o abordare naturală în diagnoza sistemelor complexe. S-a demonstrat că descompunerea ierarhică oferă modularitate efectivă pentru a organiza baze de cunoștințe de dimensiuni mari putând adresa și rezolvarea unor probleme locale.

În continuare este reluată descrierea metodelor calitative fiind prezentate tipologile de strategii de căutare pentru diagnoza defectelor. După cum am amintit anterior există două mari abordări în strategiile de căutare pentru diagnoza defectelor și anume căutarea

topografică și căutarea simptomatică. Căutarea topografică presupune căutarea efectuată în sistemul ce operează cu defect, având ca referință un şablon al operării normale și astfel defectul va fi găsit și identificat local.[22] Modul de operare a unui sistem de diagnoză bazat pe căutarea topografică este descris în figura următoare,

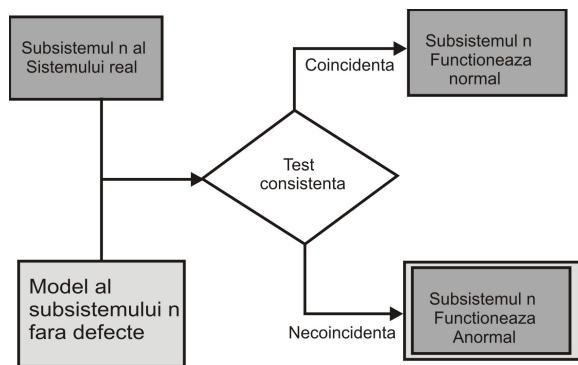


Figura 4 Modul sintetic de operare al căutării topografice

Un aspect important al căutării topografice constă în faptul că această metodă nu emite presupuneri vis-à-vis de modurile de operare cu defecte, se fac presupuneri doar în cazul operării normale.

În continuare este redată o prezentare a căutării simptomatice, care se bazează pe un set de informații despre operarea anormală a sistemului ca fiind şablonul de căutare pentru a găsi un set potrivit într-o colecție de simptome cunoscute legate de anumite condiții de operare anormale ale sistemului.[27] Primul tip de căutare simptomatică este cea bazată pe tabele de look-up, fiind și cea mai simplă metodă, şablonul (tiparul) comportamentului anormal și simptomele corespunzătoare fiind stocate într-un tabel (crescând totuși complexitatea odată cu complexitatea sistemului). Căutarea bazată pe ipoteză și test e bazată pe tipare de referință generate on-line, prin modificarea unui model funcțional în corespondență cu un defect ipotetic. Diagnoza se realizează în 3 etape, prima etapă este formularea ipotezelor, apoi determinarea efectelor defectului ipotetic asupra sistemului și în final compararea rezultatelor cu informațiile obținute de la sistem. [28]

Structura generală care descrie sintetic acest tip de căutare pentru diagnoza defectelor este descrisă în figura 5.

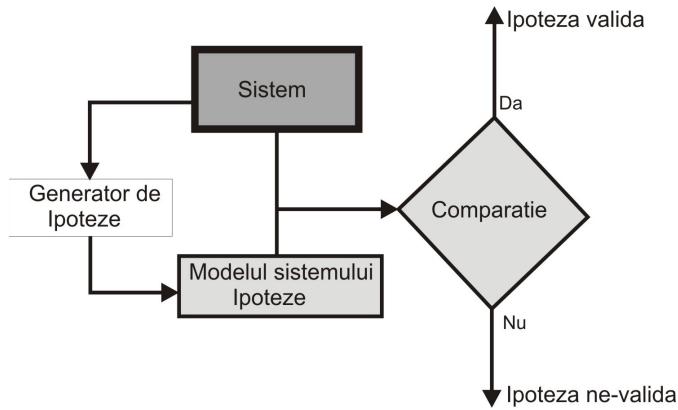


Figura 5 Generarea ipotezelor și testarea

În cazul utilizării pentru diagnoza în buclă închisă o ipoteză candidat este aleasă și este testat dacă referința este consistentă cu procesul. Dacă aceasta nu se potrivește cu procesul se alege una nouă, informația care rezultă la apariția unei diferențe este ulterior utilizată pentru generarea unei noi ipoteze. În cazul buclei deschise se utilizează mai multe modele de referință cu ipoteze diferite. Comparând referința cu procesul, cel mai apropiat model de referință față de proces este identificat ca fiind reprezentativ pentru proces. Deși tehnicele calitative au o serie de avantaje descrise de-a lungul prezentării ele prezintă și un mare dezavantaj care rezidă în generarea unor soluții false.

În continuare sunt prezentate tehnicele de diagnoză a defectelor, bazate pe cunoașterea istoricului sistemului. Spre deosebire de metodele bazate pe model unde sunt necesare cunoștințe apriori despre proces, în cadrul acestei metode, doar disponibilitatea unei cantități mari de informație despre proces este necesară. Extragerea caracteristicilor sistemului pentru diagnostic se poate realiza fie cantitativ, fie calitativ.[23] Din abordarea calitativă putem aminti două metode pentru extragerea informației despre proces, cu sisteme expert și metode de modelare a trendului (QTA). Metodele cantitative se pot clasifica în metode statistice și metode ne-statistice. Prima metodă, descrisă în continuare, se referă la extragerea caracteristicilor unui sistem utilizând reguli ce stau la baza sistemelor expert. Sistemul expert realizează următoarele etape în rezolvarea problemei propuse. Prima etapa este achiziția cunoștințelor, apoi are loc alegerea modului de reprezentare a cunoștințelor, codificarea cunoștințelor într-o bază de cunoștințe și în final dezvoltarea unor proceduri de inferență pentru luarea deciziilor pentru diagnostic.[25] Avantajul major al acestei abordări constă în faptul că sistemele expert pot opera în prezența incertitudinilor și în plus presupun o

dezvoltare facilă și transparența operării. Totuși limitările unui sistem expert apar prin însăși specificitatea regulilor emise, puterea de reprezentare limitată și dificultatea în updateul bazei de reguli. O altă abordare în extragerea calitativă a caracteristicilor procesului este abstractizarea informației despre trendul (tendența) procesului. Analiza trendului și prediciția sunt componente importante ale monitorizării proceselor și controlului prin supervizare. Modelarea trendului poate fi utilizată pentru a explica cele mai importante evenimente care au loc în proces, cu interes în diagnoză și prediciția stărilor viitoare. În ceea ce privește diagnoza, reprezentarea calitativă a trendului oferă informații prețioase despre comportamentul procesului. În cele mai multe cazuri defectele ce apar în operarea sistemului lasă un trend distinct în senzorii monitorizați, care poate fi utilizat pentru a identifica sursa funcționării anormale a sistemului.[25] În continuare se face trecerea la prezentarea metodelor cantitative de extragere a caracteristicilor sistemului pentru diagnoza defectelor. Aceste metode cantitative formulează rezolvarea problemei de diagnostic ca o problemă de recunoaștere a formelor (pattern recognition). Întâi problema de recunoaștere a formelor, este de a realiza clasificarea datelor în clase predeterminate. Metodele statistice (clasificatorul Bayes) utilizează cunoștințe apriorice despre distribuția claselor pentru a realiza clasificarea. Pe de altă parte, abordări precum analiza componentei principale (PCA) extrag informația despre trendurile majore ale datelor, utilizând un număr mic de factori relevanți. Primul aspect se referă la extragerea statistică a caracteristicilor sistemului, din informațiile disponibile despre sistem. În realitate sistemele sunt suspuse perturbațiilor aleatoare, evoluția viitoare nefiind determinată în totalitate de stările anterioare și prezente și de comenziile viitoare. În acest caz măsurătorile sunt considerate a fi serii de timp statistice. Astfel problema diagnozei unui sistem se poate formula ca problema detecției modificărilor parametrilor unui sistem stochastic dinamic sau static. În continuare este realizată o analiză formală a metodei analizei componentei principale (PCA) și metoda celor mai mici pătrate parțială (PLS). Teoretic vorbind metoda PCA se bazează pe o descompunere ortogonală a matricii de covarianță a variabilelor sistemului de-a lungul mai multor direcții punând în evidență maximul variației datelor. Principalul obiectiv în utilizarea PCA constă în găsirea factorilor care au valori mai mici decât lotul de valori originale și care descrie cel mai bine trendul major din lotul original. Considerăm p dimensiunea vectorului variabilelor măsurate ale procesului, X este un vector $n \times p$ – dimensional reprezentând vectorul scalat și mediat al măsurătorilor a cărui matrice de covarianță este Σ . Liniile lui X corespund fiecărui eșantion, iar coloanele corespund variabilelor. Matricea de covarianță poate fi redusă la o matrice L

printr-o matrice ortonormală $p \times p$ – dimensională, U , astfel încât $\Sigma = ULU'$. Coloanele lui U sunt vectorii de încărcare a componentei principale. Varianța este descrisă de elementele diagonale ale lui L , $\lambda_1, \lambda_2, \dots, \lambda_n$ fiind valorile proprii ale Σ . Transformarea care caracterizează metoda este $T = XU$ sau $\theta_i = Xu_i$ și astfel $X = TU' = \sum_{i=1}^p \theta_i u_i'$. Matricea transformării $T = (\theta_1, \theta_2, \dots, \theta_p)$ care reprezintă vectorul ce conține scorurile componentei principale care se definesc ca fiind valorile observate ale componentelor principale pentru cele n observații. La utilizarea metodei PCA în diagnoza defectelor și controlul calității se impune respectarea unor pași. Se stabilește pentru început un model bazat pe informațiile obținute din operarea normală și monitorizarea se realizează prin compararea factorilor cu acest model nominal. PCA este o procedură utilizată pentru o singură matrice de date, matricea variabilelor procesului X . Deseori se adaugă un vector al indicilor de calitate. Este de dorit să se includă toate informațiile disponibile în etapa de monitorizare și să se utilizeze variabilele de proces pentru a predicta și detecta schimbările care pot apărea în variabilele de calitate. Pentru ca acest lucru să fie posibil se utilizează metoda celor mai mici pătrate parțială (PLS) care modeleză relația între două componente, realizând și analiza variațiilor în datele procesului. Au fost dezvoltate multiple variațiuni ale celor două metode (MSPCA) aducându-se îmbunătățiri substanțiale în ceea ce privește aplicabilitatea, execuția și complexitatea implementării. După cum am menționat anterior diagnoza defectelor este în esență o problemă de clasificare și poate fi încadrată în categoria metodelor statistice pentru pattern recognition. Un exemplu bun ar fi în acest sens clasificatorul Bayes, care realizează în esență o combinare a estimărilor instantanee utilizând cunoștințe despre proprietățile statistice ale modurilor sistemului în prezența defectelor. Pe de altă parte un interes major în literatură pentru rezolvarea problemei de diagnoză a defectelor îl au rețelele neuronale.[28] Sunt recunoscute avantajele și utilizarea RN în probleme de clasificare și de aproximare a funcțiilor neliniare, fiind utilizate și pentru diagnoză. În strategiile de învățare supervizată prin alegerea unei topologii specifice de rețea, aceasta este parametrizată, în sensul că problema se reduce doar la estimarea ponderilor. Ponderile sunt învățate utilizând explicit diferența dintre valoarea dorită și cea curentă pentru a controla căutarea. Din acest motiv RN supervizate sunt potrivite pentru clasificarea defectelor din moment ce RN sunt capabile să genereze, deci să clasifice, regiuni arbitrarе din spațiu (forme). În continuare este realizată o comparație sintetică a metodelor descrise pentru implementarea diagnozei defectelor, accentul căzând pe metodele cele mai des

utilizate și urmărind criterii cum ar fi viteza de detecție și identificare, izolabilitatea, robustețea, adaptabilitatea, cerințele de modelare și altele (figura 6).

Metoda	Observere	Digrafuri	Ierarhie de abstractizare	Sisteme expert	QTA	PCA	Rețele Neuronale
Viteza de detecție și diagnoză	Da	?	?	Da	Da	Da	Da
Izolabilitatea	Da	Nu	Nu	Da	Da	Da	Da
Robustețea	Da	Da	Da	Da	Da	Da	Da
Adaptabilitatea	Nu	Da	Da	Nu	?	Nu	Nu
Cerințe de modelare	?	Da	Da	Da	Da	Da	Da
Eficiență Computațională	Da	?	?	Da	Da	Da	Da
Identificarea mai multor defecte	Da	Da	Da	Nu	Nu	Nu	Nu

Figura 6 Comparație între metodele prezentate pentru diagnoza defectelor

În finalul acestei prezentări a cadrului general de interes pentru tema de dezvoltare aleasă se impune precizarea unor aspecte de interes în proiectare și care întregesc descrierea problematicii diagnozei și toleranței la defecte. Primul aspect important util în proiectare se referă la localizarea optimală a senzorilor. Astfel cea mai mare parte a informațiilor despre starea sistemului provine de la senzori și de aici necesitatea unei poziționări optimale a senzorilor încă din faza de proiectare a sistemului. Ideea de bază este de a poziționa senzorii pentru a îmbunătăți observabilitatea, detectabilitatea și separabilitatea. Indexul detectabilității caracterizează capacitatea sistemului de a detecta anumite defecte și de a le separa în funcție de domeniul funcțional. Poziționarea senzorilor se realizează minimizând acești trei indici, ținând cont și de alți factori cum ar fi costurile, amplitudinea și densitatea defectelor. Un al doilea aspect se referă la reconcilierea datelor măsurate de la senzori care presupune de fapt 3 etape de analiză a semnalelor de la senzori, identificarea parametrilor care au depășit anumit prag, estimarea valorii la care s-au stabilizat parametrii și rectificarea măsurătorilor. Ultimul aspect se referă la controlul prin supervizare, activitate care se situează de fapt între control/reglare și planificare.

Raportat la problema diagnozei rolul supervisorului va fi de a utiliza informația disponibilă, de la sistemul de diagoză pentru a verifica și monitoriza buclele din nivelul de conducere. Dacă au apărut schimbări în buclele de control, supervisorul va găsi alte configurații de control și va seta un nou set de restricții pentru traectoria de stare care să îmbunătățească operarea sistemului către atingerea valorilor impuse pentru indicii de performanță. În final este prezentată o arhitectură generică, integrată, surprinzând modul de transformare al informației din nivelul de măsurare către nivelul de luare a deciziilor.[28]

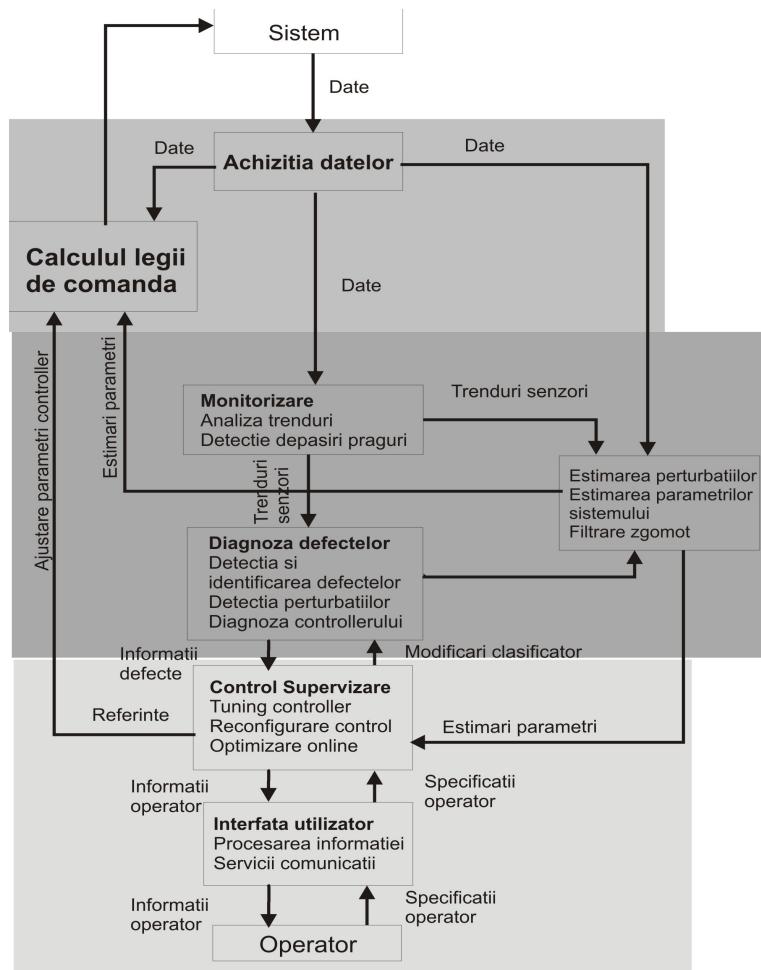


Figura 7 Structură generică, integrată pentru control, diagnoză și toleranță la defecte

În continuare se trece la o descriere completă a sistemului robotic dezvoltat și sunt surprinse aspecte constructive necesare pentru analiza și sinteza sistemului de conducere cu capacitați de diagoză și toleranță la defecte.

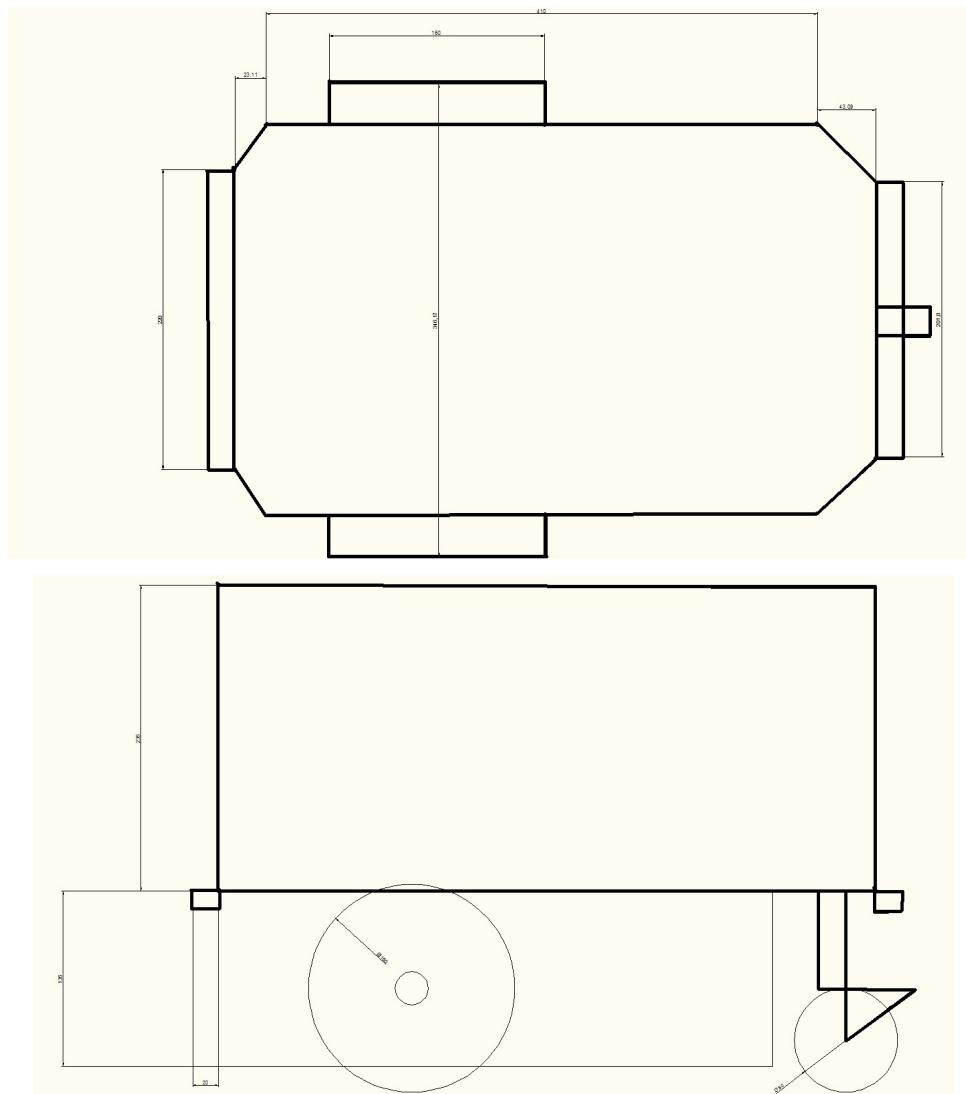
Capitolul 2

**Descrierea sistemului dezvoltat : robot mobil diferențial cu
2 roți motoare și o roată directoare**

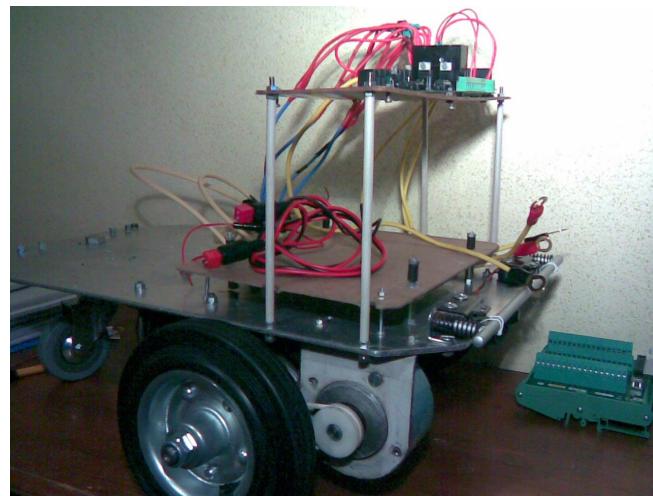
Descrierea sistemului dezvoltat: robot mobil diferențial cu 2 roți motoare și una directoare

Sistemul mecatronic dezvoltat este o platformă mobilă cu două roți motoare antrenate de 2 motoare de curent continuu, cu reductor și o roată pasivă directoare de tip castor. Șasiul a fost construit din aluminiu pentru a reduce masa totală dar și pentru a conferi robustețe întregii structuri. În figura următoare sunt prezentate câteva imagini ale robotului mobil în diferite faze de construcție.

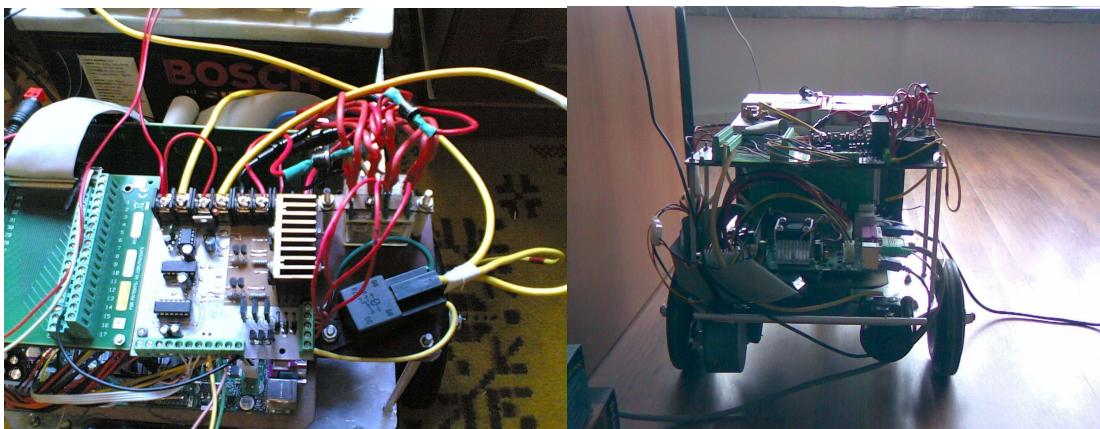
Prima fază : proiectarea CAD a șasiului și caroseriei robotului mobil



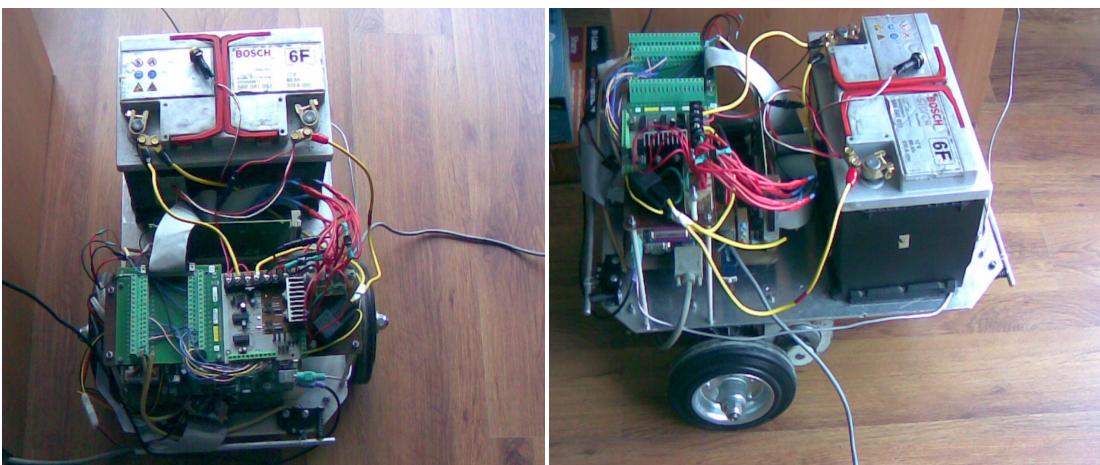
Faza a 2 a : şasiul robotului mobil şi subsistemul de locomoţie cu cele două motoare de curent continuu, encoderele şi bumperele



Faza a 3 a : subsistemul electronic, circuitele de electronică de putere și unitatea de procesare



Faza a 4 a: integrarea componentelor de bază



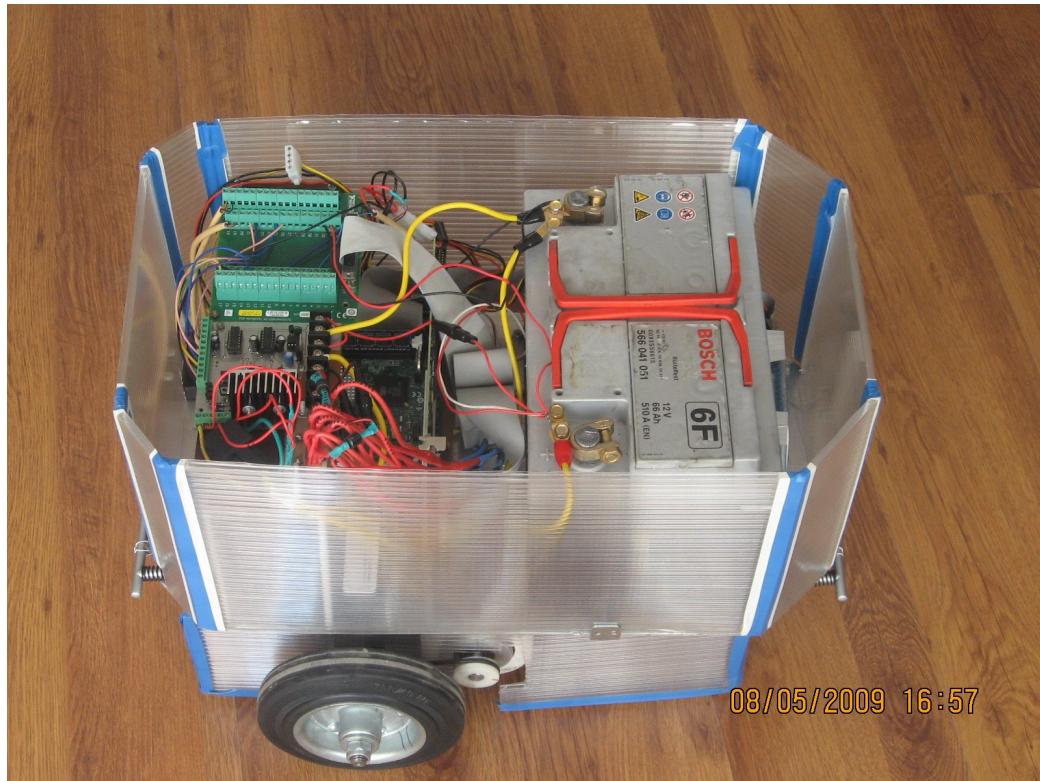
Faza a 5 a : structura finală

Figura 8 Fazele dezvoltării hardware a robotului mobil

2.1 Aspecte privind proiectarea și realizarea robotului mobil. Subsisteme specifice ale structurii generice de robot mobil diferențial cu 2 roți motoare și o roată directoare

Motivul alegerii structurii de robot mobil diferențial cu 2 roți motoare și una directoare provine din faptul că această structură de robot este simplă, o structură minimală ușor de proiectat și implementat și mai ales ușor de modelat. În continuare este făcută o descriere succintă a cadrului de interes pentru studiul sistemului robotic dezvoltat. Pentru a putea defini un vehicul robot, trebuie în primul rând făcut apel la una din definițiile de bază ale unui robot: “un dispozitiv mecanic care poate fi programat pentru a îndeplini anumite sarcini ce-i sunt date prin control automat”. O primă clasificare care se poate face, rezultă din faptul că există o diferențiere între vehiculele ghidate, precum AGV-urile (en. Automated Guided Vehicles) și cele neghidate. [29] Un vehicul ghidat este de regulă limitat la un set de

traекторii predefinite în spațiul său de lucru. Aceste traекторii pot fi marcate prin şine, linii optice sau magnetice, sau pur și simplu pot fi alcătuite dintr-o serie de poziții programate aprioric în memorie. Vehiculele ghidate nu pot, sub nici o formă, să părăsească traectoria prestabilită. O a doua clasificare are la bază mediul în care operează vehicoul robot. În timp ce vehiculele robot ghidate sunt terestre, cele neghidate pot acționa și sub apă respectiv în spațiu. Cea mai mare categorie însă este cea a vehiculelor robot neghidate terestre, care la rândul lor se pot clasifica în funcție de tipul de sistem de locomoție. Dintre toate sistemele de locomoție, cele mai des întâlnite sunt cele cu roți. În literatura de specialitate, vehiculele robot neghidate terestre cu roți se mai numesc și roboți mobili sau vehicule robot mobile autonome. În continuare se face o prezentare la nivelul subsistemelor unei structuri generice de robot mobil diferențial realizând o raportare directă la specificul sistemul dezvoltat.

2.1.1 Subsistemul de locomoție

Locomoția este procesul care asigură deplasarea robotului prin mediu. În cazul locomoției cu ajutorul roților trebuie ținut seama de anumite aspecte foarte importante, cum ar fi stabilitatea, manevrabilitatea și controlabilitatea.[30] Stabilitatea se referă la aspecte care privesc numărul și geometria punctelor de contact cu suprafața de operare, localizarea centrului de greutate și a centrului geometric, analiza stabilității în regim static și dinamic, analiza gradului de înclinare, profilul ariei de contact, unghiul de contact și tipul de frecare cu suprafața de operare a robotului. În funcție de mediul în care va fi utilizat robotul analiza se poate orienta fie spre analiza forțelor ce acționează asupra robotului (analiza dinamicii), fie pe analiza mișcării robotului la nivelul vitezelor (analiza cinematică). Una din tehniciile de estimare a poziției unui robot mobil în mediu este odometria. Prin odometrie se calculează distanța parcursă de un robot mobil în funcție de numărul de revoluții ale roților sale. În cazul unei roți ideale, la fiecare rotație a acesteia se poate considera că robotul a parcurs o distanță egală cu $2\pi r$, unde r reprezintă raza roții respective. În practică însă, datorită forțelor de frecare și a aluncărilor, estimările sunt mult mai puțin precise. Un robot mobil aflat într-un plan are 3 grade de libertate: o poziție (x, y) și o orientare θ față de axa orizontală. Tripletul (x, y, θ) redă poziția (relativă sau absolută) robotului mobil și constituie variabila efectivă de control a sistemului de locomoție. Datorită faptului că un robot mobil nu are control complet asupra celor trei variabile, o serie de manevre mai mult sau mai puțin complexe trebuie

îndeplinite pentru ca robotul mobil să ajungă dintr-un anumit punct în spațiu în altul. În structura roboților mobili, pe lângă roțile motoare, se regăsesc fie roți adiționale, fie alte puncte de contact care asigură stabilitatea robotului. Cele mai des întâlnite sunt roțile de tip castor care au 2 grade de libertate și permit rotirea în jurul axului central și o rotire în jurul punctului de contact cu platforma(offset). Sistemele de locomoție pot avea proprietăți și componente diferite unul față de celălalt.[31] În continuare este descris subsistemul de locomoție a robotului mobil dezvoltat. Primul nivel al sistemului de locomoție îl reprezintă roțile. Cele două roți motoare ale robotului sunt prevăzute cu jante din aluminiu și suprafață de rulare de cauciuc pentru a diminua efectele alunecărilor. Diametrul roților este de 160 mm și suportă sarcini de până la 140 de kg. Roata pasivă de tip castor are diametrul de 75mm și poate suporta o sarcină de până la 55 de kg (figura 9).



Figura 9 Tipurile de roți utilizate la construcția robotului

Următoarea componentă a subsistemului de locomoție sunt cele două motoare de curent continuu cu reductor ce asigură un cuplu suficient pentru deplasarea robotului la viteze acceptabile, luând în considerare și sarcina totală care se ridică la o valoare de 25 kg. Motoarele de curent continuu bi-direcționale ale producătorului Bosch au tensiunea de alimentare de 12V, puterea nominală de 40W și un curent maxim de 15A (în sarcina maximă). Viteza nominală este de 151 min^{-1} și un cuplu continuu de 2,5Nm utilizând un raport de reducție de 52:2. În figura 10 sunt redate caracteristicile motoarelor.

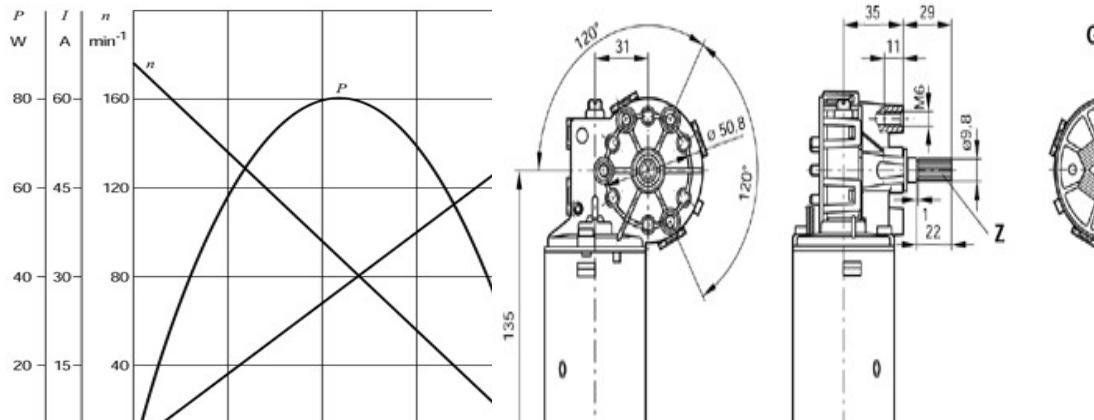


Figura 10 Caracteristicile motoarelor de curent continuu

Comanda motoarelor de curent continuu se realizează prin semnal PWM, generat de driverul de putere (punte H) care primește de la placa de achiziție un semnal analogic (tensiune în intervalul 0-5V) și generează la ieșire un semnal modulat cu factor de umplere variabil. Descrierea detaliată a driverului de putere pentru comanda motoarelor este prezentată în anexe.

2.1.2 Subsistemul de percepție

Percepția în cazul roboților mobili este realizată de un sistem senzorial, format din mai mulți senzori a căror informație este fuzionată și utilizată la localizare și control. În structura unui robot mobil pot intra mai multe categorii de senzori, senzori de stare internă (care servesc la obținerea de informații legate de funcționarea robotului, poziția relativă a elementelor, cuprelor cinematic, vitezele și accelerațiile liniare și unghiulare, deformațiile elementelor lanțului cinematic), senzori de stare externă (care dă informații despre mediul înconjurător și despre interacțiunile robot/mediu; servesc la identificarea prezenței și stabilirea tipului, poziției, orientării și a altor proprietăți ale obiectelor din mediu; la identificarea obstacolelor și forțelor de interacțiune robot/mediu), senzori pasivi (măsoară energia care intră în senzor, senzori de temperatură, microfoane, camere CCD sau CMOS) și senzori activi (emit energie în mediu și măsoară reacția mediului; au performanțe bune dar suferă din cauza interferențelor).[30] O clasificare sintetică a senzorilor se poate realiza cu scopul de a evidenția caracteristicile acestora și astfel de a delimita capacitatea robotului. Astfel accentul cade pe senzorii externi care pot fi cu contact (cuplați direct sau cuplați indirect) sau fără contact (de proximitate, optici sau de investigare). Se emit următoarele ipoteze conform cărora, orice senzor este afectat de zgomot, informația redată de senzor este incompletă și nu se poate realiza un model complet al unui senzor.

În principiu, orice model al unui senzor ar trebui să includă și un model intern al zgromotului care poate afecta senzorul în momentul citirii informației. Problema de a recupera informația din mediul din datele primite de la senzor poate fi destul de complexă. În principiu orice tip de senzor poate fi afectat de mai multe tipuri de erori. Dintre acestea, cele mai importante sunt erorile incidentale, erorile sistematice și erorile stochastice. Erorile incidentale apar ocazional și pot avea un efect neprevăzut asupra informației, ele provenind în cea mai mare parte de la măsurători efectuate greșit. Erorile sistematice au o influență predictibilă asupra acurateții informației, acestea provenind de la o interpretare greșită a parametrilor în algoritmii de estimare, sau din cauza unor neconcordanțe (incertitudini) în modelare. În fine, erorile stochastice, au un caracter aleator, ele diferind de fiecare dată când robotul execută aceeași operație. Structura de robot mobil diferențial dezvoltată este structura minimală care permite operarea robotului în regim de trajectory tracking (urmărirea unei curbe parametrizează cu restricții de timp). Sistemul senzorial al robotului dezvoltat cuprinde două encodere, pentru poziționare (odometrie) și două bumpere. Bumperele sunt cel mai des întâlnit tip de senzori de contact, fiind reprezentate prin simple contacte care dau la ieșire o valoare binară corespunzătoare stării curente, determinând o acțiune de frânare de urgență la contact. Encoderele sunt senzori de stare internă utilizati pentru implementarea sistemului odometric, care returnează poziția robotului. Encoderele incrementale Telemecanique (figura 10), dă la ieșire 500 impulsuri pe revoluție și utilizează o tensiune de alimentare de 5V semnalul de ieșire fiind direct preluat de intrarea în placa de achiziție dotată cu rezistori de pull-up.

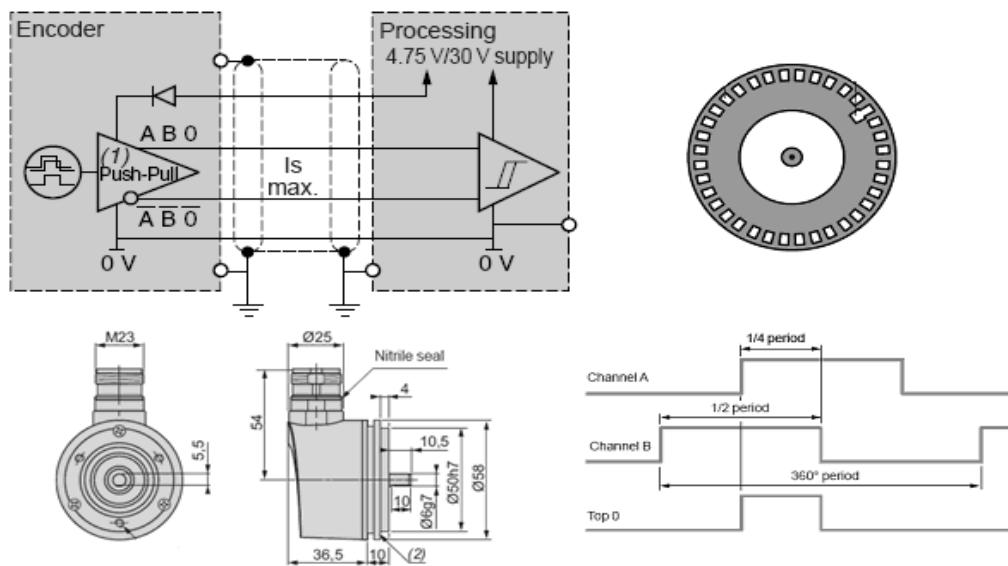


Figura 11 Caracteristicile mecanice și electronice ale encoderelor

2.1.3 Subsistemul de prelucrare a datelor și comunicație

Sarcinile pe care un robot mobil trebuie să le îndeplinească pot fi de la foarte simple la extrem de complexe. Totul depinde de scopul final al robotului mobil construit. În funcție de tipul de sarcini atribuite unui robot mobil, procesarea și descompunerea sarcinilor în acțiuni simple pe care robotul le poate executa necesită prezența unei unități centrale de procesare. Unitatea centrală de procesare se poate afla fie pe robot (en. onboard), fie la distanță conectată de acesta prin diverse metode (en. offboard). În mod normal, se dorește ca robotul mobil să fie autonom, deci să poate duce la bun sfârșit oarecum independent și sarcini mai complexe, fără intervenția unui program de control din partea utilizatorului, aflat la distanță. În multe cazuri însă, prezența unei unități de procesare pe robot duce la o autonomie scăzută a bateriei acestuia. O altă problemă ar putea fi creșterea în greutate a robotului, în funcție de tipul de unitate centrală de procesare aleasă. De asemenea, în cazul în care necesitățile computaționale sunt mari, s-ar putea ca o unitate centrală de procesare să nu existe astfel încât să se încadreze în limitele fizice și geometrice ale robotului, caz în care aceasta nu poate fi instalată. Există totuși o serie de avantaje în a avea o unitate de procesare onboard. Astfel dispar întârzierile asociate comunicației între o unitate de procesare offboard și componentele electronice ale robotului și în cazul unor probleme de comunicare, robotul poate acționa în continuare autonom. Unitățile de procesare onboard au acces direct la hardware-ul robotului, în timp ce unitățile offboard trebuie să apeleze la un anumit protocol de comunicație care să interfețeze între componente și unitate. Evident, soluția cea mai bună este ca robotul să poate funcționa atât autonom prin intermediul unei unități de procesare onboard, cât și în regim de comunicare cu alte unități de procesare mai performante, care pot asigura necesitățile computaționale pentru sarcini mai complexe. În practică unitățile de comunicare onboard se aleg astfel încât să aibă suficientă putere de calcul pentru a îndeplini majoritatea sarcinilor simple, cu specificația să nu consume prea multă energie. Unitățile offboard sunt de regulă reprezentate de calculatoare personale. În continuare este prezentată platforma de procesare a datelor existentă pe robot și care oferă suportul pentru implementarea aplicației de control tolerant la defecte a robotului mobil. Astfel robotul prezintă o platformă bazată pe un procesor Intel Pentium4 și o placă de bază VIA Mini ITX standard cu 1GB de memorie RAM și capacitatea de stocare 80GB. Deși este o placă destinată pentru aplicații embedded consumul de energie este mare deoarece procesorul nu este optimizat pentru astfel de aplicații. Interfața cu hardwareul a aplicației se realizează

prinț-o placă de achiziție de date PCI NI 6024E care este o placă multifuncțională I/O. Din capacitatele sale de I/O amintim 16 intrări analogice single-ended (sau 8 diferențiale), convertor ADC cu aproximări succesive de 12 biți rata de eșantionare maximă garantată fiind de 220kS/s; două ieșiri analogice la o rezoluție DAC (double-buffered, multiplicator) de 12 biți și o gamă de valori între +/- 10V pentru tensiunile de ieșire; 8 canale I/O digitale, programabile și 2 timere/countere de uz general de 24 de biți. Roboții mobili trebuie să posede capacitați de comunicare fie cu alți roboți din mediu fie cu un operator uman, pentru a raporta dacă o anumită sarcină dată a fost îndeplinită cu succes sau nu și pentru a transmite informații de stare. Comunicația cu fir (uzual, linie serială RS232) este modalitatea cea mai simplă de transfer de date între robot și operator. O serie de probleme asociate transmisiei fără fir, dispar în cazul comunicației de date cu fir, cu prețul îngădării mobilității robotului. Avantajul principal al sistemelor de comunicare cu fir este acela că, pe lângă datele transmise între robot și operatorul uman, se mai poate transmite și un curent electric, eliminând astfel necesitatea unei baterii onboard pe robot și mărimind autonomia robotului. Un dezavantaj direct al folosirii unui sistem de comunicare cu fir, ar fi faptul că spațiul de lucru al robotului va fi limitat de lungimea maximă admisibilă a cablului. De asemenea, cablul ar putea interfeța cu sistemul de locomoție sau chiar cu senzorii, aceștia putând raporta valori eronate. Din aceste considerente, nu se recomandă folosirea sistemelor de comunicare cu fir decât în cazurile în care folosirea tehnologiilor fără fir ar fi o problemă. Sistemele de comunicare fără fir se folosesc de o serie de tehnologii de transmitere a datelor prin aer, dintre care amintim: transmisia prin unde în spectrul infraroșu, bluetooth, radio modem-uri, WiFi și altele. Mecanismele de transmitere a datelor prin unde infraroșu sunt cele mai puțin robuste, ele neputând fi folosite decât la distanțe relativ mici. De asemenea, ele funcționează doar în linie dreaptă în câmp deschis, orice obstacol aflat între dispozitivele de transmisie-recepție obturând semnalul. Un alt impediment al tehnologiilor de transmitere în infraroșu este lumina solară. Tehnologiile WiFi sunt deja răspândite în domeniul calculatoarelor personale și oferă un suport complet de integrare al roboților în rețele de calculatoare. O problemă a acestor tehnologii ar fi faptul că sunt consumatoare de energie, lucru care nu este prielnic robotului mobil. Eliminând problema energiei consumate, tehnologiile Bluetooth oferă servicii similare cu dispozitivele WiFi, însă distanța maximă admisibilă între emițător și receptor este mult mai mică decât în cazurile celorlalte tehnologii. În momentul de față, tehnologiile Bluetooth se folosesc în interiorul clădirilor sau în laborator, unde distanța maximă fără repetitor este de maxim 20m. În fine, tehnologiile radio modem oferă o soluție wireless pentru acele

dispozitive care sunt deja conectate prin cablu. Consumul de energie al modemurilor radio este de regulă redus, ele fiind optimizate pentru aplicații unde nu se dorește o viteză foarte mare de transmisie a datelor. În cazul robotului dezvoltat am ales utilizarea unui modul de comunicație wireless pe portul USB care are încorporat un chipset (Ralink RT73) ce realizează preluarea pachetelor TCP/IP emise de aplicația de control utilizând facilitățile stivei implementate în sistemul de operare și împachetează apoi informația (date și comenzi) în pachete compatibile cu standardul 802.11b/g la o viteză maximă de 54Mbps pentru comunicarea cu clientul aplicației distribuite conectat la un router wireless.

2.2 Modelarea matematică a robotului mobil cu 2 roți motoare și una directoare

2.2.1 Modelul cinematic al robotului

Cinematica este o ramură a mecanicii care studiază mișcarea corpurilor independent de cauzele care o produc. Modelul cinematic cuprinde analiza din punct de vedere geometric a robotului redând acele ecuații ce pun în evidență legătura între mărimele de control și comportamentul robotului în spațiu.[38] Pentru a analiza cinematica robotului mobil trebuie studiate constrângerile impuse de roțile robotului. Astfel pe durata determinării modelului cinematic pentru un robot mobil se consideră robotul ca fiind un corp rigid cu roți ce se deplasează în plan orizontal.[34] Gradul maxim al modelului cinematic este astfel 3, două grade pentru poziția în plan și un grad pentru orientarea de-a lungul axei verticale care este ortogonală pe plan. Pentru a putea determina poziția robotului mobil în plan trebuie să stabilim o relație între sistemul de referință global (sau inerțial) al planului și sistemul de referință al robotului. Axele X_i și Y_i definesc sistemul de referință global, iar $\{X_r, Y_r\}$ este sistemul de referință al robotului care se alege într-un punct P de pe șasiu, de fapt fiind centrul de rotație al robotului. Poziția punctului P în sistemul de referință global este dată de coordonatele (x, y) și de diferența unghiulară dintre sistemul de coordinate global și sistemul de coordinate a robotului, notat θ . Poziția robotului se poate scrie ca un vector de trei elemente $\xi_l = [x \ y \ \theta]^T$ și pentru a descrie mișcarea robotului este necesar să transformăm mișcarea din sistemul de referință global în mișcare de-a lungul sistemului de referință al robotului. În figura 12 este redată reprezentarea robotului în plan pentru analiza cinematică.

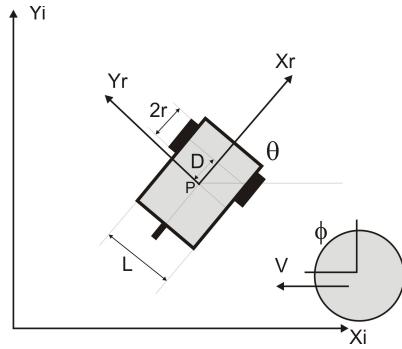


Figura 12 Reprezentarea robotului pentru analiza cinematică

Relația de transformare între cele două sisteme de referință se poate realiza utilizând ecuația 22 prezentată în continuare,

$$\dot{\xi}_R = R(\theta) \dot{\xi}_I = R(\theta) \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}, R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, (22).$$

Cunoscând viteza robotului data de $\dot{\xi} = [\dot{x} \dot{y} \dot{\theta}]^T$ care poate fi definită de viteza de rotație a roții ϕ , unghiul direcției β_i , viteza unghiulară $\dot{\beta}_i$ și de parametrii geometrici ai robotului putem obține descrierile pentru cinematica directă în ecuația 23,

$$\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\dot{\phi}_1, \dots, \dot{\phi}_n, \beta_1, \dots, \beta_m, \dot{\beta}_1, \dots, \dot{\beta}_m), (23)$$

și ecuațiile pentru cinematica inversă sunt redate în ecuațiile 24,

$$[\dot{\phi}_1 \dots \dot{\phi}_n \beta_1 \dots \beta_m \dot{\beta}_1 \dots \dot{\beta}_m]^T = f(\dot{x}, \dot{y}, \dot{\theta}), (24).$$

Cunoscând raza unei roți, r , lățimea robotului (wheel base), θ orientarea (heading) și vitezele de rotație ale roților ϕ_l , ϕ_r se poate obține modelul cinematic direct pentru robotul cu două roți diferențiale, descris de ecuația 25,

$$\dot{\xi} = [\dot{x} \dot{y} \dot{\theta}]^T = f(L, r, \theta, \phi_l, \phi_r), (25).$$

Putem determina mișcarea robotului în coordonatele sistemului de referință global funcție de coordonatele sistemului de referință local, prin inversa matricii de transformare $R(\theta)$. Se pornește cu calculul mișcării fiecărei roți în sistemul de coordonate local și apoi translatăm ecuațiile găsite în sistemul de coordonate global. Contribuțiile fiecărei roți la viteza de translație a punctului P pe direcția Xr sunt

$$\dot{x}_r = R\dot{\varphi}_r, \dot{x}l = R\dot{\varphi}_l, \quad (26)$$

iar viteza totală de translație a robotului este

$$v_R = \frac{v_l + v_r}{2}, \quad (27).$$

Contribuțiile fiecărei roți la viteza unghiulară a robotului sunt descrise de ecuațiile 28,

$$\omega_l = 2r\dot{\varphi}_l/L, \omega_r = 2r\dot{\varphi}_r/L, \quad (28),$$

viteza unghiulară totală a robotului fiind dată de următoarele ecuații echivalente ,

$$\dot{\theta} = \omega_l + \omega_r, \omega_R = \frac{v_R - v_L}{L}, \quad (29).$$

În final modelul cinematic al robotului mobil cu locomoție de tip diferențial poate fi redat prin următorul set de ecuații echivalente,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r\dot{\varphi}_r + r\dot{\varphi}_l \\ 0 \\ 2r\dot{\varphi}_l/L - 2r\dot{\varphi}_r/L \end{bmatrix}, \text{ sau} \begin{cases} \dot{x} = v \cos\theta \\ \dot{y} = v \sin\theta \\ \dot{\theta} = \omega \end{cases}, \quad (30).$$

Cinematica directă ține seama de modelul geometric al robotului și de vitezele de rotație ale roților pe care le transformă în viteze globale. Înainte de trece la studiul cinematicii roților se impune prezentarea modificărilor aduse modelului cinematic generic întrucât în cazul robotului dezvoltat centrul de rotație se află translatat cu o distanță D față de poziția clasică din model. Astfel în cazul de față robotul este descris de următorul model cinematic,

$$\begin{cases} \dot{x} = v_R \cos\theta - D\omega_R \sin\theta \\ \dot{y} = v_R \sin\theta + D\omega_R \cos\theta \\ \dot{\theta} = \omega \end{cases}, \quad (31).$$

Pentru implementare se discretizează modelul de mai sus obținându-se următoarele ecuații, în care T_S este perioada de eşantionare,

$$\begin{cases} x(k+1) = x(k) + [r_r\omega_r(k)\left(\frac{\cos(\theta(k))}{2} - \frac{D\sin(\theta(k))}{L}\right) + r_l\omega_l(k)\left(\frac{\cos(\theta(k))}{2} + \frac{D\sin(\theta(k))}{L}\right)]T_S, \\ y(k+1) = y(k) + [r_r\omega_r(k)\left(\frac{\sin(\theta(k))}{2} + \frac{D\cos(\theta(k))}{L}\right) + r_l\omega_l(k)\left(\frac{\sin(\theta(k))}{2} - \frac{D\cos(\theta(k))}{L}\right)]T_S, \\ \theta(k+1) = \theta(k) + \left(\frac{r_r\omega_r(k) - r_l\omega_l(k)}{L}\right)T_S \end{cases}, \quad (32)$$

Ecuațiile modelului cinematic sunt utilizate în cadrul implementării de față în sistemul de odometrie, care se concretizează de fapt ca fiind calea de reacție în bucla de control pentru poziționare. În continuare sunt prezentate alte aspecte legate de cinematica robotului mobil și studiul cinematicii roților. Astfel se fac următoarele presupuneri via-a-vis de mișcarea roților, mișcarea roților se va realiza numai în plan orizontal, contactul roții cu solul va fi un punct, rotile nu sunt deformabile, mișcarea este de tip rostogolire pură, nu există alunecări și nu există frecare în cazul rotirii în jurul punctului de contact. Roata pasivă a robotului este de tip castor și este descrisă de ecuațiile din figura 13,

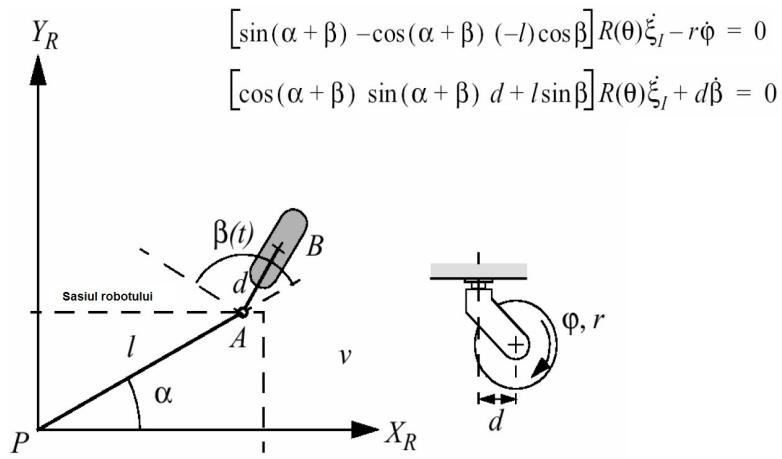


Figura 13 Ecuațiile roții pasive

Deși un aspect important cinematica roții pasive nu a fost considerată în implementarea de față doarece introducerea de noi termeni și calcule suplimentare în modelul matematic ar fi determinat o creșterea a timpului de calcul și astfel ar fi fost nevoie de o mărire a perioadei de eşantionare pentru a obține rezultate valide pentru faza de control în timp real.

În cinematica roboților apar restricții. Astfel fiecare roată fixă sau directoare standard poate impune restricții în mișcarea robotului. Fiecare roată are o axă de rotație perpendiculară pe planul roții. Roata trebuie să se miște de-a lungul unui cerc cu raza R astfel încât centrul acestui cerc să fie axa roții,. Centrul cercului este denumit centru instantaneu de rotație (ICR) și astfel putem defini manevrabilitatea robotului mobil. Manevrabilitatea poate fi considerată o combinație între mobilitate (dată de restricțiile de alunecare) și orientare (contribuția direcției). În acest context limitările modelului cinematic al șasiului robotului redau holonomia sistemului. Astfel un robot holonomic nu are restricții în cinematică, poate merge în orice direcție la orice moment de timp, iar un robot nonholonomic are restricții cinematice impuse de roțile fixe și de roțile directoare standard. Ultimul aspect legat de analiza modelului cinematic al robotului se referă la sinteza controllerului cinematic, care are rolul de a urmări traекторia descrisă prin puncte (coordonate) și / sau viteză în funcție de timp, neînținând seama și de mărimile dinamice ale sistemului. Această analiză preliminară a controlului robotului mobil este necesară pentru a stabili cadrul de interes în sinteza controllerului Sliding Mode pentru poziționare utilizând modelul cinematic al robotului. Astfel, în cazul controlului mișcării în buclă deschisă putem considera de exemplu că traectoria este împărțită în segmente diferite, linii, arce de cerc. Controllerul trebuie să calculeze o comandă astfel încât robotul să parcurgă o traекторie lină. Apar însă probleme legate de fezabilitatea calculului traectoriei, impunându-se limitări fizice în privința vitezelor și accelerărilor robotului și mai ales în cazul în care mediul se modifică dinamic.[37] Controlul în buclă deschisă se referă la cazul în care dorim să minimizăm eroarea între poziția inițială și poziția finală a robotului, figura 14.

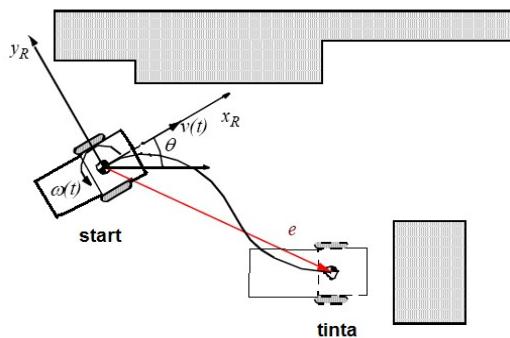


Figura 14 Operarea în buclă deschisă a robotului mobil

Astfel se impune găsirea matricii K a controllerului

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \quad (33),$$

unde $k_{ij} = k(t, e)$, astfel încât

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = Ke = K \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad (44)$$

să ducă la anularea asimptotică a erorii, e . În cazul controlului mișcării în buclă închisă ecuațiile cinematice ale robotului în coordonate inerțiale sunt date de ecuațiile 30, erorile dintre poziția inițială și cea finală fiind date de următoarele relații în coordonate polare,

$$\begin{aligned} \rho &= \sqrt{\Delta x^2 + \Delta y^2}, \\ \alpha &= -\theta + \alpha \tan 2(\Delta y, \Delta x), \quad (45) \\ \beta &= -\theta - \alpha \end{aligned}$$

Ecuațiile cinematice în coordonate polare vor fi considerate pentru două cazuri prezentate în ecuațiile 45 și reprezentarea sintetică din figura 15

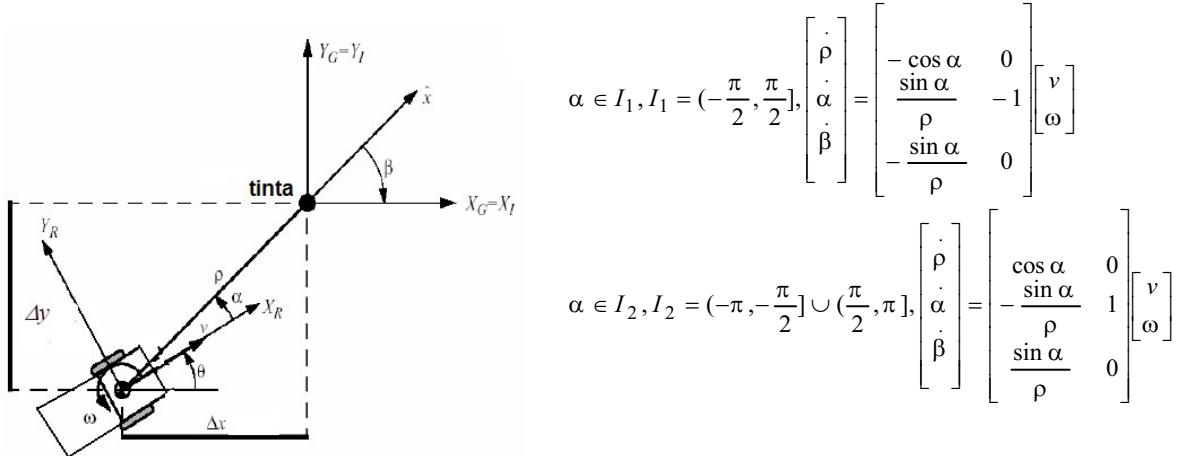


Figura 15 și ecuațiile 45 Reprezentarea în coordonate polare

Pentru reprezentarea polară se impun câteva remarci, astfel pentru α în primul caz poziția finală a robotului este pe direcția înainte, iar pentru al doilea caz direcția este opusă.

Datorită faptului că în implementare am utilizat o lege de comandă neliniară (control discontinuu) se vor emite la sinteza controllerului ipoteze privind stabilitatea locală și cuplajul intern al variabilelor eroare a căror convergență trebuie asigurată prin legea de comandă.

2.2.2 Modelul dinamic al robotului

În continuare se trece la studiul modelului dinamic al robotului mobil utilizând formularea Lagrange.[29] Datorită faptului că pentru moment am ales implementarea controlului tolerant la defecte utilizând doar modelul cinematic, descrierea modelului dinamic va fi doar una formală, pentru o posibilă viitoare utilizare în control. Modelul dinamic al robotului leagă parametrii mișcării robotului mobil cu forțele aplicate și cu cuplurile. În timp ce dinamica directă analizează răspunsul robotului mobil la o anumită comandă, dinamica inversă este utilizată la sinteza controllerului. Ecuațiile dinamicii unui robot mobil pot fi deduse utilizând fie metoda Newton – Euler, fie formalismul Lagrange. În continuare se consideră deducerea modelului dinamic utilizând formalismul Lagrangean pentru a obține ecuația dinamicii robotului mobil. Ecuația dinamicii depinde de mai mulți parametri fizici disponibili, însă pentru cei care nu sunt disponibili se va realiza o estimare utilizând o metodă de identificare. Ecuația dinamică completă de mișcare este neliniară și complexă de aceea se recurge la o decuplare, reducând problema la a analiza un grad de libertate în mișcare. Se pornește de la structura descrisă în figura 12 și considerăm valabile condițiile de mișcare fără alunecare și de pură rostogolire a roților. În acest caz considerăm β unghiul de orientare al roții pasive și d distanța de la punctul de fixare al roții pasive la roata pasivă, r_1 raza roții pasive și φ_p poziția unghiulară a roții pasive, l este distanța de la centrul sistemului de referință al robotului la punctul de fixare al roții pasive, m masa roților motoare, m_1 masa roții pasive, I_{r1} inerția roții pasive, I_{r2} inerția roții dreapta, I_{r3} inerția roții stânga, I_0 inerția robotului, M masa robotului, F_i sunt forțele și cuplurile de frecare ale roților, C_{s1} cuplu de frecare Coulomb roata pasivă, C_s cuplu de frecare Coulomb roată activă, C_{v1} coeficient de frecare vâscoasă roata pasivă, C_v coeficient de frecare vâscoasă roată activă, (x_m, y_m) coordonatele centrului de masă. Vectorul de coordonate generalizate care descrie complet mișcarea robotului este :

$$q(t) = (x, y, \theta, \beta, \varphi_l, \varphi_r, \varphi_p), \quad (46)$$

După cum se poate observa sistemul are 5 constrângeri independente și se poate demonstra că posedă 2 grade de libertate.

Ecuațiile dinamicii se pot deduce din formularea Langrangeanului:

$$\frac{d}{dt} \left[\frac{\partial [L(q, \dot{q})]}{\partial \dot{q}} \right] - \frac{\partial [L(q, \dot{q})]}{\partial q} = A(q)^T \lambda + B(q)\tau - F_r, \quad (47)$$

În ecuația de mai sus L este funcția Lagrange a sistemului, egală cu diferența între energia cinetică a sistemului și energia potențială. În cazul robotului mobil energia potențială este 0 și astfel

$$L(q, \dot{q}) = E_c = \frac{1}{2}(\dot{q})^T M(q) \dot{q}, \quad (48).$$

S-a demonstrat că

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} = A(q)^T \lambda + B(q)\tau - F_r, \quad (49),$$

unde, M este o matrice 7×7 -dimensională reprezentând inerția, C este matricea forțelor și cuplurilor centrifuge și Coriolis, A este matricea constrângerilor, λ este vectorul de dimensiune 5 al multiplicatorilor lui Lagrange asociați cu constrângerile cinematice independente, B este matricea cuplurilor și forțelor externe ce acționează asupra robotului mobil și F_r este vectorul forțelor de frecare. Datorită constrângerilor există o matrice $S(q)$ care satisfac relația,

$$\dot{q} = S(q)\eta, \eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}, \quad (50).$$

O proprietate importantă a matricii S , care dă posibilitatea eliminării multiplicatorilor lui Lagrange din ecuație, este ,

$$S(q)^T A(q)^T = 0, \quad (51).$$

$$\text{Derivând ecuația 50 obținem, } \ddot{q} = \frac{d[S(q)\eta]}{dt} = \frac{\partial [S(q)\eta]}{\partial q} \dot{q} + S(q)\dot{\eta}, \quad (52).$$

Înmulțind ecuația 49 cu $S(q)^T$ și înlocuind \ddot{q} cu ecuația 52 obținem următoarea formă de reprezentare a dinamicii,

$$S(q)^T M(q) S(q) \dot{\eta} + S(q)^T M(q) \frac{\partial S(q)}{\partial q} S(q) \dot{\eta}^2 + S(q)^T C(q, \dot{q}) S(q) \dot{\eta} = S(q)^T A(q)^T \lambda + S(q)^T B(q) \tau - S(q)^T F_r$$

din care efectuând înlocuirile necesare putem obține modelul dinamic al robotului mobil în spațiul stărilor. Astfel notând

$$\begin{aligned} J(q) &= S(q)^T M(q) S(q), \quad g(q, S(q) \dot{\eta}) = S(q)^T M(q) \frac{\partial S(q)}{\partial q} S(q) \dot{\eta}^2 + S(q)^T C(q, \dot{q}) S(q) \dot{\eta}, \\ G(q) &= S(q)^T B(q), \quad f_r = S(q)^T F_r, \end{aligned} \quad (53)$$

putem da reprezentarea structurală a modelului dinamic al sistemului,

$$\begin{cases} J(q) \dot{\eta} + g(q, \dot{\eta}) = G(q) \tau - f_r \\ q = S(q) \mu \end{cases}, \quad (54)$$

În continuare sunt explicitate matricile din modelul de mai sus. Astfel avem ecuațiile 55,

$$S(q) = \begin{bmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \\ \frac{-\sin\beta}{d} & -\frac{d+l\cos\beta}{d} \\ \frac{\cos\beta}{d} & -\frac{l\sin\beta}{d} \\ \frac{r_1}{r} & \frac{r_1}{L} \\ \frac{1}{r} & \frac{2r}{L} \\ \frac{1}{r} & -\frac{L}{2r} \end{bmatrix}, \quad M(q) = \begin{bmatrix} R(q)^T M(\beta) R(\theta) & R(\theta)^T V(\beta) & 0 \\ V(\beta)^T R(\theta) & I(\beta) & 0 \\ 0 & 0 & I(\phi) \end{bmatrix}, \quad R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M(\beta) = \begin{bmatrix} M(\beta)_{11} & 0 & M(\beta)_{13} \\ 0 & M(\beta)_{22} & M(\beta)_{23} \\ M(\beta)_{31} & M(\beta)_{32} & M(\beta)_{33} \end{bmatrix}, \quad V(\beta) = \begin{bmatrix} m_1 d \cos\beta \\ m_1 d \sin\beta \\ I_{r1} + m_1 d^2 + m_1 l d \cos\beta \end{bmatrix},$$

$$I(\phi) = \begin{bmatrix} I_{r1} & 0 & 0 \\ 0 & I_{r2} & 0 \\ 0 & 0 & I_{r3} \end{bmatrix}, \quad I(\beta) = [m_1 d^2 + I_{r1}]$$

$$\begin{aligned}
 M(\beta)_{11} &= M(\beta)_{22} = M + 2m + m_1 \\
 M(\beta)_{13} &= M(\beta)_{31} = My_m + m_1 l + m_1 d \cos \beta \\
 M(\beta)_{23} &= M(\beta)_{32} = Mx_m + m_1 d \sin \beta \\
 M(\beta)_{33} &= M(x_m^2 + y_m^2) + 2m \frac{L^2}{4} + m_1 l^2 + m_1 d^2 + 2m_1 l d \cos \beta + I_{r1} + I_0
 \end{aligned}$$

$$C(q, \dot{q})\ddot{q} = \frac{d[M(q)]}{dt}\dot{q} - \frac{1}{2} \frac{\partial [q \quad M(q)\dot{q}]}{\partial q}, \quad B = \begin{bmatrix} 0_{5 \times 1} \\ I_{2 \times 2} \end{bmatrix}, \quad \begin{cases} F_r = [0_{3 \times 1} \quad F_1 \quad 0 \quad F_2 \quad F_3]^T \\ F_i = C_s sign(\dot{\phi}_i) + C_v \dot{\phi}_i, i = l, r \\ F_l = C_{s1} sign(\dot{\beta}) + C_{v1} \dot{\beta} \end{cases}$$

Un aspect important în studiul dinamicii robotului mobil este studiul modelului motorului de curent continuu. Pentru robotul mobil dezvoltat putem cunoaște doar tensiunea aplicată motoarelor și astfel ne propunem să determinăm o relație între tensiune și cuplu. Astfel cuplul generat de fiecare motor este $\tau_m = K_T i_a$, unde K_T este constanta de cuplu și i_a este curentul aplicat armăturii. Cuplul aplicat roțiilor motoare este $\tau = t_r \mu (\tau_m - \tau_{fr})$, unde t_r este raportul de reducție, μ este eficiența reducatorului, τ_{fr} este cuplul de frecare. Tensiunea aplicată armăturii motorului este $V_a = R_a i_a + L_a \frac{di_a}{dt} + K_E t_r \dot{\phi}_i$, unde K_E este constanta tensiunii contra-electromotoare, R_a este rezistența armăturii, L_a este inductanța armăturii și $\dot{\phi}_i$ fiind vitezele unghiulare ale roțiilor. Neglijând inductanța, curentul prin motor este dat de relația

$$i_a = \frac{V_a - K_E t_r \dot{\phi}_i}{R_a} \text{ și astfel cuplul aplicat roțiilor este } \tau = t_r \mu [K_T \left(\frac{V_a - K_E t_r \dot{\phi}_i}{R_a} \right) - \tau_{fr}]. [47]$$

Datorită faptului că nu toți parametrii modelului dinamic sunt cunoscuți se impune utilizarea unei metode de identificare parametrică pentru a-i determina, parametrii considerați fiind inerțiile roțiilor și coeficienții de frecare vâscoasă și Coulomb. Pentru a determina parametrii de mișcare se poate utiliza o metodă de identificare integrală (c.m.m.p) evitând necesitatea calculului accelerării (ce poate introduce zgromot) și astfel reducând numărul de intrări în algoritmul de identificare. Parametrii necunoscuți se pot identifica efectuând experimente decuplate pentru fiecare grad de libertate și astfel se pot obține valori valide pentru etapa de simulare a modelului dinamic.[46] În continuare se face o descriere a modalităților de control dezvoltate pentru roboții mobili.

2.3 Abordări și aspecte legate de metodele de control și operarea roboților mobili

În contextul roboților mobili se urmărește atingerea autonomiei și dobândirea inteligenței de către robot. Inteligența este legată de capacitatele robotului de a observa, înțelege și învăța diferite aspecte ale sarcinilor sale. Cunoștințele dobândite pe parcursul procesului de învățare trebuie să ducă la performanțe mai bune pentru executarea sarcinilor pe viitor.[37] De asemenea, prin procesul de înțelegere al anumitor fenomene, informațiile obținute pot duce la soluții pentru rezolvarea anumitor sarcini noi, sarcini pentru care robotul nu a fost proiectat inițial. În multe aplicații se impune aplicarea tehnicielor de inteligență artificială pentru controlarea robotului mobil. Inteligența unui robot este demonstrată prin modul în care robotul poate naviga eficient în mediu, cu alte cuvinte, prin modalitatea prin care procesul de navigare diferă de cazul în care acțiunile robotului ar fi controlate direct de un operator uman. Autonomia unui robot mobil se referă la capacitatea acestuia de a executa independent sarcinile ce-i sunt date aprioric, fără intervenția unui operator uman.[39] Dacă în executarea sarcinilor sale, robotul întâlnește situații pentru care nu a fost programat, este imperativ ca el să continue să funcționeze corect, fără intervenție exterioară, îmbinând astfel conceptele de inteligență și autonomie. Se spune despre un robot mobil că este automat dar nu autonom, în momentul în care o situație neașteptată apare, duce la oprirea execuției sarcinii pe care robotul mobil o desfășura în momentul respectiv. Dacă robotul se poate descurca cu noua situație întâlnită, în sensul că o ignoră sau face mici ajustări și continuă sarcina, robotul are un grad mic de autonomie. Dacă însă robotul se adaptează perfect la situația nouă prin observarea caracteristicilor mediului, atunci gradul de autonomie al robotului mobil este mare. Procesul de navigare determină mișcarea robotului mobil în mediul său de lucru. Procesul de percepție înregistrează și prelucrează datele de la senzori, transformându-le în ecuații de mișcare pentru navigator.[35] Descompunerea în blocuri funcționale a robotului este prezentată în figura 11.

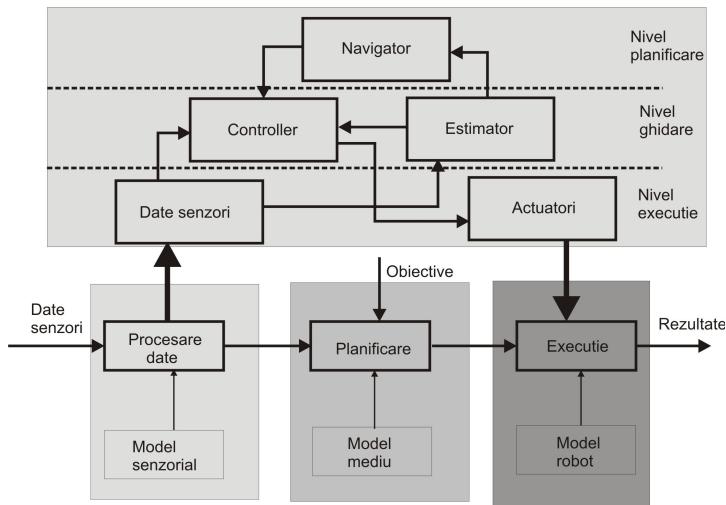


Figura 16 Descompunere funcțională

Procesul de navigare presupune mișcarea robotului în mediul său curent de operare. Procesul de percepție preia și prelucrează date de la senzori și le transformă în ecuații de mișcare pentru navigator. După cum se poate observa din figura de mai sus, prima etapă constă în prelucrarea datelor de la senzori, în funcție de modelul senzorial aferent existent. Rezultatul prelucrării se îmbină cu sarcinile atribuite robotului mobil în procesul de planificare. În această etapă, un rol important îl joacă modelul mediului existent. În fine, etapa finală este cea de execuție, în care rezultatele procesului de planificare se îmbină cu modelul robotului, trimițând comenzi necesare către motoarele robotului.[44] Pe măsura execuției algoritmilor de control, navigatorul trebuie să aibă în vedere și actualizarea celor 3 modele. Erorile existente între modelele vechi și realitate trebuie minimizate și noile modele calculate trebuie stocate în locul celor vechi. Cele trei modele nu vor fi perfecte niciodată datorită incertitudinilor existente în realitate. Pentru reducerea incertitudinilor se pot folosi diversi algoritmi de adaptare. Problema fundamentală a metodei de funcționare a navigatorului descrisă anterior este că necesită un timp de execuție mult prea mare. La fiecare pas, comanda către motoare trebuie să fie calculată din datele de la senzori prin planificare. Datorită acestui fapt, robotul nu va putea niciodată naviga cu viteze mari. O serie de calcule complexe vor uza de puterea de calcul a unității centrale de procesare. Soluția problemei de mai sus se obține prin împărțirea funcțiilor navigatorului în două categorii: de nivel jos (nivel de ghidare), respectiv de nivel înalt (nivel planificare). [43]

Funcțiile de nivel jos sunt reprezentate printr-un controler de traiectorii și un estimator de poziție (nivelul de ghidare). Acest nivel acționează ca o buclă închisă rapidă. Funcțiile de nivel înalt, operează în paralel cu funcțiile de nivel jos, oferind informațiile necesare controlerului pentru a opera, dar la un interval de timp mai mare decât rata cu care operează nivelul ghidare. Un robot mobil poate opera fie în buclă deschisă (dezavantajul major fiind că în cazul modificării mediului de operare controllerul nu va mai putea compensa eroarea neavând informație din partea sistemului senzorial), fie în buclă închisă. Funcțional operarea generică a unui robot mobil se poate descrie prin următoarele etape, achiziția de informații de la senzori, planificarea operației următoare ținând cont de obiectivele pe termen scurt(ex : ocolire obstacol) și de cele pe termen lung (ex: deplasarea pe o traiectorie impusă) și emiterea comenzi către efectori pentru atingerea obiectivelor.

În contextul anterior s-a făcut referire la modul generic de operare a unui robot sub forma navigării, în continuare se vor prezenta principiile care stau la baza celor trei tipuri de operare pentru roboți mobili și anume, trajectory tracking, path following și point stabilization. Înainte de a trece la descrierea efectivă a metodelor dezvoltate pentru controlul mișcării roboților mobili se impune descrierea cadrului de interes în dezvoltarea și evoluția metodelor de control al mișcării pentru roboții mobili. Cercetările în această direcție au început prin încercarea de a obține urmărirea perfectă a referinței de viteză, astfel încât obiectivul principal era găsirea unei metode de stabilizare în buclă închisă pentru modelul cinematic.[38] Un aspect important este faptul că la nivelul cinematic modificările parametrilor dinamicii robotului nu sunt sensibile și astfel poate apărea instabilitatea. Astfel, putem considera că un control la nivel dinamic este la fel de important ca și cel la nivelul cinematic. Pentru a elmina problemele care apar în controlul utilizând controlere clasice (ex PID), care nu păteau asigură performanțe acceptabile în prezența neliniarităților și variației în timp a parametrilor, s-a propus proiectarea unor controlere dinamice neliniare. Utilizarea unor controlere neliniare determină o creștere a gradului de stabilitate, deși au o structură mai complexă și sunt mai greu de implementat.[42] În controlul mișcării obiectivul este de a controla viteza robotului mobil astfel încât poziția curentă a robotului, notată cu $P = [x, y, \theta]^T$ să urmărească traiectoria de referință $P_r = [x_r, y_r, \theta_r]^T$. Pornind de la ipoteza unei urmăriiri perfecte a referinței de viteză și considerând doar sistemul de locomoție în care vectorul vitezelor era intrarea pentru robot, s-a dezvoltat o buclă de control utilizând doar modelul cinematic, figura 17,

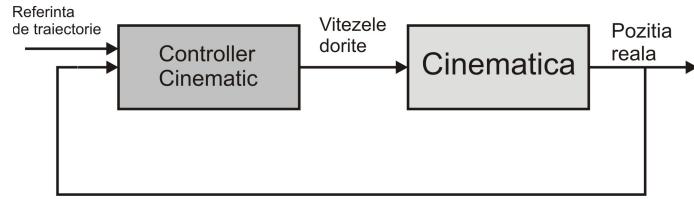


Figura 17 Bucla de control al mișcării utilizând modelul cinematic

Ulterior dezvoltarea s-a orientat către proiectarea controllerelor dinamice pentru controlul mișcării. În realitate robotul este descris de două aspecte, primul se referă la dinamica sa și al doilea la cinematică pentru a obține cele mai bune performanțe în controlul mișcării. Astfel bucla de control a mișcării va trebui să fie descompusă în două bucle, o buclă internă care va depinde de dinamica robotului și care va fi utilizată pentru a controla variabilele viteză (liniară și unghiulară) prin actuatorii robotului și o buclă externă pentru controlul poziției robotului. Astfel structura din figura 17 va fi extinsă și o imagine concretă a abordării curente este descrisă în figura următoare.

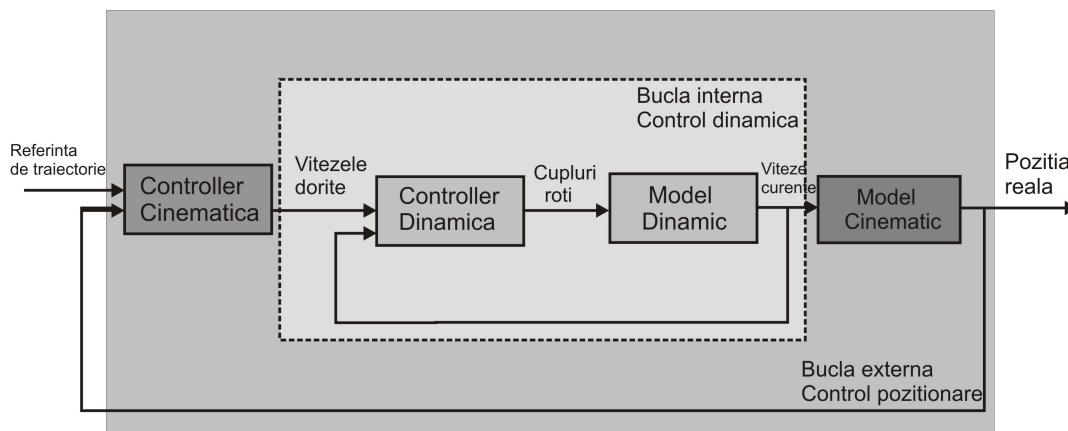


Figura 18 Structura generică de control al mișcării robotului mobil

S-a demonstrat că un sistem nonholonomic nu se poate stabiliza într-un singur punct de echilibru printr-o lege continuă (lină) cu reacție de la stare (și invariantă în timp), pornind de la faptul că sunt încălcate condițiile de stabilitate emise de Brockett.[42] Un robot mobil este controlabil local și pe un interval de timp scurt indiferent de structura constrângerilor nonholonomice. Astfel posibilitățile pentru controlul robotilor mobili se rezumă la utilizarea unor legi de comandă discontinue invariante în timp sau legi de comandă neliniare continue

variante în timp. Astfel în ceea ce privește modurile de control pentru roboții mobili au fost dezvoltate trei abordări fiecare implicând un grad de dificultate și limitări de realizare.

Stabilizarea într-un punct (point stabilization) are ca scop stabilizarea robotului mobil într-un punct țintă cu o anumită orientare. Această abordare reprezintă o adevarată provocare în proiectarea controllerului în contextul în care robotul are constrângeri nonholonomice și nu se poate atinge obiectivul utilizând o lege de comandă cu reacție de la stare, propunându-se metode care presupun sinteza unei legi de comandă continuă variată în timp, discontinuă și invariantă în timp sau o lege de comandă hibridă. Teorema lui Brockett demonstrează că stabilizarea se poate obține dacă se introduce o discontinuitate fie în comandă fie în timp. Tehnicile de sinteză a unei legi discontinue invariante în timp pot fi de două tipuri și anume, utilizând legi de comandă continue pe porțiuni, fie utilizând controlere sliding mode. Controlul sliding mode poate oferi o rată de convergență bună fără să alunece pe o înfășurătoare către echilibru, însă deseori apare problema oscilațiilor (chattering) la comutarea controllerului pe o altă lege de comandă. Controlerle continue pe porțiuni pot fi de mai multe feluri și utilizează o transformare de coordonate introducând un punct de discontinuitate în origine. Acest tip de controlere oferă o convergență exponențială fără apariția chatteringului determinând traiectorii liniare în mișcarea robotului, însă nu sunt la fel de robuste la incertitudini și perturbații ca un controller sliding mode. Legile de comandă variante în timp și cele hibride suferă în ceea ce privește rata de convergență care este scăzută, determinând generarea unor traiectorii care nu sunt liniare și astfel un comportament ne-natural al robotului.[40]

În cazul controlului mișcării pentru path following robotul mobil are ca obiectiv convergența și urmărirea unei căi fără a avea specificații sau restricții temporale. Ideea de bază este că viteza de înaintare a robotului urmărește un anumit profil de viteză în timp ce controllerul acționează asupra orientării pentru a-l conduce pe traiectoria impusă. În aplicațiile cu roboți mobili se obișnuiește ca traiectoria de operare să fie memorată sau generată în prealabil de un modul de path planning. În ceea ce privește path following există o clasificare care descrie mai multe posibilități de abordare. În continuare este redată o imagine sintetică a problemei de path following pentru roboții mobili.

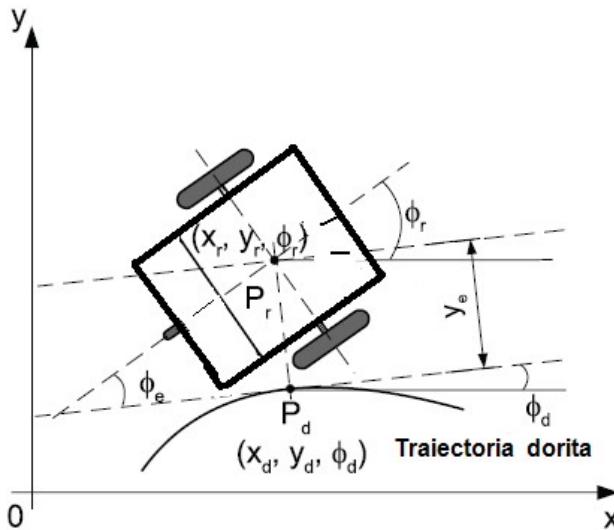


Figura 19 Descrierea sintetică a problemei de path following

Prima categorie a metodelor de path following pornește de la faptul că punctul dorit din traiectorie este obținut printr-o proiecție normală de-a lungul unui vector n . Astfel, această proiecție alege punctul traectoriei dorite care are coordonata x_e nulă. Se impune o valoare constantă pentru viteza liniară pentru a garanta mișcarea continuă a robotului. Se sintetizează o lege de control asimptotic convergentă și se ține cont că traectoriile care conțin cercuri de rază foarte mică (raza nulă și curvatură infinită) sunt interzise asigurând astfel că proiecția normală există și este unică. A două categorie se bazează pe transformarea ecuațiilor cinematicii robotului într-o altă scală temporală aleasă pentru a fi în concordanță cu distanța traectoriei. Totuși în acest caz traectoriile dorite se limitează doar la linii drepte. A treia categorie se bazează pe minimizarea distanței Euclidiene între punctul real și cel dorit obținându-se rezultate bune pentru traectorii mixte. În final sunt prezentate anumite aspecte care sunt specifice operării în regim de path following. Astfel pentru path following este necesară doar forma globală a traectoriei și nu are un rol important evoluția traectoriei (evoluția temporală). Pentru a determina poziția dorită se utilizează o anumită relație, de fapt funcția de proiecție, ce are rolul de a proiecta vectorul de poziție actual pe traectoria de referință. În cazul în care robotul se oprește punctul țintă trebuie de asemenea să rămână la aceeași valoare și modificările să fie dependente de mișcarea robotului. În cazul path following traectoria de referință nu poate "trage" robotul (ca în cazul trajectory tracking), însă robotul trebuie să se mișeze independent, convergența la traectorie fiind asigurată de legea de comandă sintetizată de controller. În cazul operării în regim de trajectory tracking

este garantat că sistemul va converge către traectoria dorită într-un interval de timp deterministic utilizând o lege de comandă asimptotic stabilă. Pentru a rezolva problema trajectory trackingului s-au dezvoltat abordări utilizând controllere sliding mode, tehnici de control adaptive robuste, backstepping și metode soft computing (rețele neuronale).[37]

În continuare este redată o imagine sintetică a problemei de trajectory tracking pentru roboții mobili și se realizează o scurtă analiză formală din punctul de vedere al sintezei legii de comandă pentru controlul mișcării roboților nonholonomici în regim trajectory tracking.

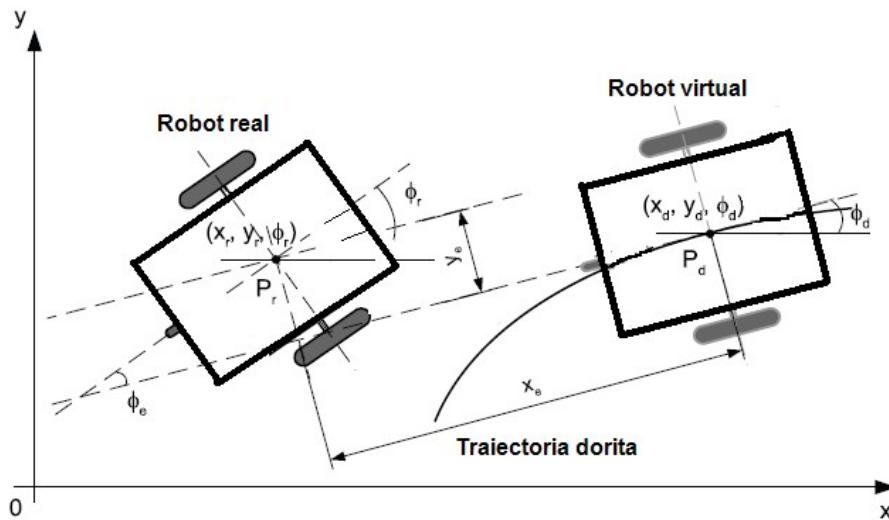


Figura 20 Descrierea sintetică a problemei de trajectory tracking

Considerând un sistem în reprezentare de stare $\dot{x} = Ax + Bu$, cu perechea (A, B) controlabilă și matricea K aleasă astfel încât $A+BK$ este stabilă din punct de vedere Hurwitzian. Se consideră de asemenea o traectorie $t \rightarrow x_r(t)$ cu $\dot{x}_r = Ax_r + Bu^r$. Legea de comandă este dată de

$$u(x, x_r, u^r) = u^r + K(x - x_r), \quad (56)$$

care se bazează pe $x_e = x - x_r$ și $\dot{x}_e = (A + BK)x_e$. Din moment ce $A + BK$ este stabilă legea de comandă din ecuația 56 stabilizează asimptotic orice traectorie de referință admisibilă. S-a demonstrat prin teorema lui Brockett că nu se poate sintetiza un controller generic continuu și invariant în timp pentru stabilizarea asimptotică a punctelor fixate pentru care $u^r = 0$ (care se pot stabiliza prin controllere variante în timp). Din moment ce nu se poate realiza stabilizarea universală a tuturor traectoriilor admisibile au fost dezvoltate trei direcții de lucru. Prima direcție presupune stabilizarea unei părți a stării sistemului, un exemplu fiind

problema controlului roboților în care orientarea nu este considerată prioritară ca obiectiv.[35] A doua direcție privește stabilizarea asimptotică a anumitor traекторii, astfel se impun condiții suplimentare care să aigure că u^r nu converge la 0. A treia direcție presupune relaxarea obiectivului stabilizării asimptotice fiind admise și erori de urmărire mărginite și de valori mici. În acest sens s-a dezvoltat abordarea funcției transverse care stă la baza obținerii unor controlere ce asigură mărginirea uniformă a erorii de urmărire, cu o precizie arbitrară pre-specificată indiferent de traectoria de referință, astfel eliminându-se problema delimitării traectoriilor admisibile. În cazul sistemelor nonholonomice stabilizarea asimptotică nu poate fi obținută mereu și mai ales nu se poate asigura, de fapt nu se poate asigura combinarea stabilității și convergenței către traectoria dorită. Astfel problema se rezumă la a alege între o convergență rapidă, însosītă de sensibilitatea la erorile de model și la zgomotul de măsură sau o convergență mai lentă dar mai robustă.[40]

Problema sintezei controllerului pentru trajectory tracking va fi reluată în capitoalele următoare care privesc utilizarea unui controller sliding mode pentru robotul mobil dezvoltat analizând și problemele enunțate în acest capitol legate de stabilizarea asimptotică.

Capitolul 3

| Descrierea aplicației software de control tolerant la defecte

Descrierea aplicației software de control tolerant la defecte

De interes major în dezvoltarea sistemului este aplicația de control tolerant la defecte în timp real. Astfel urmărind aspecte precum controlul distribuit, autonomia robotului, extensibilitatea structurii și flexibilitatea implementării am dezvoltat o aplicație puternică care susține la nivel inferior comunicarea facilă cu hardware-ul, implementarea taskurilor de control și diagnoză în timp real la nivelul intermediar și un nivel superior al comunicației cu alte entități din mediul de operare.[48]

3.1 Conducere în timp real. Instrumente utilizate, facilități specifice și analiza performanțelor soluției utilizate

Orice aplicație în timp real este un sistem reactiv. Astfel ea se poate iniția fără un set explicit de intrări și apoi poate primi anumite intrări la care reacționează prin anumite ieșiri. În mod generic o aplicație de control în timp real controlează un mediu extern. În acest caz cerințele și constrângerile temporale și de răspuns ale aplicației sunt mai dure.[60] Aplicația de timp real trebuie să reacționeze la evenimentele exterioare asincrone într-un anumit interval de timp impus de dinamica mediului extern controlat. Algoritmii execuți de aplicația în timp real trebuie să se execute periodic cu o perioadă ce trebuie să fie minimă și deterministică. O concepție greșită despre sistemele de timp real este că au nevoie de resurse și de putere de calcul foarte mare, ei bine în realitate resursele necesare sunt proporționale cu cerințele de operare ale sistemului controlat și edificatoare în această direcție este afirmația “real time doesn’t mean fast it means just in time” ! Sistemele în timp real trebuie să raspundă la evenimente periodice sau neperiodice și pot fi evenimente externe sistemului sau interne. Sistemele în timp real sunt la bază concurente și datorită faptului că mediul controlat e dominat de evenimente care se desfășoară în paralel, sistemul de timp real se bazează pe concurența proceselor interne pentru asigurarea atingerii tuturor obiectivelor.[61]

În continuare sunt prezentate câteva caracteristici importante ale unui sistem în timp real. Astfel după cum am menționat sistemele în timp real au cerințe dominate de constrângeri temporale, deci atributile cele mai importante fiind determinismul la nivel logic (funcțional) dar și la nivel temporal (dinamic) și predictibilitatea. Un alt aspect se referă la complexitate, (sistemele în timp real presupunând un număr foarte mare de intrări și ieșiri) și

la localizare, în genere sistemele în timp real având și componente distribuite. Un ultim aspect se referă la siguranța în operare și la fiabilitatea componentelor constituente, care trebuie să se reflecte în timpi mari de operare efectivă chiar și în prezența unor defecte.[52]

3.1.1 Utilizarea unui sistem de operare real time bazat pe LinuxOS. Utilizarea Linux-RTAI pe platformă bazată pe Intel P4

În ceea ce privește sistemul robotic mobil dezvoltat atributele de timp real se regăsesc în partea de control și monitorizare, în sensul că aplicația software dezvoltată are rolul de a asigura operarea robotului în contextul unor restricții temporale și de siguranță pre-stabilite. În implementare caracteristicile de timp real au fost introduse la nivelul sistemului de operare ce rulează pe platforma de prelucrare a datelor de pe robot. Astfel utilizând o platformă Intel P4 bazată pe un procesor x86 am instalat o distribuție standard de Linux OS și anume RedHat9 pe kernel 2.4.24 peste care am aplicat patchul RTAI (Real Time Application Interface) pentru obținerea de facilități hard real time. Motivația alegerii unui sistem de operare de tip UNIX rezidă în faptul că sistemul de operare este open-source și astfel s-au minimizat costrurile implementării; în plus kernelul oferă fără nici o modificare caracteristici de timp real (soft real time) bune, însă pentru a permite creșterea performanțelor în contextul unor taskuri consumatoare de resurse și minimizarea latențelor s-a decis modificarea kernelului de bază pentru îmbunătățirea caracteristicilor de timp real (hard real time) prin utilizarea RTAI. În continuare sunt prezentate acele trăsături de bază ale sistemului de operare în timp real care au fost urmărite la dezvoltare și apoi se vor studia aspecte privind capacitatea și mecanismele interioare ale kernelului Linux în ceea ce privește suportul pentru timp real.[54] Din punct de vedere funcțional un sistem de operare în timp real nu diferă de unul de uz general însă apar diferențe la nivelul dimensiunilor (micro-kernel, nano-kernel), la modul de programare al taskurilor periodice cu sau fără mecanisme de IPC (Inter Process Communication), la planificarea strictă a taskurilor de timp real (introducerea unui sistem de priorități) și nu în ultimul rând minimizarea latenței prin introducerea preemptibilității. Sistemul de operare în timp real utilizat în cadrul proiectului este RTAI și a fost dezvoltat de către DIAPM (Departamentul de Inginerie Aerospațiale de la Politehnico Milano). Nucleul (kernelul) sistemului de operare are rolul de a media accesul la resursele hardware pentru aplicațiile utilizator. [69] Nucleul are două moduri (contexte) de execuție, în mod utilizator sau în mod kernel, care sunt descrise în figura 21,

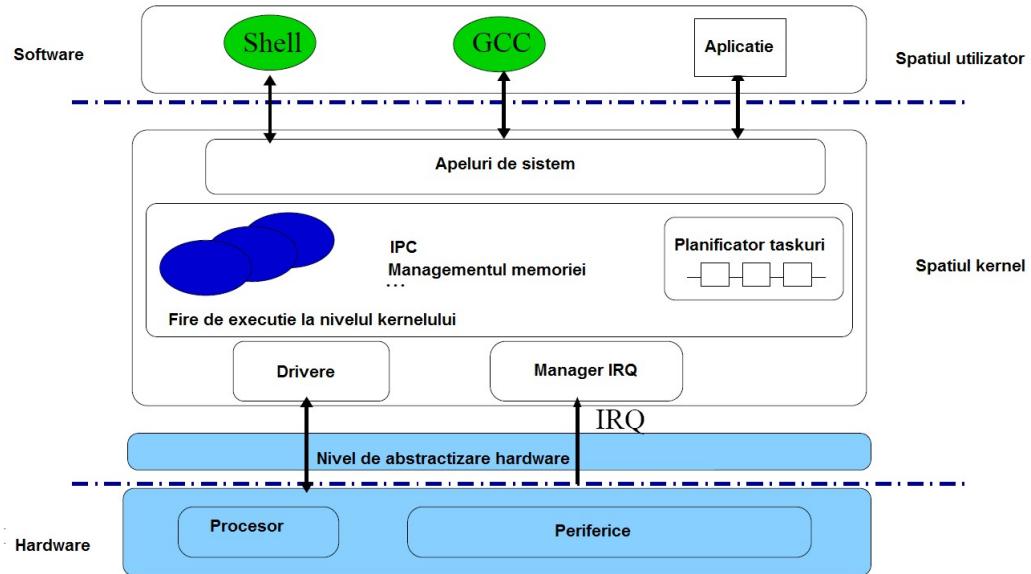
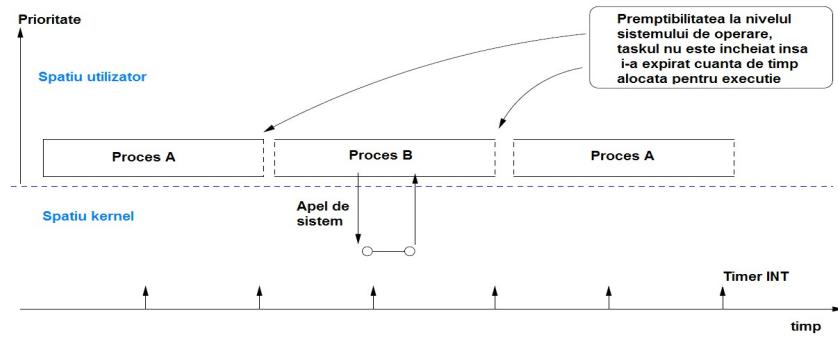
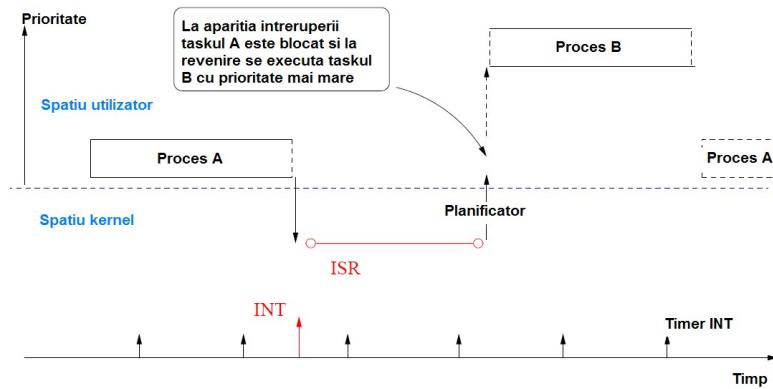


Figura 21 Arhitectura simplificată a sistemului de operare utilizat

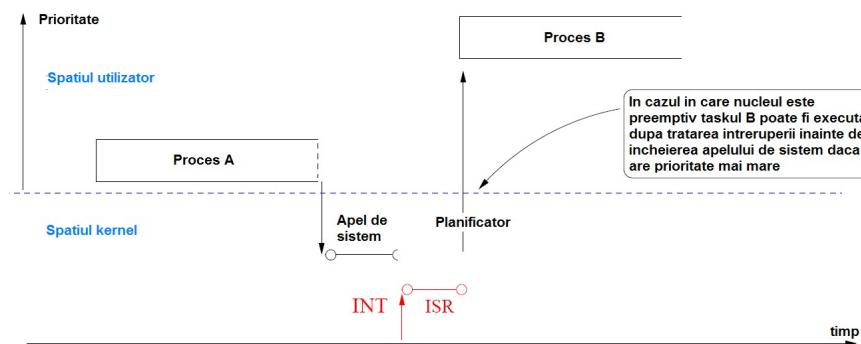
Spațiul utilizator este contextul de execuție al aplicației care poate accesa și servicii ale nucleului prin apeluri de sistem și apeluri de funcții specifice oferite de interfață de programare pentru placa de achiziție. Spațiul kernel reprezintă contextul de execuție al sistemului de operare, unde are loc managementul proceselor și firelor de execuție, planificarea și managementul întreruperilor. Prin acest nivel se poate accesa nivelul hardware, însă important de menționat este că există moduri privilegiate de acces pentru nivele. Un aspect important pentru analiza facilităților de timp real oferite de sistemul de operare se referă la preemptibilitate și latență. Preemptibilitatea este capacitatea sistemului de operare de a întrerupe execuția unui task în favoarea altuia cu o prioritate mai mare, fiind necesară pentru a satisface politica planificatorului și de a putea asigura inițierea unei schimbări de context. Sunt surprinse în continuare, în figura 22, patru scenarii posibile care pot apărea în operare și sunt de interes în analiza satisfacerii cerințelor de timp real.[52,53]



scenariul 1



scenariul 2



scenariul 3 și scenariul 4

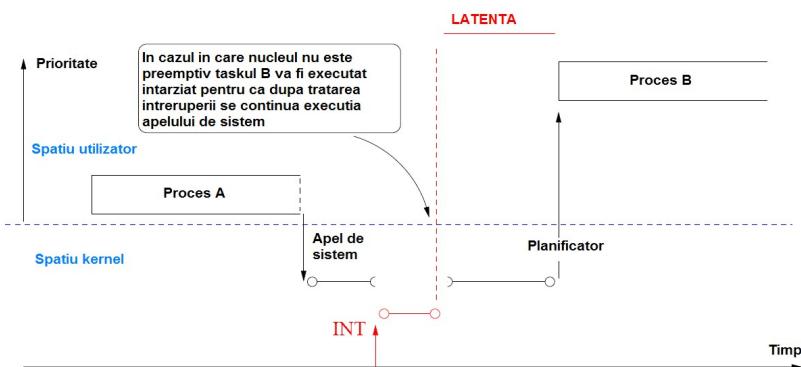


Figura 22 Scenarii de execuție în nucleul sistemului de operare

Scenariile anterioare au avut rolul de a pune în evidență limitările sistemului de operare Linux de uz general în ceea ce privește performanțele de timp real, care se reflectă de fapt în optimizarea debitului de tratare a aplicațiilor în detrimentul timpului de răspuns, utilizarea unui număr mare de puncte de preemptivitate care incetinesc sistemul de operare și gestiunea ne-deterministică a memoriei și perifericelor. Sistemul de operare în timp real încearcă să obțină o minimizare a latenței globale a sistemului.[57] Această latență globală este o sumă a căror termeni sunt descriși în figura 23,

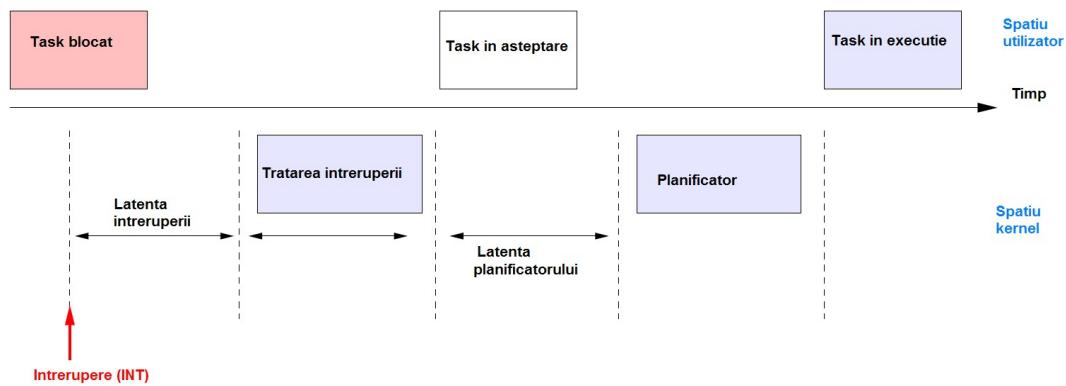


Figura 23 Latență globală a sistemului de operare

Pentru a face trecerea la un sistem de operare în timp real în contextul utilizării Linux există mai multe posibilități. Prima abordare se referă la modificarea nucleului, prin mărirea gradului de granularitate (gradul de preemptivitate a nucleului), prin reducerea numărul de secțiuni critice și apeluri mai dese către planificator (dar mai judicios). A doua abordare care se referă și la implementarea proprie are la bază conceptul de dual-kernel sau co-nucleu, care constă în adăugarea unui nucleu de timp real distribuției Linux existente fără a-i altera funcționalitatea și bazându-se pe mecanisme de tipul virtualizării întreruperilor și unui mecanism specific pentru IPC între Linux și micro-nucleul de timp real (domenii).[63] În abordarea considerată micro-nucleul de timp real se inserează între Linux și hardware, are un planificator separat și nu depinde de secțiunile critice al Linux. La apariția unei întreruperi micro-nucleul o capturează pentru a o utiliza în rutinele sale de timp real înainte de Linux, care va receptiona doar o întrerupere virtuală, făcându-l astfel un domeniu de prioritate secundară. Micro-nucleul are timpi de comutare între contexte foarte mici, cu o latență sub 20 µs și are acces la toată funcționalitatea domeniului Linux, nemijlocit. În ceea ce privește

mecanismul de la baza acestei abordări dual-kernel istoric vorbind au existat două implementări, RTHAL (RealTime Hardware Abstraction Layer) care este la baza primelor versiuni de RTAI și ADEOS (Adaptive Domain Environment for Operating Systems) care permite partajarea resurselor hardware între mai multe sisteme de operare concurente către care a migrat și RTAI în 2003.[56] La bază, ADEOS este un nivel de abstractizare al resurselor disponibil ca un patch peste nucleul de bază Linux, care permite existența mai multor sisteme de operare (domenii) pe aceeași mașină.[49] Domeniile pot fi invizibile unul altuia însă toate sunt supervizate de ADEOS. Rolul minimal al unui domeniu este de a concura pentru a procesa evenimentele exterioare (întreruperi) sau cele interne (excepții) în concordanță cu prioritatea care i-a fost acordată. Pornind de la capacitatele sale de virtualizare ADEOS poate oferi o interfață de programare generică pentru domeniile supervizate care este independentă de arhitectura mașinii pe care operează. Structura de bază pe care se bazează ADEOS este lanțul de domenii client care concură pentru controlul evenimentelor, domeniile emițând cereri de notificare pentru întreruperi externe sau virtuale, pentru apelurile de sistem emise de aplicațiile Linux sau alte evenimente declanșate de codul executabil din nucleu. ADEOS asigură că evenimentele sunt distribuite domeniilor client în funcție de prioritatea statică din sistem asigurând o livrare predictibilă și bine sincronizată.[50] Toate domeniile active sunt puse într-o coadă conform cu prioritățile lor formând un pipeline abstract. Astfel evenimentele care apar (incluzând întreruperile) sunt introduse în capul cozii (preluate de către cel mai prioritar domeniu) și avansează către capătul cozii (domeniul cu prioritatea cea mai scăzută). În continuare este redată o imagine sintetică a sistemului ADEOS unde mai multe domenii partajează evenimentele ce parcurg pipelineul abstract,

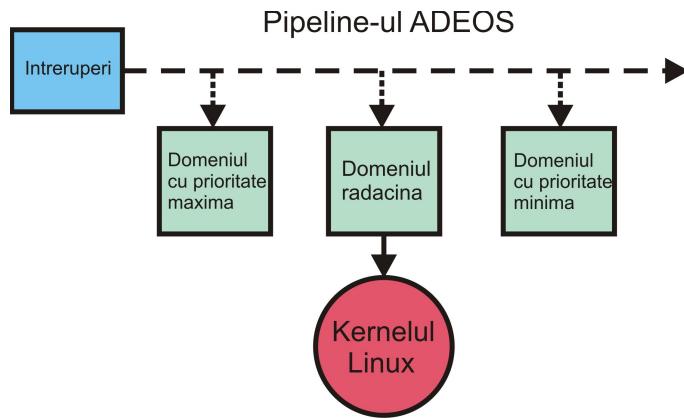


Figura 24 Reprezentarea pipelineului ADEOS

În figura 24 kernelul linux ocupă locul domeniului rădăcină întrucât pe baza kernelului Linux se instalează celelalte domenii, efectiv prin încărcarea unor module. Pentru a realiza o distribuire a întreruperilor într-un mod prioritizat ADEOS implementează schema protecției optimiste a întreruperilor dezvoltată de Stodolsky, Chen și Bershad, prezentată în figura 25. [50] Astfel fiecare etaj al pipelineului ocupat de un anumit domeniu poate fi dezactivat, în sensul că următoarea întrerupere nu va mai fi livrată măgerului de întreruperi a domeniului și nu se va mai realiza propagarea către domeniile de joasă prioritate. Întreruperile care apar între timp sunt acumulate în logul de întreruperi al domeniului și vor fi tratate ulterior când etajul va fi activat printr-o operație de sincronizare. Prin această metodă domeniile își protejează secțiunile critice de preemptia cauzată de rutinele de tratare a întreruperilor. După ce un domeniu și-a încheiat procesarea întreruperilor se emite un apel special către ADEOS care realizează o alocare a procesorului următorului domeniu ca prioritate din pipeline.

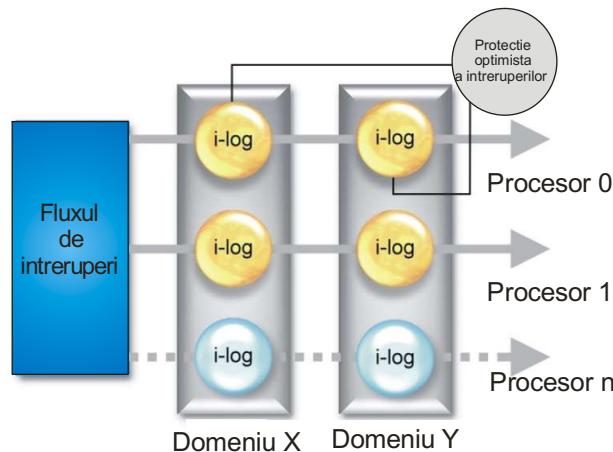


Figura 25 Protecția optimistă a întreruperilor

Întreruperile nu sunt singurul tip de evenimente care pot parurge pipelineul, întrucât însuși kernelul Linux sau aplicațiile care rulează pot genera astfel de evenimente. Ideea de bază este că evenimentele sunt notificări sincrone a unor capcane (traps), excepții sau altor acțiuni desfășurate de către kernelul Linux. În continuare este prezentat specificul RTAI ca soluție pentru dobândirea caracteristicilor de timp real pentru aplicația dezvoltată. RTAI este alcătuit din cinci părți complementare. Prima componentă este nivelul de abstractizare al hardwareului (HAL) care oferă o interfață pentru accesul la hardware și care redă suportul funcțional pentru Linux cu capacitatea hard real time. A doua componentă este nivelul de compatibilitate Linux care oferă o interfață către sistemul de operare Linux, și astfel oferind RTAI posibilitatea de a fi integrat în managementul taskurilor Linux, fără a influența operarea

Linux. A treia componentă este nucleul de operare în timp real care introduce funcționalitatea hard real time pentru planificarea taskurilor, tratarea intreruperilor și securitate. A patra componentă este LX/RT care oferă suport pentru soft și hard real time în spațiul utilizator printr-o interfață de programare (API) pentru a oferi o funcționalitate similară apelurilor de funcții din spațiul kernel și din spațiul utilizator și un IPC simetric pentru cele două moduri. Ultima componentă a RTAI sunt pachetele de funcționalitate extinsă , care cuprind drivere, interfețe de programare pentru diverse dispozitive, watchdogs software.[52] În această categorie intră și o altă componentă importantă a aplicației dezvoltate și anume setul de drivere, Comedi, pentru I/O în timp real utilizând placă de achiziție. În ceea ce privește managementul taskurilor și planificarea (scheduling) RTAI oferă o întreagă varietate de taskuri de timp real și planificatoare, oferind atât taskuri care ajung în cozile planificatorului sistemului de operare dar și unități de execuție care nu sunt planificabile (timere, tasklets, ASRuri).[52,53] În ceea ce privește configurațiile de planificare RTAI oferă alternative complementare, pentru planificare uni/multi-procesor, pentru sisteme multiprocesor simetrice, cu planificare periodică și one-shot, o planificare bazată pe priorități statice sau Round Robin pentru hard real time. În ceea ce privește metodele de comunicare inter-proces (IPC) RTAI oferă toată gama de primitive pentru sincronizare, semafoare, mutexuri, spinlockuri, variabile condiționale și flaguri. În ceea ce privește transferul, fluxul de date între procese RTAI implementează comunicarea prin mesaje, mailboxuri, cozi de mesaje POSIX, FIFOuri, memorie partajată și remote procedure calls, iar în ceea ce privește managementul memoriei RTAI asigură o implementare simetrică și un management dinamic al memoriei.[63] Un aspect important este capacitatea de a permite integrarea de device drivere real time cum ar fi Comedi.

3.1.2 Utilizarea setului de drivere pentru operații I/O real time, Comedi

Comedi s-a dezvoltat ca un proiect open-source orientat pe dezvoltarea de drivere, instrumente și librării care să ofere suportul pentru diferite plăci de achiziție și sisteme de achiziție de date pentru efectuarea de operații I/O cu semnale analogice sau digitale, generare și măsurare de frecvențe, numărare impulsuri etc.[58] Proiectul s-a dezvoltat sub forma unor module kernel și a unei librării de funcții pentru programarea în spațiul utilizator. Astfel Comedi este colecția de drivere pentru o mare varietate de plăci de achiziție de date, driverele având un nucleu comun pentru funcționalitate generică și module individuale de nivel scăzut

specific fiecărui dispozitiv. Comedilib este o librărie de funcții destinată spațiului utilizator, pentru programarea aplicațiilor, configurare și calibrarea dispozitivelor. Kcomedilib este un modul kernel care oferă aceeași interfață oferită de Comedilib în spațiul utilizator, în spațiul kernel, fiind indicată pentru taskuri real time. Un device driver este o componentă software care realizează interfațarea cu anumit hardware, realizând conversia din funcții de nivel superior emise de utilizator în comenzi dependente de dispozitiv. Plăcile de achiziție de date suportate sub Comedi lucrează cu tipuri diferite de semnale: intrări analogice, ieșiri analogice, intrări digitale, ieșiri digitale, intrări pe counter/impuls, ieșiri pe timer/impulsuri. Lucrul cu semnale digitale nu necesită un efort prea mare sub Comedi, configurările necesare referindu-se la numărul canalului (uneori adresa pe bus) și direcția, intrare sau ieșire. Semnalele analogice sunt ceva mai greu de utilizat. Tipic un canal de achiziție analogic poate fi programat pentru a genera sau citi o tensiune între două praguri superior și inferior (în cazul de față -10V | +10V) și utilizând suportul hardware al plăcii se poate seta eşantionarea unor anumite canale într-o anumită ordine și de a memora datele pe placă, apoi se poate utiliza DMAul sau o rutină de tratare a întreruperilor pentru a realiza un dump al datelor într-o anumită zonă de memorie. În cazul semnalelor bazate pe impulsuri (semnale de la encodere, timere, countere) se adaugă, față de tratarea semnalelor tipic digitale, anumite specificații temporale semnalului. Comedi organizează hardwareul utilizând o ierarhie generică pornind de la canal (Channel), care este componenta hardware de la nivelul cel mai scăzut, reprezentând proprietățile fizice ale unui singur canal de date (plaje de valori, tensiune de referință, polaritate, factor de conversie între mărimi fizice etc.); sub-dispozitivul (sub-device) caracterizează un set de canale cu funcționalitate identică implementate fizic în același circuit (un set de 16 ieșiri analogice) menținând informații despre numărul de canale și tipul lor; dispozitivul (device) care este un set de sub-dispozitive implementate fizic pe aceeași placă.^[64] De exemplu placa utilizată în cadrul proiectului National Instruments PCI 6024E, are un sub-dispozitiv cu 16 canale de intrare analogice, un sub-dispozitiv distinct cu 2 ieșiri analogice și un al treilea sub-dispozitiv cu 8 I/O digitale. Fiecare obiect dispozitiv încapsulează informații cu privire la codul de identificare al producătorului, identificatorul dat de sistemul de operare, numărul de sub-dispozitive etc. În ceea ce privește funcționalitatea pentru achiziția de date oferită de Comedi aceasta se centreză pe lucrul cu canele sau seturi de canale. Astfel putem avea o achiziție singulară (single acquisition) utilizând funcții speciale care relizează o singură operațiune de achiziție pe un canal specificat (ex `comedi_data_read()`, `comedi_data_write()`, `comedi_dio_read()`,

comedi_dio_write()). Altfel putem avea o instrucțiune sau o listă de instrucțiuni care realizează (dacă e posibil) mai multe achiziții de date pe un canal specificat într-un mod sincron (funcția blocându-se până la terminarea achiziției, ex comedi_do_insn(), comedi_do_insnlist()). În cazul în care se dorește obținerea informațiilor de pe un set de canale diferite cu anumită secvență și temporizare se utilizează scanările (scans) care sunt implicit integrate în comenzi Comedi și care sunt prezentate în figura următoare,

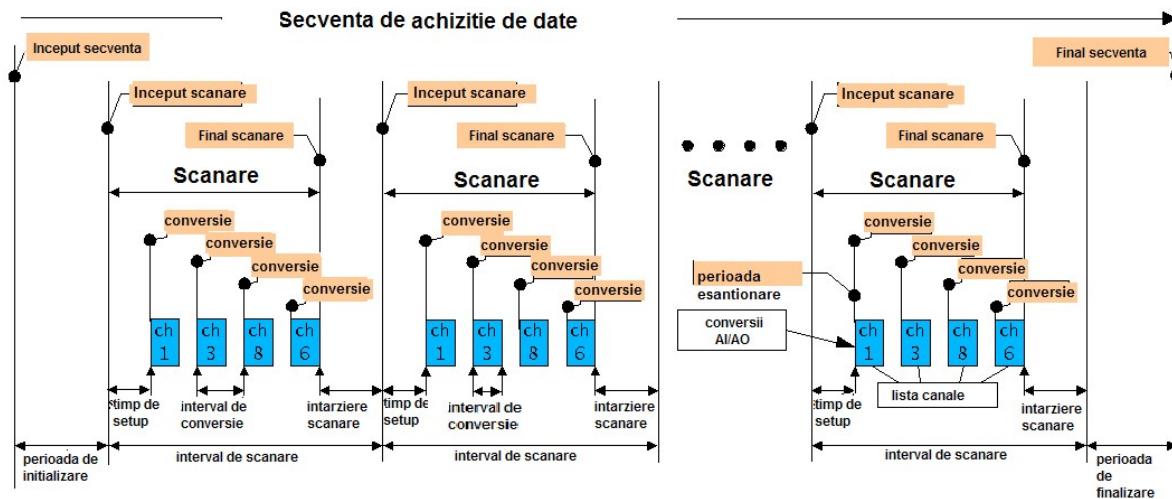


Figura 26 Secvență de achiziție de date

O comandă este un set de scanări pentru care parametrii au fost setați, stabilind momentul de declanșare și finalul achiziției de date. La apelul unei funcții (comedi_command()) se inițiază o achiziție asincronă de date, susținută prin mecanisme precum DMA și utilizarea rutinelor de tratare a întreruperilor. Pentru comenzi se poate seta evenimentul care să poată declanșa (trigger) și să îsta achiziția datelor. Pe lângă capacitatea de accesare a funcționalităților plăcii de achiziție Comedi oferă posibilitatea realizării unei interogări a dispozitivelor instalate pentru a afla informații despre parametrii prestabili și informații de stare necesare. Un alt aspect care este surprins în interfața de programare a Comedi este managementul bufferelor și caracterul event driven al achizițiilor de date, totul sub posibilitatea implementării unor mecanisme de securitate pentru a asigura execuția unor operații atomice asupra zonelor critice de cod sau de structuri de date. În continuare este prezentată arhitectura sistemului distribuit de control și monitorizare al robotului.

3.2 Prezentarea nivelelor aplicației de control tolerant la defecte. Aspecte generale de proiectare a aplicației

Arhitectura sistemului software dezvoltat e formată din aplicația C++ care rulează pe unitatea de procesare embedded sub Linux-RTAI și Comedi care oferă accesul direct la senzorii și actuatorii robotului prin placă de achiziție și care prin intermediul unor threaduri server (de control și de date) oferă o interfață de control aplicației client Java sub Linux permisându-i să activeze execuția fazelor și să primească informații direct de la senzorii robotului. Aplicația Java are rolul de a asigura activarea sau dezactivarea fazelor de execuție ale robotului prin utilizarea unui instrument care-i permite să determine o execuție secvențială sau paralelă a taskurilor robotului, numit Grafct. Structura se poate extinde cu un client Java cu drepturi restrânsă cu rolul doar de a monitoriza execuția și de a primi informații despre starea sistemului, însă la momentul actual nu a fost implementat efectiv.

Arhitectura concretă a aplicației este descrisă în figura următoare,

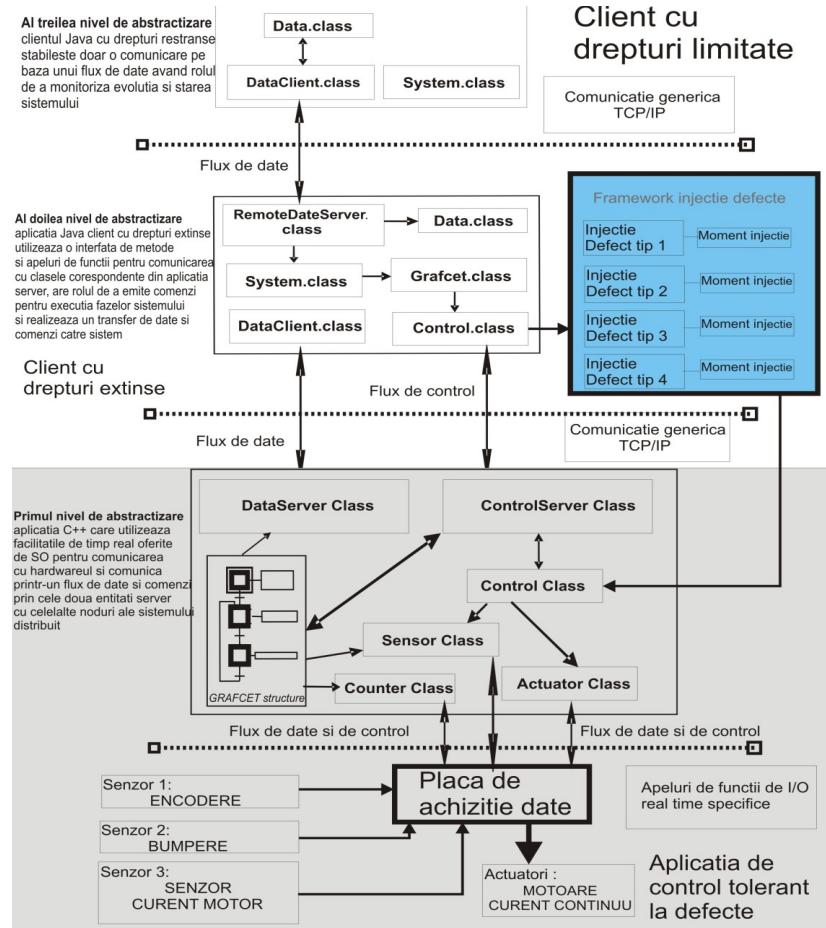


Figura 27 Arhitectura generală a aplicației dezvoltate

Urmând o vedere de ansamblu aplicația dezvoltată se bazează pe un tipar de dezvoltare orientat obiect care conferă flexibilitate și extensibilitate. În continuare sunt descrise nivelele individuale ale aplicației.

3.2.1 Nivelul de bază, de interfață cu hardware-ul

Interfața cu hardwareul, în sprijinul senzorii și actuatorii robotului, se realizează la nivelul aplicației server ce rulează pe robot. Un scenariu tipic de funcționare pornește cu citirea informațiilor de la senzori și în funcție de valorile citite se calculează o comandă pentru actuatori, citirile și comenzi fiind efectuate prin intermediul plăcii de achiziție.[60] Abstractizarea relațiilor cu senzorii și actuatorii s-a concretizat în designul aplicației prin crearea claselor Counter, Sensor și Actuator. Astfel la nivel funcțional clasa Sensor are rolul de a oferi un mijloc de a accesa valorile de la senzorii robotului. Datorită faptului că rețeaua de senzori este eterogenă clasa generică a fost extinsă cu două clase specifice pentru lucru cu senzori care oferă informație analogică sau senzori care oferă informație digitală. Astfel la nivel funcțional clasa generică Sensor se bazează pe o funcție accesoriu de tip get() care este apoi redefinită în subclase. În cazul în care senzorul conectat oferă informație analogică la ieșire acesta este definit în aplicație ca un obiect AnalogSensor, care încapsulează informație despre dispozitivul Comedi la care este conectat, canalul, plaja de valori ale semnalului și o referință de tensiune. Întrucât senzorii care se pot conecta pot măsura diverse mărimi fizice funcția get() citește informația de pe canalul selectat cu un apel al funcției Comedi comedi_data_read() apoi apelează o funcție specială care realizează de fapt o conversie din mărimea fizică măsurată în tensiune, physicalDataToVolts(), care are rolul de a stabili dependența de pe caracteristica statică între mărimea de intrare și tensiune. În cazul de față am creat două instanțe ale clasei AnalogSensor, pentru interfațarea cu senzorul de curent de pe driverul de putere pentru motoarele de curent continuu, cu rol în monitorizare și siguranță și pentru monitorizarea tensiunii pe baterie. În cazul în care senzorul conectat oferă o informație digitală se instanțiază clasa DigitalSensor. Astfel obiectul creat încapsulează informații despre canalul plăcii de achiziție unde este conectat, identificatorul dispozitivului Comedi și informație specifică de configurare pentru intrare sau ieșire. Pentru a accesa valorile primite de la senzor se utilizează o metodă accesoriu get() care returnează valoarea binară primită de la senzor. În cazul robotului am creat două instanțe senzor digital pentru cele două bumpere, frontal și spate, care în starea necomprimată dau 1 logic pe intrarea plăcii,

iar la comprimare comută în 0 logic, lucru care este captat de program și se interpretează ca o măsură de siguranță dând comandă PWM 0 către motoare. Deși face parte funcțional din clasa Sensor, clasa Counter a fost dezvoltată în paralel ca o clasă distinctă pentru interfațarea cu encoderele. Astfel un obiect Counter încapsulează informații privind dispozitivul Comedi (un câmp generic utilizat la toate clasele pentru a asigura compatibilitatea cu ierarhia Comedi prezentată anterior) și canalul pe care este conectat encoderul. Datorită faptului că placa de achiziție dispune de două intrări counter implementate hard, nu a mai fost nevoie de a dezvolta soft funcții speciale de numărare a trenurilor de impulsuri provenite de la encodere. Pentru a asigura un control eficient al operației de citire a encoderelor robotului am implementat funcții de reset, pentru a preveni overflow-ul counterelor de 12 biți, funcție de setare a direcției de numărare și a setării punctului de 0 pentru encoderul incremental și în final o funcție accesor getCount() pentru a prelua numărul de impulsuri din bufferul counterului în aplicație și pentru a utiliza această valoare la calculele de odometrie, pentru determinarea poziției robotului. Pornind de la ideea construirii unei structuri minimale de robot mobil, sistemul senzorial este limitat dar suficient pentru a asigura controlul și a implementa facilitățile de redundanță analitică pentru diagnoză. În ceea ce privește sistemul de locomoție comenziile către actuatori sunt emise prin intermediul funcțiilor clasei Actuator. În acest caz similar cu senzorii, putem avea două tipuri de actuatori, actuatori comandați prin semnale digitale (DigitalActuator) sau actuatori comandați prin tensiuni continue (AnalogActuator). În cazul actuatorilor analogici obiectele corespunzătoare încapsulează informații despre canalul pe care sunt conectați, gama de valori pentru comanda admisibilă și o metodă accesor de tip set() cu rolul de a emite o comandă către actuator utilizând funcții din interfață de programare Comedi. Astfel după ce un obiect actuator a fost creat se dorește comanda acestuia, astfel se apelează funcția set(), care are rolul de testa că valoarea trimisă ca parametru este în gama de valori admisibilă, se utilizează o funcție de tipul voltsToPhysicalData care are rolul de a asigura compatibilitatea între comanda în tensiune și intrarea actuatorului, apoi se utilizează o metodă de tip comedi_data_write() care scrie valoarea comenzi în bufferul de ieșire pentru un anumit canal al plăcii de achiziție și este emis către actuator. În cazul în care avem un actuator digital se instanțiază clasa DigitalActuator specificând informații privind dispozitivul Comedi, canalul de conectare al actuatorului și informația de configurare pe intrare/ieșire. Pentru a emite comanda se apelează funcția set() specificând valoarea digitală care se trimit. În structura robotului există doi actuatori analogici, motoarele de curent continuu, care sunt comandate PWM.

În acest caz la nivelul aplicației se calculează comanda sub forma unei tensiuni analogice, care este emisă pe canalul corespunzător și preluată de driverul de putere care transformă această tensiune continuă în semnal PWM (Pulse Width Modulation) care este trimis motoarelor. În continuare este prezentată arhitectura internă a aplicației server ce rulează pe robot, urmând că în paragrafele următoare să se face o descriere a modului de funcționare al aplicației.

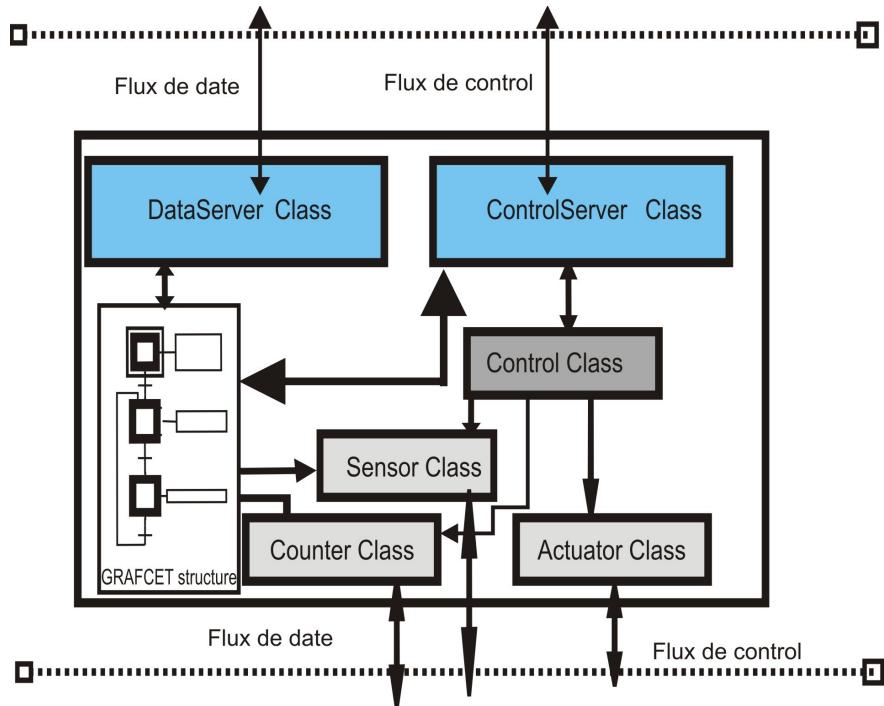


Figura 28 Arhitectura internă a aplicației ce rulează pe robot

Ponind de la arhitectura aplicației primului nivel se trece în continuare la prezentarea suportului pentru implementarea algoritmului de control și diagnoză a defectelor.

3.2.2 Nivelul de implementare a algoritmilor de conducere, diagnoza și toleranță a defectelor

Studiul nivelelor funcționale diferă de studiul nivelelor concrete de implementare în sensul că suportul pentru implementarea controlului și diagnozei se realizează concret prin conlucrarea celor două nivele ale aplicației, aplicația server C++ și aplicația client Java. La nivelul aplicației ce rulează pe robot se implementează efectiv taskurile de control și diagnoză, însă setarea parametrilor specifici de operare și activarea fazelor se realizează la

nivelul aplicației client. Pentru a descrie aplicația la nivelul robotului se impune prezentarea aspectelor specifice ale instrumentului utilizat pentru execuția taskurilor, care redă de fapt modul de evoluție dorit al sistemului. Grafcet (Graphe Fonctionnel de Commande Etape - Transition) este un model folosit foarte mult pentru sistemele discrete logice și a devenit unul din instrumentele de bază pentru modelarea sistemelor de conducere bazate pe evenimente. [62] Elementele specifice Grafcet au fost abstractizate și s-au concretizat sub forma unor clase. Clasa Stage modeleză o etapă activă iar operarea sistemului se concretizează sub forma unei succesiuni de etape care redau de fapt traectoria de stare a sistemului. Se precizează pentru fiecare etapă acțiunea care se va efectua, doar atunci când etapa este activă. Acțiunile pot fi însoțite de condiții care vor determina execuția acțiunii doar dacă condiția este îndeplinită și etapa este activă. Clasa Stage are ca subclase clasele Action, Cycle și Phase. Schimbarea comportamentului sistemului este marcată prin tranziții de la o etapă la alta, o tranziție fiind validată dacă toate etapele precedente sunt activate și dacă condiția logică (receptivitatea) este validată. Pentru a descrie mai bine funcționalitatea claselor specifice Grafcet sunt descrise cele două posibilități de execuție ale taskurilor asociate cu operarea sistemului.

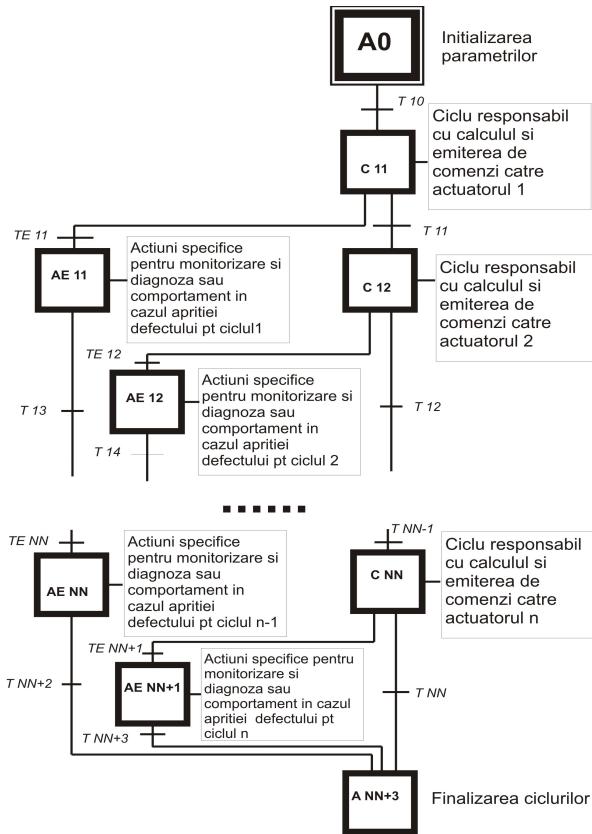


Figura 29 Execuția etapelor în fază serie

După cum se poate observa în faza serie acțiunile și ciclurile au loc secvențial. În implementare, fiecare etape îi corespunde o acțiune, în cazul de față prima acțiune este de initializare a parametrilor, apoi se introduc ciclurile care sunt de fapt elementele care oferă suportul pentru execuția ciclică a algoritmului de control, apoi avem tranziții către acțiuni specifice de monitorizare și diagnoză, care în structura serie se execută după execuția fiecărei iterații a ciclului. Această strucțură nu este însă folosită în cadrul aplicației de control al robotului întrucât în acest caz monitorizarea și controlul trebuie să fie realizate în paralel, iar acțiunile specifice de monitorizare și diagnoză se vor efectua concurrent cu controlul. Structura de Grafcat generică paralelă pentru 2 actuatori de la care am pornit în implementarea curentă este redată în figura următoare.

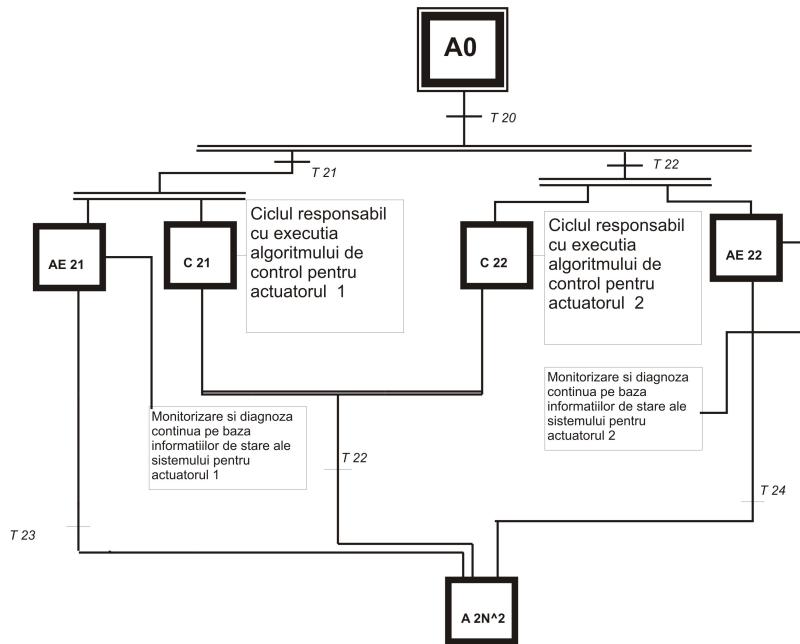


Figura 30 Structura generică pentru execuția etapelor în faza paralelă

Pornind de la structura generică prezentată în figura anterioară în implementarea curentă am introdus calculul comenzii cu algoritmul sliding mode pentru ambele motoare (ambii actuatori) în același ciclu, iar în paralel cu execuția taskului de control are loc și o monitorizare continuă a comportamentului robotului pe baza informațiilor de stare de la senzori (figura 31).

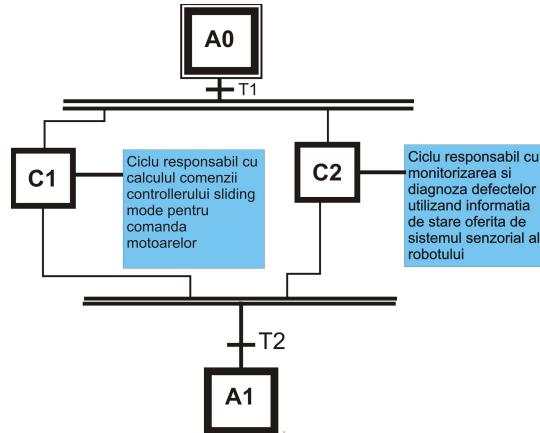


Figura 31 Grafcet paralel implementat în aplicația curentă

Astfel prima etapă este inițializarea parametrilor robotului, parametrilor de control și parametrilor pentru modulul de detecție, apoi identificarea și în final reconfigurarea comenzi în prezența defectelor. După activarea tranziției initiale se activează etapa de execuție paralelă a ciclului pentru control și a ciclului pentru monitorizare și diagnoză. Apoi prin activarea tranzițiilor finale se activează execuția etapei de final. O altă clasă importantă a aplicației server este clasa Control care are rolul de a seta parametrii pentru execuția etapelor Grafcetului și are acces prin funcții specifice de get(), set() la parametrii de execuție și control, implementează interfața exportată către aplicația Java pentru controlul execuției prin metode de tipul exe_phase(), stop() și are un rol important și în stabilirea arhitecturii sistemului, cu privire la numărul și tipul senzorilor și actuatorilor (tabele de senzori și actuatori). Clasa Control este responsabilă și de managementul operațiilor de timp real utilizând apele de funcții specifice din interfața de programare a RTAI și Comedi. Prezentarea sintetică a facilităților și funcțiilor specifice RTAI utilizate poate fi realizată prin figura următoare.[56]

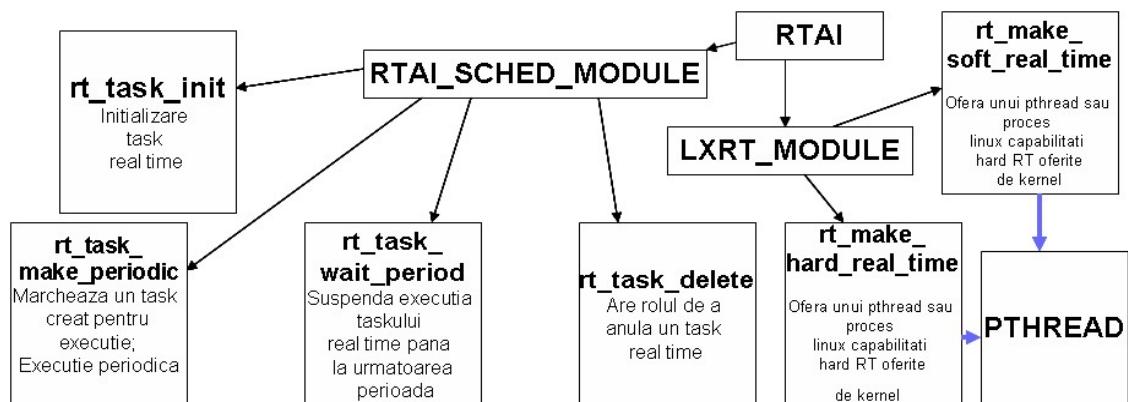


Figura 32 Facilități de lucru în timp real oferite de RTAI utilizate în aplicație

Înainte de a trece la următorul nivel al aplicației mai sunt prezentate câteva aspecte legate de implementare. Astfel clasa Thread oferă suportul pentru execuția în mediu multithreading, prin utilizarea unor metode specifice de tip debug () (cu rolul de a determina o localizare mai eficientă a erorilor în cazul executiei unei etape sau efectuarii unei tranzitii), metode de tip setName() sau getName() cu rolul de a seta/returna numele threadului activat și utilizează metode de asigurare a siguranței în mediul multithreaded. Clasa Action este o subclăsa a clasei Thread și oferă facilitatea de emitere a comenzi de control pentru elementele de execuție (metode de identificare formală a unei acțiuni și comanda specific), are rolul de a interpreta etapele grafctului. Clasa Stage extinde clasa Thread și implementează conceptul de bază al conducerii prin metoda Grafct, implementând mecanisme pentru managementul (add(), start(), stop()) etapelor. În final se pune accentul pe implementarea clasei Cycle care determină de fapt lucrul cu facilitățile hard real time aduse de soluția aleasă. Astfel în alcătuirea ei se pot distinge metoda de tipul run() care folosește metode specifice inițializării agentului de timp real și a creării taskului corespunzător folosind metode specifice RT_SCED+Pthread și metode de setare/returnare a perioadei de execuție cu evaluarea condițiilor de finalizare a execuției și managementul taskurilor de control al buclelor. Descrierea clasei Cycle se va realiza și ulterior în descrierea modului de implementare a celor două taskuri de control și diagnoză.

3.2.3 Nivelul superior al comunicației între nodurile sistemului distribuit

Nivelul de comunicație asigură schimbul fluxurilor de date și comenzi între nodurile sistemului distribuit.[59] În ceea ce privește implementarea nivelului de comunicație între aplicații s-au implementat două clase server, ControlServer și DataServer. Clasa ControlServer are rolul de a transfera pachete de date care conțin comenzi de la client, în ceea ce privește activarea sau sistarea activității sistemului, iar clasa DataServer are rolul de a asigura un flux continuu de date pentru transferul parametrilor robotului către aplicația client. La nivel de implementare clasa ControlServer se bazează pe lucrul cu socketuri și la crearea unui obiect se utilizează informații privind adresa IP a clientului și portul de comunicație. Metodele specifice clasei sunt metodele de tip connect(), care așteaptă în background conectarea unui client și returnează o notificare, metode de tip serve_requests() care au rolul de a prelua informația din pachete și de a o transforma în comenzi explicate sau cereri către

celelalte clase, metode de tipul serve_client() în care în funcție de comanda primită de la client se accesează anumite informații sau se activează execuția anumitor activități (start robot, stop robot) care de fapt redau legătura prin intermediul clasei Control cu elementele specifice de implementare ale aplicației de control. Serverul de date este implementat prin clasa DataServer și are rolul de a construi buffere pentru a stoca informații legate de evoluția sistemului în operare și de a le trimite aplicației client. Sunt utilizate metode care asigură conectarea mai multor clienți connect() și metode specifice de transmitere de date, generic intitulate send(), care pot implementa transferul datelor formatate din buffere realizându-se și un management al cererilor de la clienți. În continuare este redată arhitectura clientului Java accentul căzând pe clasele care asigură comunicația cu robotul.

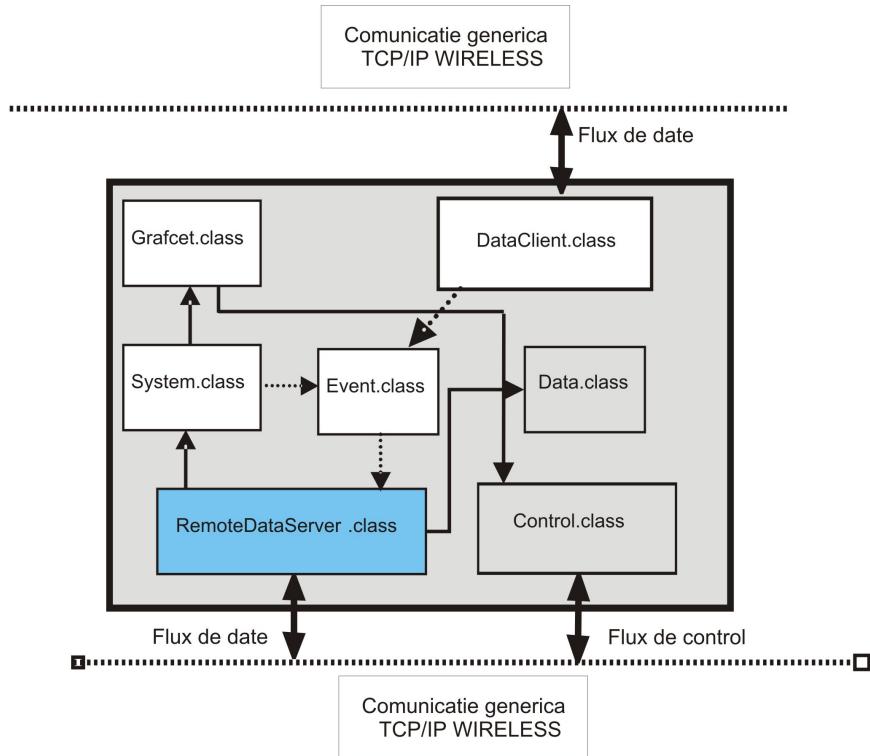


Figura 33 Arhitectura aplicației client Java

Aplicația client Java are o structură asemănătoare cu a aplicației de pe robot pentru a asigura compatibilitate la nivel funcțional. Astfel făcând referire la nivelul de comunicație, au fost implementate clasele RemoteDataServer, care comunică printr-un flux de date cu clasa DataServer a aplicației ce rulează pe robot și are rolul de a prelua informațiile de stare și

parametrii curenți ale mărimilor de interes pentru robot și de a le transmite local clasei System și clasei Data. Mai departe clasa Data are rolul de a prelua și a stoca informațiile primite cu scopul de a emite loguri și alte surse de observație valide ce descriu întreaga perioadă de operare a robotului. Clasa Systm are rolul de a îngloba structura actuală a sistemului, numărul de senzori și actuatori și tipul lor pentru a putea prelua și transmite valorile parametrilor din bufferele de recepție de la robot către clasa Grafct. Clasa Grafct este similară cu cea dezvoltată în aplicația server cu diferența că aici are loc stabilirea efectivă a structurii Grafctului pentru execuție. După stabilirea parametrilor și structurii Grafct pentru execuție parametrii sunt trimiși clasei Control care comunică cu clasa ControlServer printr-un flux de comenzi care se reflectă de fapt în comenziile individuale de operare ale robotului. Clasa DataClient are rolul de a asigura comunicarea cu clientul cu drepturi restrâns, care nu a fost implementat în faza actuală. În implementarea actuală clientul java este construit pe baza unei interfețe utilizator GUI sub forma arhive executabile Java (jar) care are rolul de a activa sau opri activitatea robotului și de a injecta defectele în cadrul operării tolerante la defecte. Datorită faptului că nu a fost posibilă introducerea unor defecte on-line, în timpul operării robotului a fost aleasă dezvoltarea unei metode de injecție software a defectelor prin alterarea anumitor parametrii ai modelului matematic al robotului. Astfel pe lângă comenziile transmise către serverul de control al aplicației server se transmit pachete care conțin variabile indicatori pentru declanșarea anumitor defecte în sistem. O altă posibilitate oferită de aplicația client este de a realiza un control la nivel de execuție pas cu pas a algoritmului de control, însă în cazul de față nu se impune. Alegerea tipului de execuție a taskurilor (serială / paralelă) se poate realiza direct din interfața utilizator și de asemenea se pot încărca fișiere care să introducă o structură Grafct diferită în funcție de problemă. În acest caz însă taskurile de control și monitorizare și diagnoză sunt concurente și există posibilitatea de a monitoriza grafic evoluția anumitor variabile ale sistemului, accesând informațiile stocate în clasa Data, provenite de la robot(opțiune încă în lucru).

În capitolul de față a fost făcută o descriere funcțională a aplicației dezvoltate la nivelul claselor. S-au prezentat aspecte privind structura ierarhică și dependențele între clasele specifice fiecărui nivel. Pentru mai multe detalii și un studiu direct al implementării se poate consulta listingul aplicației din anexă.

Capitolul 4

Aspecte specifice ale implementării controlului tolerant la defecte în timp real a robotului mobil

Aspecte specifice ale implementării controlului tolerant la defecte în timp real a robotului mobil

Obiectivul controlului tolerant la defecte în cazul robotului mobil a fost impus pentru o gamă largă de motive de ordin economic dar și din aspecte legate de siguranță, fiabilitate și robustețe. Pornind de la structura flexibilă de control distribuit în timp real dezvoltată în primă fază, am adăugat un nivel de control sliding mode, robust și eficient pentru atingerea obiectivului operării în regim trajectory tracking a robotului și un nivel superior pentru diagnoză, bazat pe model cu rolul de a detecta, identifica și apoi reconfigura prin intermediul nivelului de control legea de comandă către robot. Concret, clasa Ciclu a aplicației dezvoltate are rolul de a asigura execuția în timp real în mediu multithreading a taskului de control și a taskului de diagnoză, care în structura de Grafctcet asignată pentru aplicația curentă, corespund celor două cicluri concurente. Taskul de diagnoză se bazează pe utilizarea unui banc de filtre Kalman pentru estimarea stării. Fiecare filtru încorporează modelul cinematic al robotului și realizează o estimare pentru operarea în prezență unui anumit tip de defect. În continuare sunt prezentate aspectele specifice taskului de control și diagnoză realizând și o analiză a rezultatelor obținute, pentru un benchmark de 4 defecte, la discriminarea a două clase de defecte.

4.1 Nivelul task-ului de control al robotului în regim de trajectory tracking

Taskul de control al robotului în regim de trajectory tracking se realizează la o perioadă de eşantionare de 200ms. Perioada de eşantionare a fost aleasă pentru a permite obținerea unor rezultate valide în ceea ce privește sistemul odometric (date valide și consistente). Taskul de control se bazează pe sinteza unei structuri de conducere în cascadă cu o buclă internă pentru controlul turăției motoarelor de curent continuu, care se execută la o perioadă de eşantionare de 50 ms (reacția fiind dată de sistemul de odometrie) și o buclă externă pentru controlul poziționării robotului cu un controller sliding mode ce sintetizează o lege de comandă la fiecare 200ms (reacția fiind dată prin sistemul odometric). Structura de conducere în cascadă a robotului în regim de trajectory tracking este prezentată în figura următoare.

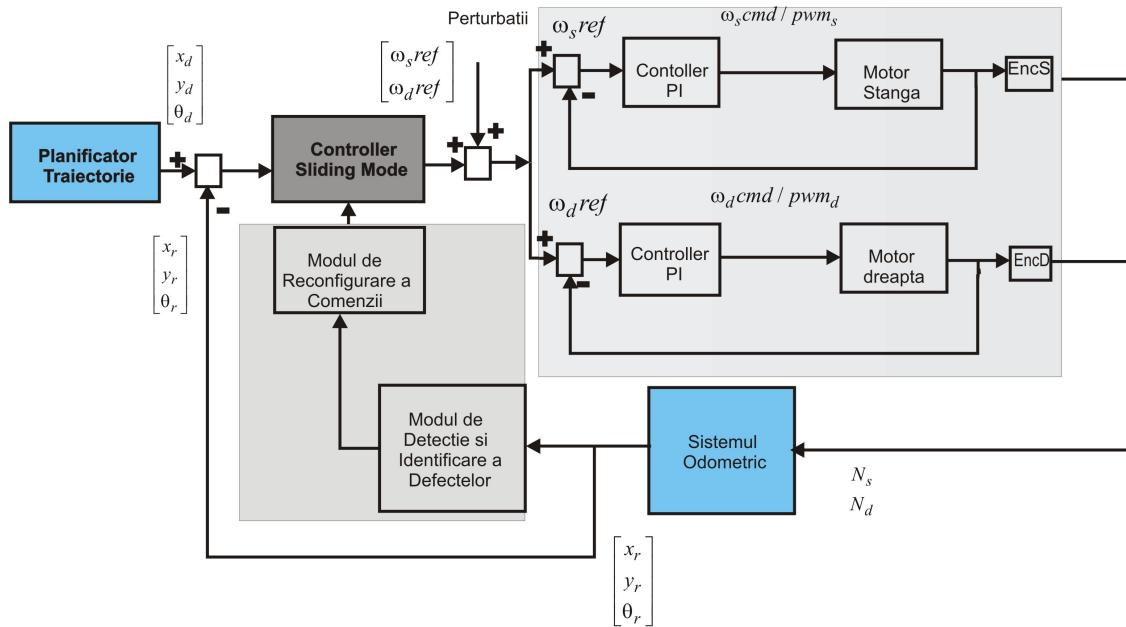


Figura 34 Structura de conducere în cascadă a robotului

4.1.1 Sinteza buclei interne pentru reglarea turației actuatorilor utilizând controlere de tip PI. Analiza performanțelor

Bucile de reglare interne realizează controlul turației motoarelor de curent continuu ale robotului. Referința pentru buclele celor două motoare este dată de controllerul sliding mode sub forma unei viteze unghiulare. Controllerul sliding mode oferă la ieșire o comandă sub forma unei viteze unghiulare și a unei viteze liniare pentru robot, care sunt însă utilizate la calculul vectorului de referințe pentru motoare (prin transformare cinematică inversă). Conversia se realizează utilizând relațiile următoare.

$$\begin{cases} \omega_s \text{ref} = (v_{c_SM} - L \frac{\omega_{c_SM}}{2})r \\ \omega_d \text{ref} = (v_{c_SM} + L \frac{\omega_{c_SM}}{2})r \end{cases}, (57)$$

Controlurile PI sintetizează o comandă sub forma unei viteze unghiulare pentru motoare, însă pentru a comanda efectiv motoarele este necesar un semnal PWM, generat dintr-o tensiune continuă. Pentru a realiza conversia între comanda dată de controlurile PI și o tensiune s-a trasat o caracteristică statică care marchează dependența tensiune de comandă – viteză unghiulară. Astfel s-a aplicat fiecărui motor o tensiune continuă în gama 1.5V – 5V (convertit în semnal PWM cu factor de umplere corespunzător 30% - 100% de către driverul

de putere) și după atingerea regimului staționar s-a măsurat valoarea vitezei unghiulare cu relații specifice ale sistemului odometric. Relația generică de calcul determinată pentru o tensiune corespunzătoare unei viteze unghiulare, $U = f(\omega)$, este dată de relația următoare,

$$U = U_i + \left(\frac{\omega - \omega_i}{\omega_j - \omega_i} \right) (U_j - U_i), U \in [U_i, U_j], \omega \in [\omega_i, \omega_j], \quad (58)$$

Valorile pentru tensiunea U se aleg cu un pas de 0.25V în intervalul 1.5V – 5V și se tabelează alături de valoarea vitezei unghiulare corespunzătoare obținute de la robot. Un aspect important care și-a manifestat influența în proiectare și operare este modalitatea de discretizare a controllerului PI. În continuare sunt redate formulările analitice pentru metodele de discretizare care au fost testate pentru sinteza controllerelor PI, marcând faptul că pentru aceeași parametri de acordare obținuți prin stabilirea unor criterii de performanță, s-au obținut cele mai bune performanțe (reducerea timpului tranzitoriu și reducerea oscilațiilor) prin utilizarea unui extrapolator de ordinul 1 (FOH). Astfel în cele ce urmează este descrisă procedura de discretizare a controllerului PI utilizând extrapolatorul de ordin 0 (ZOH), extrapolatorul de ordinul 1 (FOH), metoda Euler, metoda diferențelor înapoi și metoda Tustin, comparând expresiile obținute și influența termenilor de eroare de la momentele anterioare în calculul legii de comandă. Regulatorul PI este format prin conectarea în paralel a unui element de tip proporțional, P, descris de următoarea funcție de transfer și ecuație de funcționare în domeniul timp,

$$G_R(s)=k, \quad u(t)=ke(t) \quad k>0, \quad (59)$$

și un element de tip integrator, I, descris de următoarea funcție de transfer și ecuație de funcționare în domeniul timp

$$G_R(s)=\frac{1}{T_i s}, \quad T_i u(t)=e(t), \quad u(t)=\frac{1}{T_i} \int_{t_0}^t e(\tau) d\tau + u(t_0), \quad T_i > 0, \quad (60)$$

astfel funcția de transfer și ecuația funcționării în domeniul timp a controllerului PI este:

$$G_R(s)=k+\frac{1}{T_i s}, \quad u(t)=ke(t)+\frac{1}{T_i} \int_{t_0}^t e(\tau) d\tau + u(t_0) \quad , \quad (61)$$

cumulând efectele celor două componente.[66] Efectul componentei P se rezumă la analiza valorilor constantei de proporționalitate, k.

Astfel pentru un k suficient de mare se poate mări viteza de răspuns a sistemului automat (micșorarea timpului de creștere). În situația în care avem un sistem care nu conține integratoare alegerea unui parametru k mare poate determina micșorarea erorii de regim staționar la intrare treptă, însă poate crește și comanda către sistem foarte mult (se impune o limitare). Un dezavantaj major al componentei proporționale constă în faptul că pentru valori mari ale lui k sistemul poate avea oscilații mari în regimul tranzitoriu care pot duce chiar la instabilitate. În ceea ce privește efectul componentei integratoare este important de menționat faptul că rolul acesteia este de a anula eroarea de regim staționar. Integratorul introduce un pol în origine care determină reducerea stabilității IMEM, semnalul de ieșire devenind mai puțin amortizat, crescând suprareglajul. În ceea ce privește comanda în cazul în care avem un timp de integrare prea mic aceasta poate crește excesiv, apărând un fenomen de saturare. Deși pentru reglarea turației motorului de curent continuu este în general utilizat un controller PID (componenta derivativă având rolul de a crește viteza de răspuns a sistemului și de a reduce oscilațiile (diminuarea suprareglajului)) în implementarea curentă am folosit un controller PI care s-a dovedit a fi suficient pentru atingerea criteriilor de performanță impuse pentru sistem. În cele ce urmează se fac următoarele notații, $K_i = K_p/T_i$ și $G_R = G_{PI}$. [67] Utilizând notația specificată pentru a realiza discretizarea cu extrapolator de ordin 0 (invariantă la treptă) considerăm funcția de transfer a regulatorului și formula de discretizare:

$$G_{PI}(s) = \frac{K_p s + K_i}{s}, G_{PI}(z) = \frac{z-1}{z} Z\left[L^{-1}\left\{\frac{1}{s} G_{PI}(s)\right\}|_{t=kT}\right], \quad (62)$$

Efectuând calcule simple, utilizând transformata Laplace inversă și manipulări algebrice obținem următoarele relații pentru reprezentarea controllerului PI discret în domeniul Z și formula de calcul a legii de comandă în ecuații cu diferențe,

$$G_{PI}(z) = K_p + K_i \frac{T}{(z-1)}, \\ u(kT+T) = u(kT) + K_p[e(kT+T) - e(kT)] + K_i T e(kT), \quad (63).$$

În ceea ce privește implementarea controllerului PI discretizat cu ZOH au fost probleme în sensul manifestării unor oscilații și a unui timp de răspuns nesatisfător, deși acordarea a

fost realizată similar cu celelalte metode. Utilizând metoda de discretizare cu extrapolator de ordin 1 (FOH) relația de discretizare este,

$$G_{PI}(z) = \frac{(z-1)^2}{Tz} Z[L^{-1}\left\{\frac{1}{s^2} G_{PI}(s)\right\}|_{t=kT}], \quad (64).$$

Legea de comandă în cazul controllerului discretizat cu FOH este redată în continuare sub forma unei ecuații cu diferențe care a fost direct utilizată în implementare,

$$u(kT+T) = u(kT) + e(kT+T)(K_p + K_i T) + e(kT)(K_i T - K_p), \quad (65).$$

În cazul abordării metodei de discretizare cu metoda lui Euler (diferențe înainte) se utilizează transformarea $s \rightarrow \frac{z-1}{T}$ și efectuând calculele obținem următoarea lege de comandă în discret,

$$u(kT+T) = u(kT) + K_p e(kT+T) + (K_i T - K_p) e(kT), \quad (66).$$

În cazul utilizării metodei de discretizare cu diferențe înapoi se utilizează transformarea

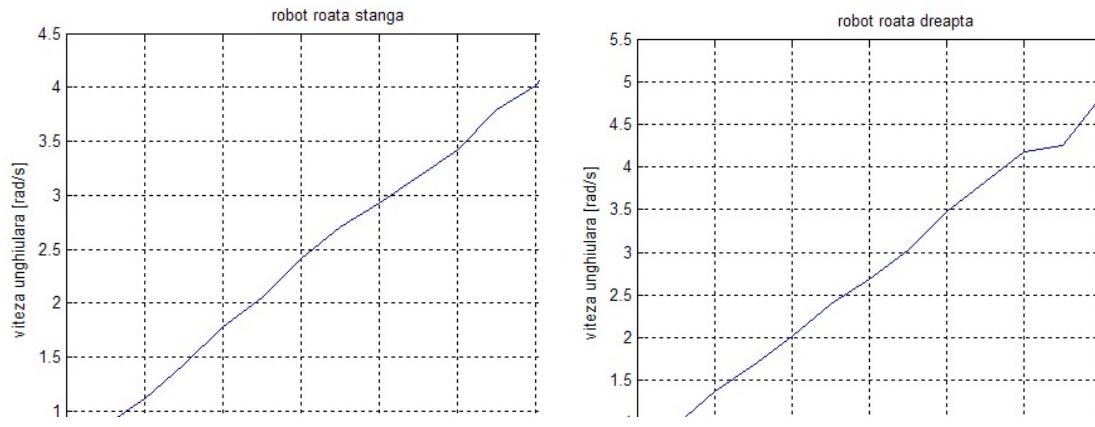
$s \rightarrow \frac{z-1}{zT}$ și efectuând calculele se obține următoarea lege de comandă în discret,

$$u(kT+T) = u(kT) + (K_p + K_i T) e(kT+T) - K_p e(kT), \quad (67).$$

În final, în cazul utilizării metodei de discretizare cu metoda Tustin se utilizează transformarea $s \rightarrow \frac{2}{T} \frac{z-1}{z+1}$ și efectuând calculele se obține următoarea lege de comandă în discret,

$$u(kT+T) = u(kT) + (2K_p + K_i \frac{T}{2}) e(kT+T) + (-2K_p + K_i \frac{T}{2}) e(kT), \quad (68).$$

În ceea ce privește obținerea caracteristicii statice pentru extragerea dependenței $U = f(\omega)$ am preluat loturi de date de la robot, ce conțin perechi de valori tensiune - viteza unghiulară și am extras dependența analitică între cele două mărimi utilizând un script Matlab, prezentat în anexe. În continuare sunt prezentate cele două caracteristici obținute pentru fiecare motor.


 Figura 35 Caracteristicile statice $U = f(\omega)$ pentru cele două motoare

Informatia generata de scriptul Matlab este utilizata pentru a crea niște tabele de look-up astfel încât putem determina tensiunea care se va aplica roților funcție de comanda de la controllerul sliding mode. În următoarea etapă utilizând loturile de date de la robot am realizat o identificare a modelelor motoarelor, individual, pentru a realiza apoi acordarea controllerului PI pentru reglarea turației. Modelul obținut prin identificare cu scriptul ident din Matlab este un model parametric de tip funcție de transfer de ordinul 1, pentru care am determinat valorile pentru parametrii K_p și K_i ai controllerului utilizând valorile pentru parametrii modelului motorului. În urma identificării, modelul motorului de curent continuu este obținut sub forma

$H(s) = \frac{K}{1 + T_p s}$ și notăm $a = 1/T_{pI}$ și $b = K/T_{pI}$ și astfel parametrii controllerului se pot obține

utilizând următoarele relații $K_i = \frac{\omega^2}{b}$ și $K_p = \frac{(2\zeta\omega - a)}{b}$. Impunând pentru criteriile de performanță

un factor de amortizare $\zeta = 0.8$, un timp de răspuns de 0.5s și o pulsație $\omega = \frac{4}{\zeta t_r}$ s-au obținut

următoarele răspunsuri în buclă inchisă,

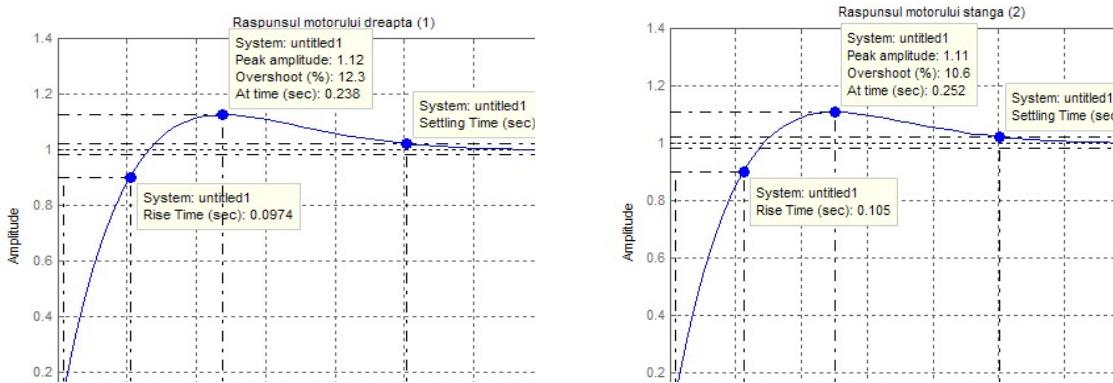


Figura 36 Răspunsul motoarelor la intrare treaptă pentru cazul 1 de acordare al controllerelor PI

În acest caz se poate observa că s-a obținut un suprareglaj peste 10% ceea ce este inacceptabil, astfel am ales, pentru a obține un suprareglaj mai mic, un compromis în a crește timpul de răspuns cu câteva zeci de ms. Astfel impunând criterii de performanță un factor de amortizare $\zeta = 1$, un timp de răspuns de 0.5s, s-a obținut un suprareglaj sub 10 % acceptabil și timpul regimului tranzitoriu a crescut cu 70-100 ms, ceea ce în cazul de față am considerat ca fiind acceptabil ca timp de răspuns. Răspunsul motoarelor în buclă închisă este redat în figura următoare,

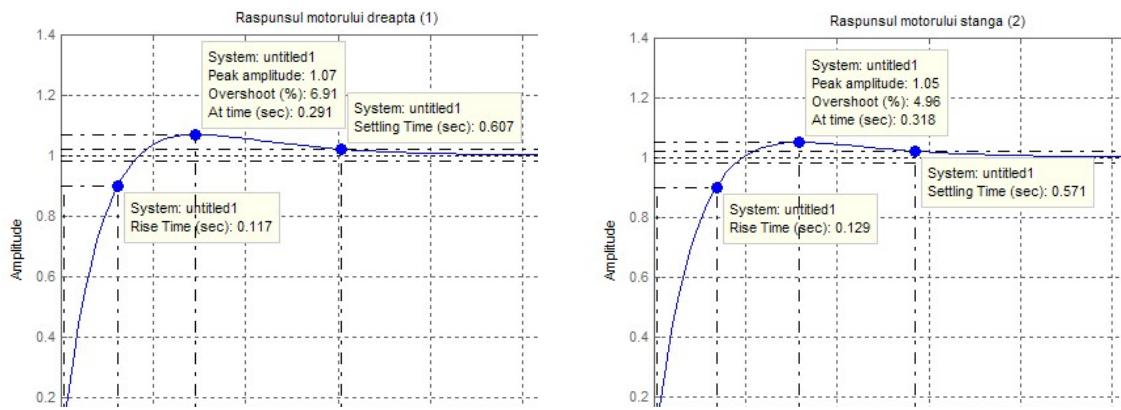


Figura 37 Răspunsul motoarelor la intrare treaptă pentru cazul 2 de acordare al controllerelor PI

Pornind de la valorile parametrilor, obținute prin programul Matlab, s-au făcut ajustări experimentale pentru atingerea acelor valori potrivite pentru obținerea unor performanțe acceptabile în operarea reală a robotului, ținând cont de variabile precum suprafață de rulare a robotului și alunecări, de faptul că motoarele nu sunt perfect identice și de faptul că transmisia mișcării de la roată la encodere (raport 1:1) poate introduce erori de calcul.

4.1.2 Sinteză buclei externe de conducere pentru poziționarea robotului în regim de trajectory tracking. Sistemul odometric

Pornind de la condițiile de control al mișcării în regimul de trajectory tracking, privind stabilizarea și convergența rapidă către traекторia de referință, bucla de control externă utilizează un controller sliding mode care oferă performanțe bune, concretizate printr-o convergență rapidă a erorilor laterale. Structura buclei de control este redată în figura 38.

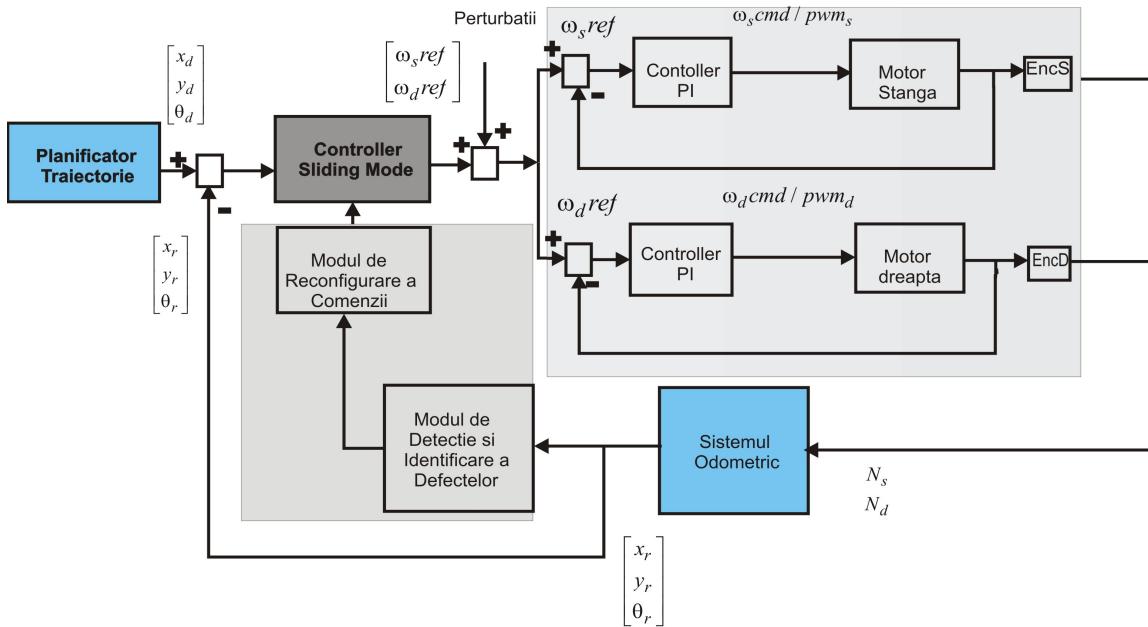
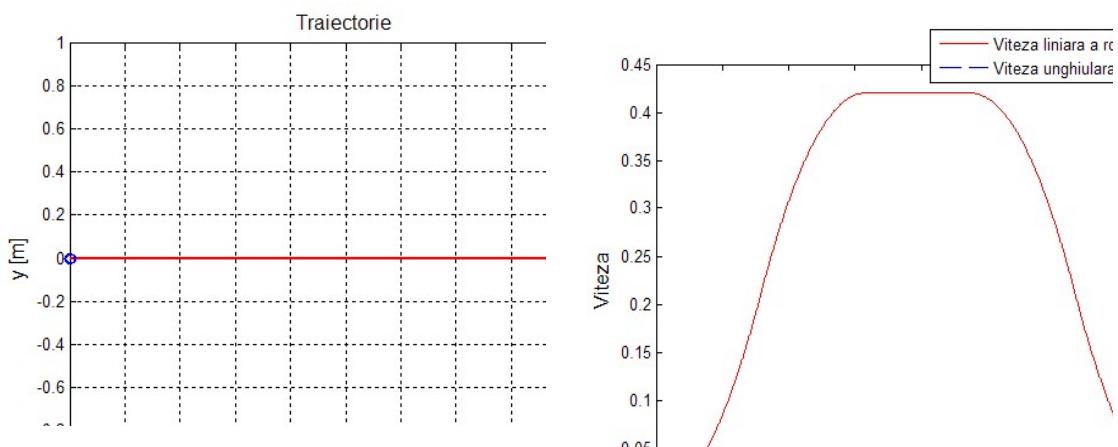
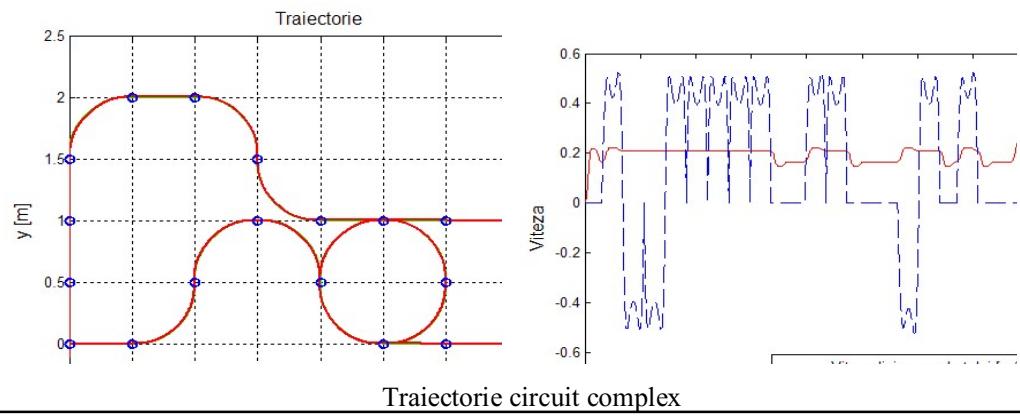
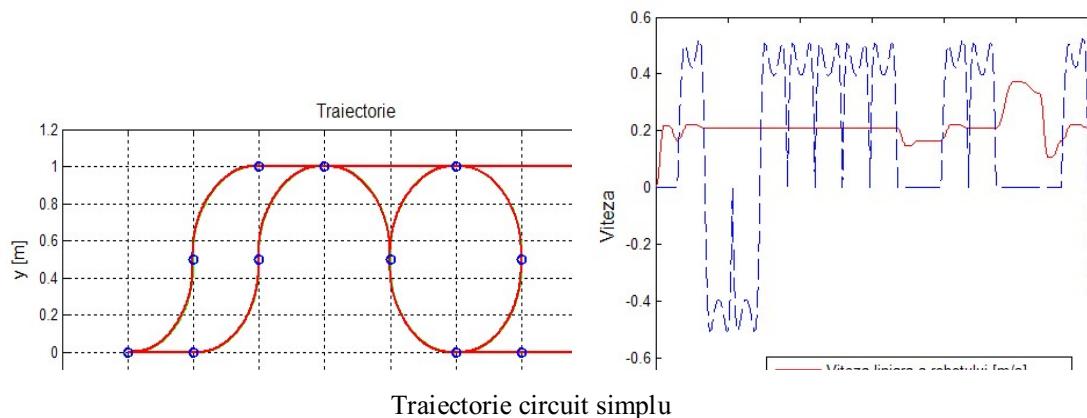
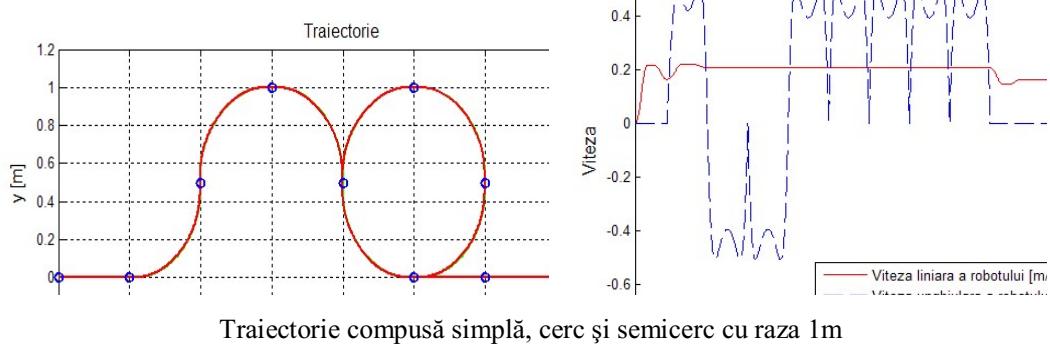
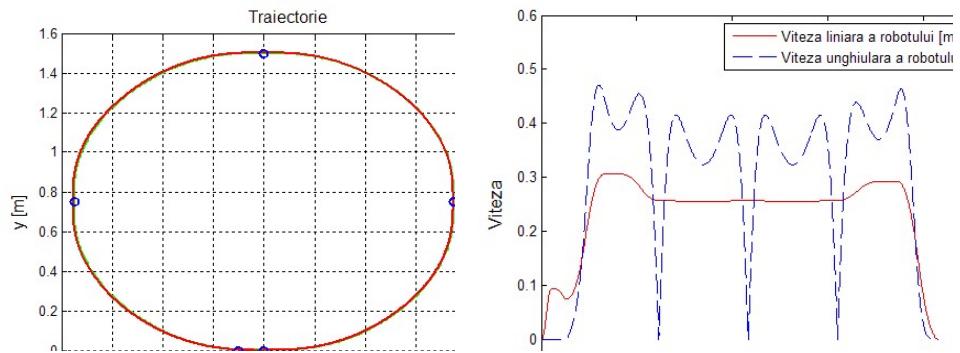


Figura 38 Structura buclei de control generale

Referința pentru controllerul sliding mode este generată de modulul de planning al traiectoriei sub forma unor referințe de viteză liniară, accelerație liniară, viteză unghiulară și accelerație unghiulară. Traiectoria generată se bazează pe utilizarea unor ecuații de gradul 5 care asigură o traiectorie lină care să confere robotului posibilitatea de a urmări traiectoria de referință cât mai bine și de a minimiza efortul de comandă. Traietoriile de test și demonstrație generate pentru robot sunt redate în figurile (Figurile 39) următoare având posibilitatea de a calcula și vizualiza și variația vitezelor liniară și unghiulară ale robotului în parcursarea traiectoriei.



Traiectorie linie dreaptă cu lungimea de 2 m



În ceea ce privește generarea traectoriilor am creat un script Matlab care pornind de la ecuațiile de mișcare ale robotului și considerând restricții privind accelerările (liniară și unghiulară) ale robotului generează traectoriile utilizând ca argumente de intrare vectori de coordonate pentru traectoria dorită. Vectorii de intrare sunt dați sub forma $[x_d \ y_d \ \theta_d]^T$ și la ieșire se obține o matrice care conține referințele pentru viteze și accelerări (care sunt utilizate mai departe ca intrare în controllerul sliding mode) ca valori de referință pentru calculul erorii de poziționare. Reacția pe bucla de control vine din modulul care implementează sistemul de odometrie. Odometria presupune determinarea poziției curente a robotului față de un punct inițial bazându-se doar pe informația de la encodere. În implementarea curentă subsistemul odometric preia informația brută (numărul de impulsuri generate de encodere) din registrele de buffer ale counterelor de 12 biți ale plăcii de achiziție și transformă aceste valori în informație de poziție. Din informația preluată din registrele counterelor se calculează numărul curent de impulsuri generate de rotația encoderelor la fiecare pas de eșantionare. Ecuațiile care descriu prelucrările sistemului odometric sunt redate în continuare,

$$\begin{cases} N_s = N_s \text{current} - N_s \text{anterior} \\ N_d = N_d \text{current} - N_d \text{anterior} \end{cases} \quad \begin{cases} \omega_s = \frac{2\pi N_s}{N_{tot} Ts} \\ \omega_d = \frac{2\pi N_d}{N_{tot} Ts} \end{cases}$$

$$v_r = \frac{r(\omega_s + \omega_d)}{2}, \quad \omega_r = \frac{r(\omega_d - \omega_s)}{L} \quad , (69)$$

$$\begin{cases} \theta_{current} = \theta_{anterior} + Ts\omega_r \\ X_{current} = X_{anterior} + Ts(v_r \cos(\theta_{current}) - d\omega_r \sin(\theta_{current})) \\ Y_{current} = Y_{anterior} + Ts(v_r \sin(\theta_{current}) + d\omega_r \cos(\theta_{current})) \end{cases}$$

unde N_{tot} este rezoluția encoderelor, Ts este perioada de eșantionare, iar celelalte notații sunt cele utilizate și la descrierea modelului cinematic al robotului mobil. Un aspect de implementare se referă la limitarea unghiului de heading, θ , la valori doar în cadrul I și II pentru a evita acumularea unei erori mari pentru care controllerul să sintetizeze o lege de comandă corespunzătoare și care să determine un comportament eronat al robotului. Pentru a minimiza erorile de odometrie s-au dezvoltat metode de calibrare și corecție adaptivă on-line ale odometriei de către Prof. Borenstein [32], care fie presupun o redundanță hardware, fie necesită execuția înaintea fiecărei misiuni a robotului a unei rutine de ajustare. Odometria este în același timp cel mai simplu dar și cel mai supus erorilor mijloc de localizare pentru roboții

mobili, întrucât poate fi foarte ușor afectată de erori. În decursul testelor s-a observat că în contextul operării robotului pe o suprafață lucioasă, deși având roți dotate cu pneuri, apare fenomenul de alunecare, la un nivel insesizabil pentru operator, însă vizibil în comanda către actuatorii robotului. Astfel, în cazul traiectoriilor care erau compuse din curbe complexe roata opusă celei de pivotare aluneca, fapt care se reflectă vizibil în reacția robotului la ieșirea din porțiunea curentă, când încerca să corecteze eroarea acumulată în calculul odometric deși în realitate el era poziționat cu o eroare acceptabilă. Un alt aspect legat de sistemul odometric se referă la testele cu robotul off-ground și cele on-ground în sensul că în cazul operării on-ground sarcina (masa robotului) influențează cuplul rezistent la axul motorului care determină preluarea unei informații diferite de către encodere față de cazul off-ground. Modul de operare al robotului a determinat în mare măsură și acordarea parametrii controllerelor PI pentru cele două motoare și implicit și acordarea parametrilor controllerului sliding mode. O altă limitare a sistemului odometric în cazul de față a reprezentat-o rezoluția scăzută a encoderelor (500PPR) care a determinat creșterea perioadei de eşantionare la 50 ms pentru buclele interne, pentru a obține informație suficientă pentru a valida calculele odometrice, utile nivelului de control pentru trajectory tracking. Pornind de la ideea proiectării unei structuri minimale care să ofere suportul pentru implementarea unei aplicații de control tolerant la defecte în timp real am neglijat pentru moment utilizarea unui sistem paralel, redundant de localizare (GPS, giroscop). Astfel, pentru a extinde în viitor structura existentă s-ar putea utiliza un accelerometru sau un giroscop pentru validarea localizării datei de odometrie, utilizarea unor elementi de ghidare la sol pentru ajustări de poziționare (ruler magnetic, magneți îngropăți sau conductori parcursi de curenti mari) sau eventual implementarea software a SLAM (Simultaneous Localization And Mapping) care ar oferi și o filtrare software a citirilor de la encodere care sunt afectate de zgomot și interferențe electromagnetice, în cazul de față utilizând doar o ecranare a conectorilor și o filtrare minimală la nivel software la efectuarea citirilor.

4.1.3 Sinteza controllerului de tip sliding mode pentru poziționarea robotului în regim de trajectory tracking

În cazul implementării de față se realizează un control combinat pentru componente longitudinale și laterale ale mișcării robotului, în contextul stabilizării robotului la traiectoria impusă, asigurând convergența erorilor laterale la 0. În aceste ipoteze robotul se poate mișca

doar pe direcția înainte pe traiectoria impusă.[68] În ceea ce privește controlul bazat pe modelul cinematic al mișcării roboților mobili în regim de trajectory tracking s-au conturat mai multe direcții. Astfel s-au dezvoltat abordări se bazează pe liniarizarea intrare-ieșire, abordări bazate pe logica fuzzy, abordări bazate pe rețele neuronale, abordări bazate pe tehnica de backstepping și nu în ultimul rând abordări bazate pe controlul sliding mode. Pornind de la o analiză din punctul de vedere al stabilității legilor de comandă, al efortului computațional și al menevrabilității, abordarea controlului sliding mode s-a dovedit a fi cea mai bună soluție.

4.1.3.1 Aspecte teoretice și de implementare ale controlului sliding mode.

Motivarea alegerii controlului sliding mode utilizând modelul cinematic al robotului

Sliding mode este o tehnică de control discontinuă aplicabilă unei game largi de sisteme. Această metodă presupune proiectarea unor funcții de comutare de variabilele de stare sau ieșire ale sistemului pentru a crea suprafete de alunecare (sliding surfaces), metoda garantând că în cazul în care traiectoria de stare a sistemului atinge suprafața ea va rămâne solidară cu suprafață, sub acțiunea funcțiilor de comutație, astfel impunându-se dinamica dorită pentru sistem.[71] Dintre avantajele utilizării controlului sliding mode se pot aminti, timpul de răspuns mic, un regim trazitoriu cu caracteristici bune și nu în ultimul rând robustețea la incertitudini și perturbații.[73] În continuare este realizată o descriere formală a controlului sliding mode și aspecte privind specificul implementării curente.

Din punctul de vedere al controlului incertitudinile de modelare pot fi clasificate în incertitudini structurate (parametrice) sau incertitudini nestructurate (dinamici nemodelate). Primul tip corespunde de fapt incertitudinilor la nivelul termenilor efectivi ai modelului sistemului și al doilea tip se referă de fapt la incertitudinea (imprecizia) ordinului sistemului. Aceste incertitudini de modelare pot determina efecte adverse asupra controlului și astfel s-au dezvoltat abordări care să țină cont de ele, în speță controlul robust și controlul adaptiv. O abordare simplă a controlului robust o reprezintă controlul sliding mode.

La baza acestei tehnici de control stă ideea că este mult mai simplu să controlezi un sistem de ordinul I (reprezentat prin EDO1) decât un sistem de ordinul n (reprezentat prin EDOn). Formalismul sliding mode introduce o reprezentare simplificatoare care realizează o înlocuire a problemelor de ordin n cu probleme echivalente de ordin I. Utilizând o astfel de

reprezentare simplificată se pot atinge performanțe bune în prezența incertitudinilor parametrice arbitrară dar cu prețul unui efort de comandă sporit. Proiectarea cu tehnica sliding mode oferă o abordare sistematică a problemei menținerii stabilității și consistenței performanțelor în contextul incertitudinilor de modelare.[69] Dacă considerăm un sistem dinamic nelinier dat de,

$$X^{(n)} = f(x) + b(x)u, \quad (70)$$

unde x este ieșirea sistemului care ne interesează, u este comanda și X este vectorul de stare,

$X = [x \ x \ ... \ x^{(n-1)}]^T$, $f(x)$ este o funcție în general nelinieră și $b(x)$ o funcție necunoscută, dar de semn cunoscut și e mărginită de funcții cunoscute de variabila x . Problema de control care se enunță este de a impune stării x să urmărească o stare variantă în timp specifică, $X_d = [x_d \ x_d \ ... \ x_d^{(n-1)}]^T$ în prezența incertitudinilor de model manifestate asupra funcțiilor $f(x)$ și $b(x)$. Pentru a asigura realizabilitatea obiectivului de urmărire utilizând o comandă finită u , trebuie ca starea dorită inițială $X_d(0) = X(0)$. Se notează $\tilde{x} = x - x_d$ eroarea de urmărire în variabila x și notăm $\tilde{X} = X - X_d = [\tilde{x} \ \tilde{x} \ ... \ \tilde{x}^{(n-1)}]^T$ vectorul erorii de urmărire. Se definește o suprafață variabilă în timp, $S(t)$, în spațiul $\mathbb{R}^{(n)}$ prin ecuația $s(\tilde{x}; t) = 0$ unde ,

$$s(\tilde{x}; t) = \left(\frac{d}{dt} + \lambda \right)^{n-1} \tilde{x}, \quad (71)$$

unde λ este o constantă strict pozitivă. Astfel pentru $n = 2$ avem $s = \tilde{x} + \lambda \dot{\tilde{x}}$ și pentru $n = 3$ $s = \tilde{x} + 2\lambda \ddot{\tilde{x}} + \lambda^2 \tilde{x}$. Înțînd cont de condiția $X_d(0) = X(0)$ problema urmăririi $X = X_d$ este echivalentă cu problema rămânerii pe suprafața $S(t)$ pentru toate valorile $t > 0$, ecuația $s = 0$ reprezentând de fapt o ecuație diferențială liniară a cărei soluție unică este $\tilde{x} = 0$ pornind din condițiile inițiale $X_d(0) = X(0)$. Astfel problema urmăririi vectorului n-dimensional X_d se reduce la a menține valoarea lui s la 0. Obiectivul simplificat de urmărire de ordin I, pentru menținerea lui s la 0 poate fi atins prin alegerea unei legi de comandă u astfel încât în afara suprafeței $S(t)$ să avem îndeplinită relația

$$\frac{1}{2} \frac{d}{dt} s^2 \leq -\eta |s|, \quad (72),$$

unde η este o constantă strict pozitivă. Relația 72 pune în evidență că pătratul distanței până la suprafață (măsurată de s^2), scade de-a lungul traекторiilor de stare ale sistemului, forțând traectoriile de stare ale sistemului să conveargă către suprafața $S(t)$. În particular, odată ajunse pe suprafață S traectoriile de stare ale sistemului rămân solidare cu suprafața, care devine un set invariant. Ecuția 72 pune în evidență condiția de alunecare (sliding condition), care implică și faptul că anumite perturbații și incertitudini dinamice pot fi tolerate menținând în același timp suprafața S atractivă (un set invariant). Condiția de alunecare este ilustrată sintetic în figura 40,

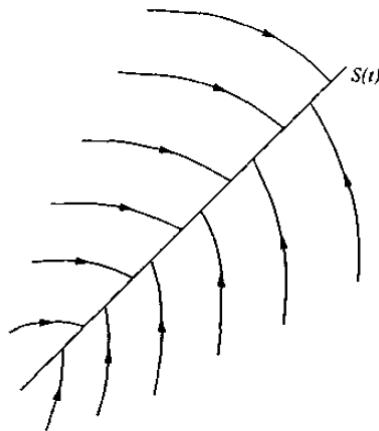


Figura 40 Convergența traectoriilor de stare către suprafață

Un alt aspect important se referă la faptul că odată solidară cu suprafața o traекторie de stare a sistemului va fi definită prin ecuația specifică suprafeței,

$$\left(\frac{d}{dt} + \lambda \right)^{n-1} \tilde{x} = 0, \quad (73).$$

Putem aprecia că suprafața de alunecare redă și localizarea spațială dar impune și dinamica sistemului. Comportamentul sistemului care satisfacă condiția de alunecare este ilustrat în figura următoare pentru cazul în care $n = 2$, fiind și cazul specific aplicației curente,

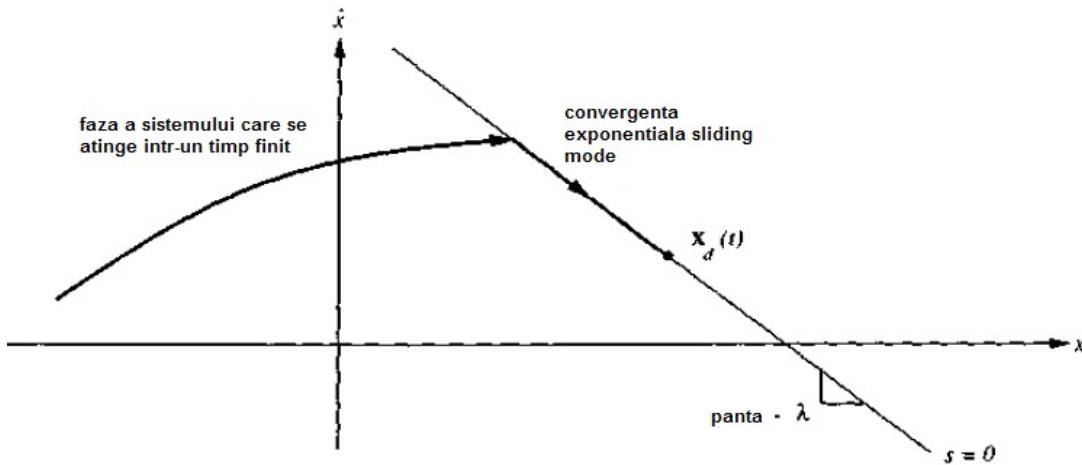


Figura 41 Reprezentarea grafică a suprafeței și condiția de alunecare

Suprafața este o dreaptă în spațiul fazelor sistemului cu panta $-\lambda$ și care conține punctele X_d . Pornind din orice condiție inițială traекторia de stare a sistemului atinge suprafața variată în timp într-un interval de timp finit și apoi alunecă pe dreapta suport a suprafeței către X_d , exponențial și cu o constantă de timp $1/\lambda$. Problema controlului sliding mode se poate formula și într-un alt mod. După cum am precizat anterior dinamica sistemului în intervalul de timp când traectoria sistemului este solidară cu suprafața este denumită alunecare. Condiția de alunecare poate fi formulată geometric ca un test pentru verificarea dacă în vecinătatea suprafeței de alunecare vectorii viteze de variație a stării sunt orientați către suprafață. În acest caz suprafața (de comutare) atrage traectoriile de stare ale sistemului când acestea sunt în vecinătatea acesteia și după ce au intersectat-o vor rămâne solidare cu aceasta. Condiția de set invariant emisă anterior poate fi văzută și ca o condiție de atraktivitate a suprafeței, în sensul că aceasta este atractivă dacă traectoriile de stare care pornesc de pe suprafață rămân în continuare solidare cu aceasta și orice traекторie de stare care începe în afara suprafeței converge cel puțin asymptotic către aceasta. În acestă accepție condițiile de alunecare se pot scrie simplificat ca $\lim_{s \rightarrow 0^+} \dot{s} < 0, \lim_{s \rightarrow 0^-} \dot{s} > 0$ și combinând cele două condiții, condiția de alunecare în vecinătatea suprafeței este $s \dot{s} < 0$. O ilustrare grafică a condiției suficiente de atraktivitate a suprafeței de alunecare este redată în figura 42.

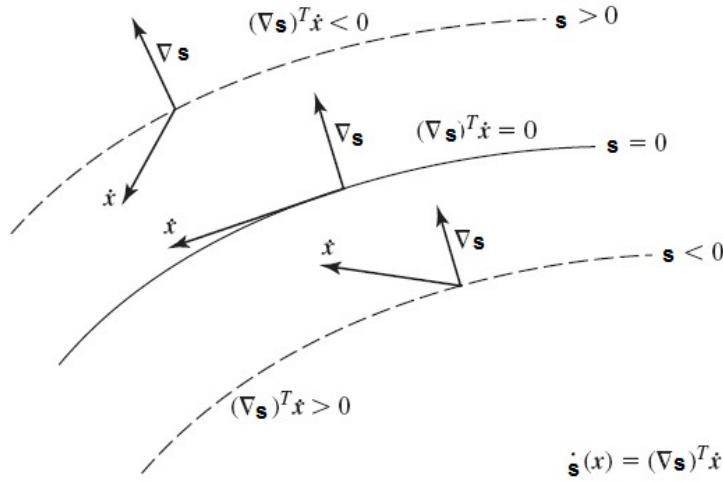


Figura 42 Condiția suficientă de atractivitate a suprafeței de alunecare

Ecuația $\dot{s}(x) = (\nabla s)^T \dot{x}$ este de fapt un produs scalar care redă derivata direcțională a lui $s(x)$ în direcția lui \dot{x} . Ideea de bază, din spatele ecuațiilor specifice ce descriu proprietățile suprafeței de alunecare, este de a alege o funcție potrivită a erorii de urmărire, s , ținând cont de ecuația 71 și de a alege o lege de comandă u astfel încât s^2 este o funcție Lyapunov a sistemului în buclă închisă în contextul incertitudinilor și perturbațiilor. Metodologia de sinteză a controllerului presupune două etape. Prima etapă implică alegerea unei legi de comandă astfel încât condiția de alunecare este satisfăcută și în plus pentru a ține cont de perturbații și incertitudini aceasta trebuie să fie discontinuă pe $S(t)$.[70] Datorită faptului că implementarea comutărilor legii de comandă nu este perfectă apare fenomenul de chattering, care nu este de dorit în practică datorită activității și efortului de comandă intens. Astfel se impune realizarea unui compromis între precizia comenzi și lărgimea de bandă, λ a controlului, prin utilizarea unei legi de comandă mai liniștită (smooth).[74] Primul pas al sintezei ia în calcul incertitudinile parametrice, în timp ce al doilea pas presupune atingerea robustății la dinamice nemodelate de înaltă frecvență. O imagine reprezentativă pentru descrierea fenomenului de chattering la comutările imperfecte ale legii de comandă discontinue sintetizată de controllerul sliding mode este redată în figura următoare. Chatteringul este un fenomen care se manifestă sub forma unor oscilații în jurul suprafeței.

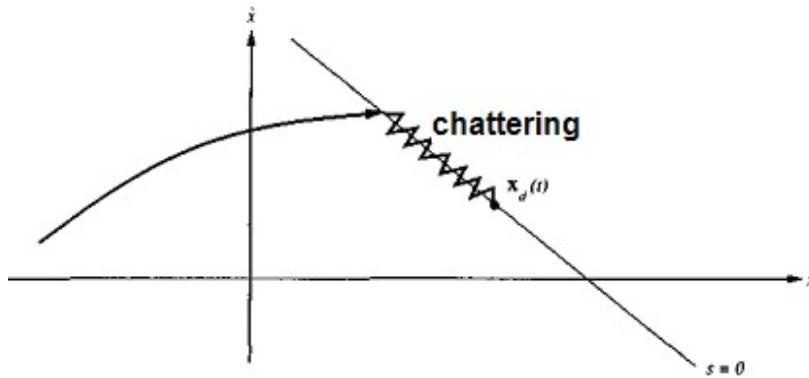


Figura 43 Fenomenul de chattering

În implementările reale chatteringul trebuie eliminat pentru a permite operarea corectă a controllerului. Acest lucru este posibil, după cum am menționat anterior, prin atenuarea (smoothing) discontinuităților legii de comandă într-un domeniu mărginit, $B(t)=\{x, |s(x;t)| \leq \Phi, \dot{\Phi} > 0\}$ din vecinătatea suprafeței de comutare. Astfel se realizează de fapt o interpolare a legii de comandă într-un domeniu mărginit care poate determina alegerea parametrului λ și limita de mărginire a domeniului definit, Φ .[72] Se poate considera că atenuarea legii de comandă în interiorul domeniului B , se realizează prin atribuirea unui filtru trece-jos dinamică locale a variabilei s , eliminând astfel chatteringul. În continuare este descrisă procedura de sinteză a controllerului sliding mode pentru controlul robotului în regim de trajectory tracking. În cadrul buclei generale de control al poziționării robotului pornind de la modelul cinematic și utilizând informația de stare de al sistemul odometric, putem obține eroarea de trajectory tracking, formată din trei componente, eroarea longitudinală, eroarea laterală și eroarea de orientare.[75] O ilustrare a modului de operare în regim trajectory tracking este descrisă în figura următoare.

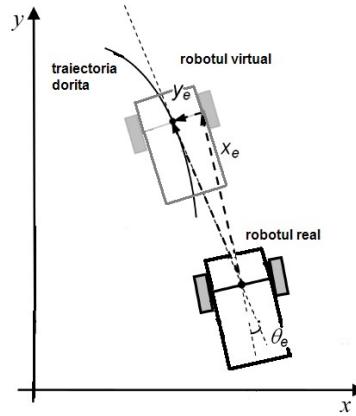


Figura 44 Operarea robotului mobil în regim de trajectory tracking

Robotul este descris de vectorul $[x_r \ y_r \ \theta_r]^T$ și definim vectorul eroare ca fiind $[x_e \ y_e \ \theta_e]^T = [x_r \ y_r \ \theta_r]^T - [x_d \ y_d \ \theta_d]^T$, $[x_d \ y_d \ \theta_d]^T$ fiind vectorul poziției dorite, a robotului virtual.

$$\begin{cases} \dot{x}_r = v_r \cos\theta \\ \dot{y}_r = v_r \sin\theta \\ \dot{\theta}_r = \omega_r \end{cases} \quad \begin{cases} \dot{x}_d = v_d \cos\theta_d \\ \dot{y}_d = v_d \sin\theta_d \\ \dot{\theta}_d = \omega_d \end{cases}, \quad \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix}, \quad (74)$$

Obiectivul aplicației este de a projecța un controller stabil și robust pentru controlul poziționării robotului, care să genereze un vector de comandă $[v_c \ \omega_c]$, în prezența perturbațiilor. Valorile corespunzătoare pentru derivatele erorii se obțin utilizând relațiile următoare,

$$\begin{cases} \dot{x}_e = -v_d + v_r \cos\theta_e + y_e \omega_d \\ \dot{y}_e = v_r \sin\theta_e - x_e \omega_d \\ \dot{\theta}_e = \omega_r - \omega_d \end{cases}, \quad (75)$$

unde valorile dorite pentru viteza liniară și unghiulară, v_d și ω_d sunt obținute din matricea de referințe care redă traiectoria generată pe baza ecuațiilor de gradul 5. Am stabilit ca obiective principale pentru controlul în regimul trajectory tracking stabilizarea la traiectoria impusă ca referință și asigurarea convergenței erorilor laterale la 0. În primă fază se realizează proiectarea suprafețelor de comutare pentru controllerul sliding mode. Astfel prima suprafață asigură convergența erorii longitudinale și este redată prin ecuația 76,

$$s_1 = \dot{x}_e + k_1 x_e, \quad (76)$$

iar a doua suprafață de comutare care asigură cuplarea internă a variabilelor eroare laterală, y_e și eroare de orientare θ_e pentru a asigura convergența ambelor variabile este redată în continuare de ecuația 77,

$$s_2 = \dot{y}_e + k_2 y_e + k_0 \operatorname{sgn}(y_e) \theta_e, \quad (77),$$

unde parametrii k_0, k_1, k_2 sunt ponderile asociate fiecărei componente a erorii în calculul suprafețelor de comutare. O formă generală pentru determinarea practică a legii de comandă este dată de ecuația următoare,

$$\dot{s} = -Qs - P \operatorname{sgn}(s), \quad (78)$$

în care Q și P sunt valori constante pozitive. Un aspect important se referă la adunarea termenului proporțional $-Qs$ care determină creșterea vitezei de convergență a traiectoriei de stare către suprafața de comutare pentru valori mari ale lui s .

Derivând ecuațiile 76 și 77 în timp obținem ,

$$\begin{aligned} \dot{s}_1 &= \ddot{x}_e + k_1 \dot{x}_e \\ \dot{s}_2 &= \ddot{y}_e + k_2 \dot{y}_e + k_0 \operatorname{sgn}(y_e) \theta_e \end{aligned}, \quad (79)$$

și înlocuind cu expresiile echivalente din definirea erorii de poziționare și derivatei erorii de poziționare și ținând cont de faptul că $\dot{\theta}_e = \dot{\theta}_r - \dot{\theta}_d = \omega_r - \omega_d$ după manipulări simple se obțin expresiile pentru mărimele de comandă, v_c și ω_c ,

$$v_c = \frac{-Q_1 s_1 - P_1 \operatorname{sgn}(s_1) - k_1 \dot{x}_e - \omega_d \dot{y}_e - \omega_d \dot{\theta}_e + v_r \theta_e \sin \theta_e + \dot{\theta}_d}{\cos \theta_e}$$

$$\omega_c = \frac{-Q_2 s_2 - P_2 \operatorname{sgn}(s_2) - k_2 \dot{y}_e - v_r \sin \theta_e + \omega_d \dot{x}_e + \omega_d \dot{\theta}_e}{v_r \cos \theta_e + k_0 \operatorname{sgn}(y_e)} + \omega_d, \quad (80)$$

Termenii Q_1, P_1 și Q_2, P_2 au rolul de a asigura viteza de convergență a vectorului de stare către suprafața de comutare, însă pentru valori prea mari se obțin oscilații mari (chattering) pentru mișcarea pe axa X (Q_1, P_1), respectiv pentru Y și θ (Q_2, P_2).[72]

Pentru verificarea stabilității se definește o funcție candidat Lyapunov $V = \frac{1}{2} s^T s$ a cărei derivată este dată de

$$\dot{V} = s_1 \dot{s}_1 + s_2 \dot{s}_2 = s_1 (-Q_1 s_1 - P_1 \operatorname{sgn}(s_1)) + s_2 (-Q_2 s_2 - P_2 \operatorname{sgn}(s_2)) = -s^T Q s - P_1 |s_1| - P_2 |s_2|, \quad (81).$$

Pentru a asigura stabilitatea trebuie ca $V \dot{V} < 0$ și pentru ca \dot{V} să fie pozitiv semidefinită este suficientă alegerea Q_i, P_i pozitiv semi-definite, $Q_i \geq 0, P_i \geq 0$.[69]

4.1.3.2 Studiul performanțelor robotului în buclă închisă pentru trajectory tracking

În continuare este realizată o analiză a rezultatelor în buclă închisă obținute pentru controllerul sliding mode proiectat. Se analizează evoluția erorilor, comanda și comportamentul sistemului în buclă închisă pentru o trajectorie circuit simplu. Astfel trajectoria și profilul vitezei liniare și vitezei unghiulare sunt ilustrate în continuare.

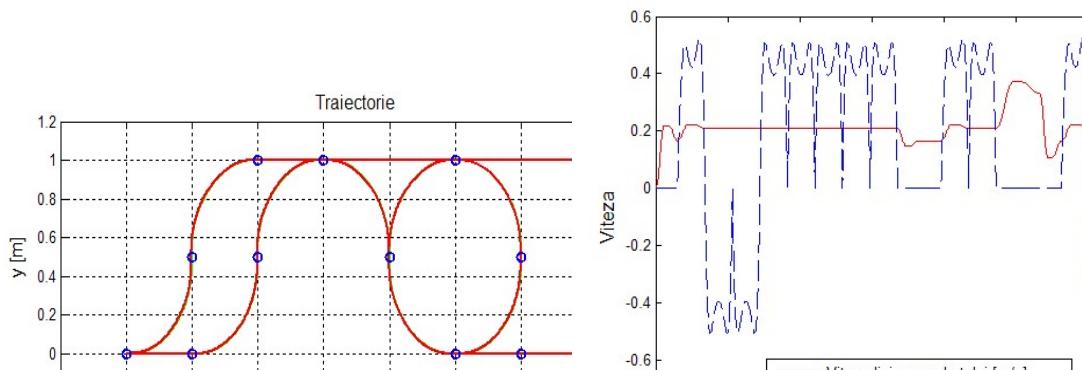


Figura 45 Trajectoria utilizată pentru analiza comportamentului robotului în buclă

Primul aspect de analiză se referă la trajectoria de referință generată de planificatorul de traectorie și trajectoria realizată de robot, care a fost generată din loturi de date preluate de la robot. În figura următoare putem observa o eroare pe anumite segmente ale traectoriei, o eroare cumulativă care a influențat progresiv mișcarea robotului până la punctul de final.

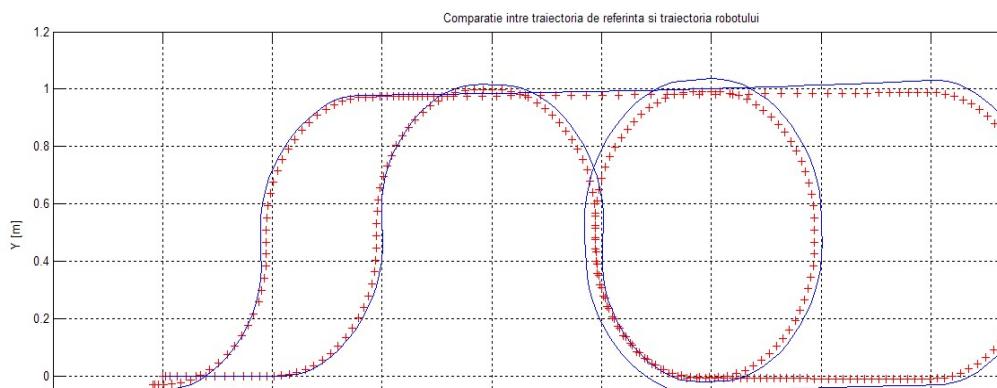


Figura 46 Comparație între trajectoria ideală (referință) și trajectoria reală

Observațiile care au fost făcute anterior sunt susținute de analiza erorii pe cele trei componente, longitudinală, laterală și de orientare. În figurile următoare avem cele 3 erori,

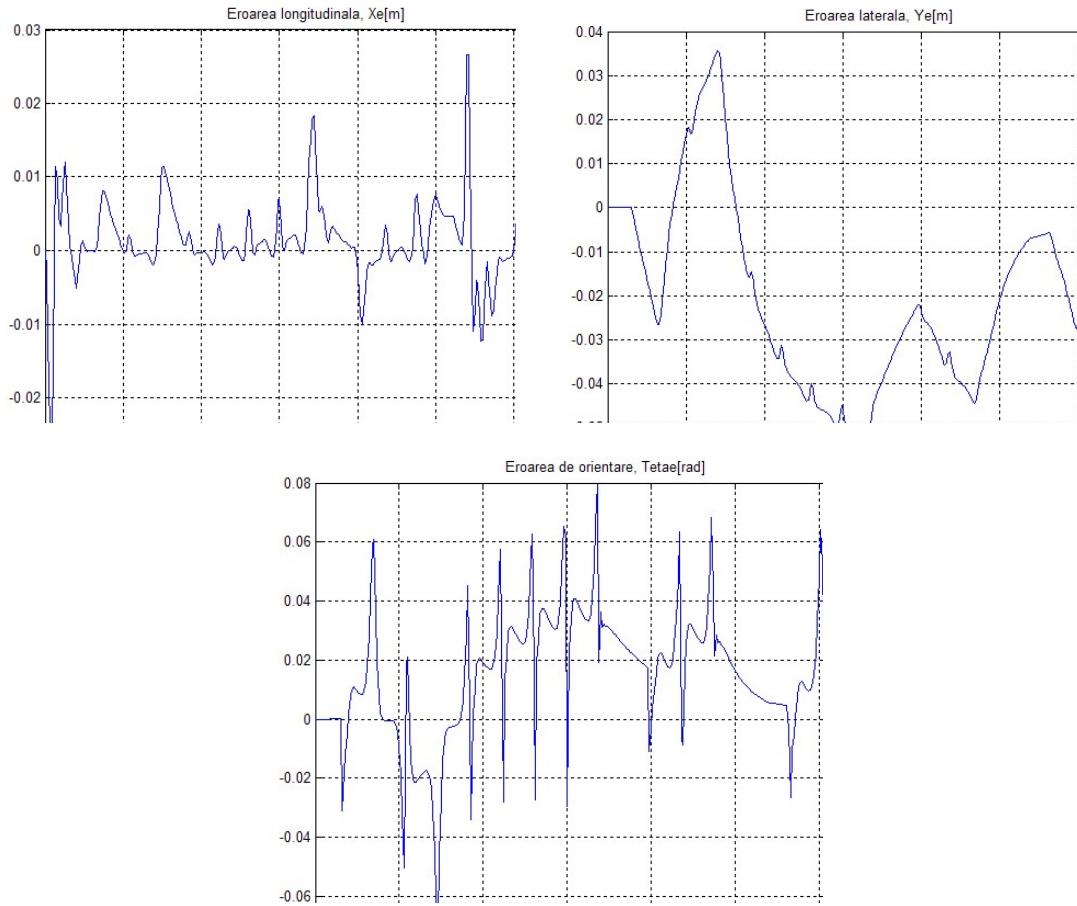


Figura 47 Cele trei componente ale erorii de poziționare

Se poate observa că eroarea se acumulează și la momentul $t = 34s$ are o valoare mare care este vizibilă în partea a doua a cercului din traекторie când la intrarea pe linia dreaptă ce urmează, controllerul minimizează eroarea, însă aceasta apare din nou la intrarea pe linia dreaptă de revenire la punctul de start. Putem observa că erorile majore care apar (în mare parte generate de sistemul odometric) se manifestă la comutarea între componente ale traectoriei, de la linie la cerc și invers. Aceste probleme pot fi rezolvate prin un tuning mai fin al controllerului sliding mode în ceea ce privește parametrii Q și P care redau încotocmai viteza de convergență către traectoria de referință. La acest moment nu a fost dezvoltată o metodă analitică universal aplicabilă de alegere a parametrilor controllerului și de aceea aceștia fiind aleși empiric pot comporta modificări permanente până la atingerea unui nivel satisfăcător al performanțelor răspunsului. În continuare se face analiza comenzi sintetizate de controllerul sliding mode și

comparația cu valorile măsurate de la robot. Astfel în figura următoare avem figurate variațiile vitezei liniare a robotului și comanda dată de controller,

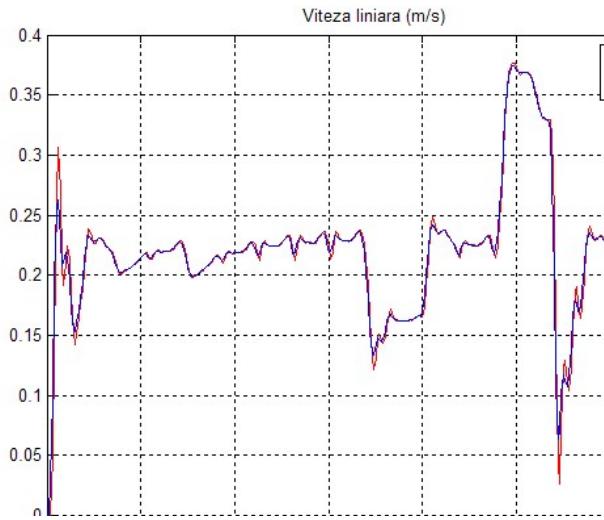


Figura 48 Viteza liniară a robotului și comanda, v_c

Se poate observa că robotul urmărește referința de viteză impusă cu o eroare medie minimă, însă valorile maxime ale erorii apar în momentele comutării de pe o componentă a traiectoriei la alta, întrucât are loc o schimbare bruscă de referință, lucru care este mai edificator în cazul analizei vitezei unghiulare a robotului și referinței de viteză unghiulară (suprareglaj semnificativ), redată în figura 49.

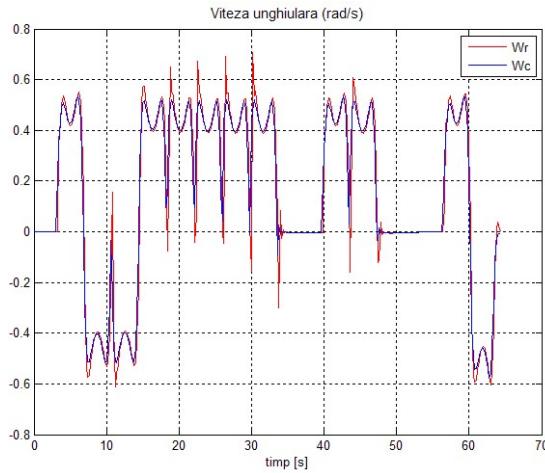


Figura 49 Viteza unghiulară a robotului și comanda ω_c

În ultimă instanță comanda sintetizată de controllerul sliding mode este transformată în referințe de viteză unghiulară pentru cele două motoare (printr-o transformare cinematică

inversă) și se poate observa cum controllerele PI sintetizate asigură că valoarea comenzi emise de controlerul sliding mode se reflectă în răspunsul motoarelor, ceea ce redă un timp de răspuns acceptabil și utilizarea unor parametri de acordare buni pentru cele două controllere PI. În figura 50 putem observa referințele și răspunsurile celor doi actuatori ai robotului,

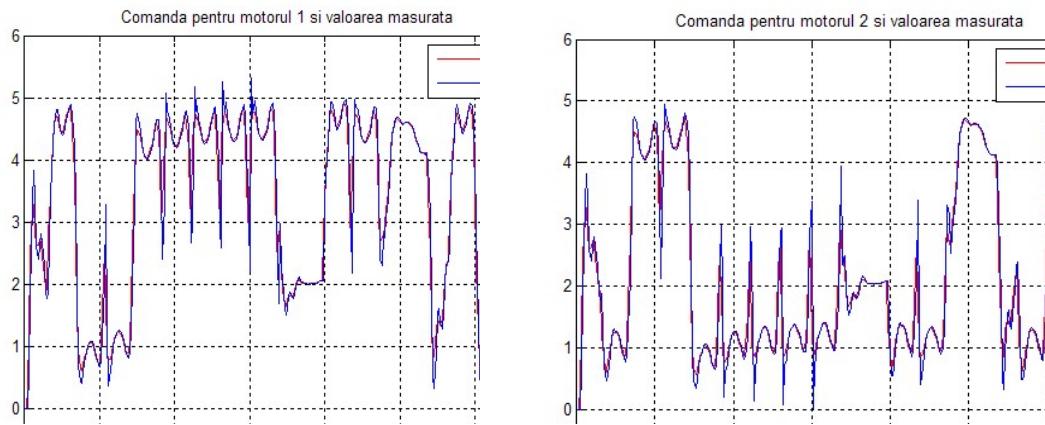


Figura 50 Analiza răspunsului actuatorilor

Se poate observa că turația (ieșirea) motoarelor urmărește atent referința, cu un suprareglaj minim (6-7%) și un timp de răspuns bun, acest lucru fiind foarte important în proiectarea sistemului pentru că a determinat alegerea perioadei de eşantionare și mai ales că pentru a asigura un control eficient pentru poziționare, nivelul inferior al reglării turației actuatorilor trebuia acordat foarte bine. Analiza făcută anterior la nivelul comportamentului robotului în buclă închisă va fi utilă pentru a realiza o comparație cu cazul în care vor apărea defectele în sistem. Se pot trage niște concluzii utile în ceea ce privește utilizarea controlului sliding mode în cazul controlului mișcării robotului în regim de trajectory tracking. Astfel scopul controllerului sliding mode a fost de a sintetiza o lege de comandă care să țină cont de incertitudinile parametrice și perturbațiilor, precum și de a realiza o cuantificare a compromisurilor modelare – performanță. Compromisul dintre performanță în urmărire și incertitudinea parametrică corespunde în practică de fapt cu înlocuirea legii de comandă discontinuă printr-o aproximare atenuată, lină (smooth).

În continuare atenția se concentrează pe analiza și sinteza sistemului de detecție, identificare și reconfigurare a controlului pentru atingerea obiectivului de control tolerant la defecte.

- 4.2 Nivelul task-ului de monitorizare, diagnoză și toleranță la defecte. Analiza și sinteza sistemului de diagnoză și toleranță a defectelor

Nivelul superior al aplicației dezvoltate este nivelul concret al implementării diagnozei și toleranței la defecte. Astfel în structura aplicației peste nivelul de control sliding mode în regim de trajectory tracking este implementat nivelul de monitorizare și diagnoză multi-model, pentru detecția, identificarea defectelor și reconfigurarea controlului pentru a asigura continuitate în operarea robotului. Făcând raportare la aplicația dezvoltată, pornind de la structura de Grafcet paralel, taskul de monitorizare și diagnoză rulează în paralel cu taskul de control ambele bazându-se pe informația primită de la sistemul odometric. În continuare este prezentat cadrul general de interes pentru analiza tipurilor de defecte specifice roboților mobili și apoi se realizează o prezentare și o analiză a rezultatelor soluției de diagnoză multi-model bazată pe filtre Kalman pentru defecte manifestate prin variația parametrilor fizici ai procesului.

4.2.1 Analiza tipurilor de defecte ale roboților mobili

Defectele care pot apărea în structura unui robot mobil pot fi clasificate în defecte ale senzorilor, defecte în actuatori și defecte marcate prin variația parametrilor fizici ai robotului. În continuare se va face o prezentare formală și se vor ilustra exemple pentru primele două categorii de defecte urmând ca cea de-a treia categorie să fie prezentată efectiv în contextul aplicației dezvoltate. Considerăm o buclă generică pentru controlul mișcării unui robot, în care considerăm că vectorul de intrare și vectorul de ieșire sunt m-dimensionali și modelul robotului dat sub formă funcție de transfer, $G(s)$, u_a este vectorul de ieșire a actuatorilor, y_m vectorul de ieșire al senzorilor, u comanda sintetizată de controller și y ieșirea, care poate să fie poziția robotului în spațiul cartezian.

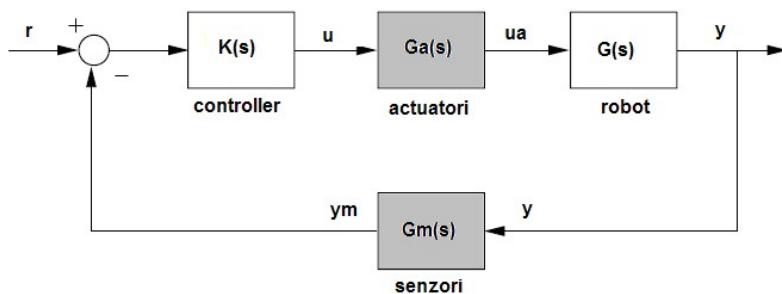


Figura 51 Sistem de control generic pentru robotul mobil

Se presupune că dinamicile senzorilor și actuatorilor sunt neglijabile. Astfel defectele în senzori și actuatori pot fi reprezentate utilizând următoarele modele,[1]

$$\begin{aligned} u_a &= E_a u + f_a \\ y_m &= E_m y + f_m, \end{aligned} \quad (82)$$

unde E_a , E_m sunt matrici mxm dimensiunionale care reprezintă defectele și au următoarea structură diagonală,

$$E_a = \begin{bmatrix} \varepsilon_1^a & 0 & \dots & 0 \\ 0 & \varepsilon_2^a & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \varepsilon_m^a \end{bmatrix}, \quad E_m = \begin{bmatrix} \varepsilon_1^m & 0 & \dots & 0 \\ 0 & \varepsilon_2^m & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \varepsilon_m^m \end{bmatrix}, \quad (83)$$

unde $0 \leq \varepsilon_i^a, \varepsilon_i^m \leq 1 (i=1,2,\dots,m)$. Vectorii $f_a = [f_1^a, \dots, f_i^a, \dots, f_m^a]^T$ și $f_m = [f_1^m, \dots, f_i^m, \dots, f_m^m]^T$ reprezintă componente constante ale ieșirilor senzorilor și actuatorilor când apare un defect. În acest context utilizarea acestui model determină următoarele scenarii tipice de manifestare a defectelor (se presupune că defectul apare în senzorul sau actuatorul i al sistemului) :

1. Indisponibilitatea (ieșirea din uz a) senzorului : $\varepsilon_i^m = 0, f_i^m = 0$;
2. Indisponibilitatea (ieșirea din uz a) actuatorului : $\varepsilon_i^a = 0, f_i^a = 0$;
3. Funcționarea parțială a senzorului : $0 \leq \varepsilon_i^m \leq 1$;
4. Funcționarea parțială a actuatorului : $0 \leq \varepsilon_i^a \leq 1$;
5. Stabilirea ieșirii senzorului pe o anumită valoare (înghețarea ieșirii senzorului) $\varepsilon_i^m = 0, f_i^m$ = valoarea constantă a ieșirii senzorului ;
6. Stabilirea ieșirii actuatorului pe o anumită valoare (înghețarea ieșirii actuatorului) $\varepsilon_i^a = 0, f_i^a$ = valoarea constantă a ieșirii actuatorului (posibil valoarea maximă, saturatie);

Pornind de la structura anterioară defectele în senzori și actuatori se pot reprezenta printr-o strcuture alternativă ilustrată în figura 52.

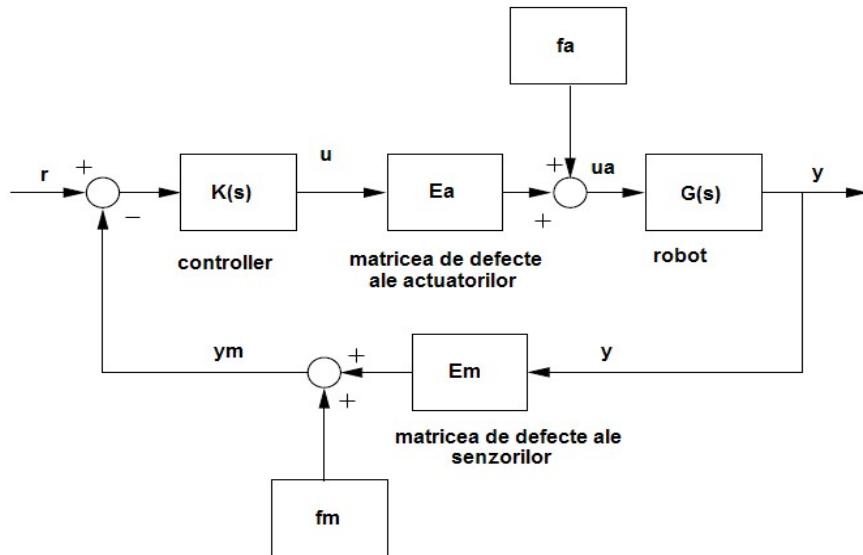


Figura 52 Reprezentarea defectelor în senzori și actuatori

Pentru a ilustra mai bine specificul defectelor ce pot apărea în cazul roboților mobili se propune pentru exemplificare un caz în care rețeaua de senzori a robotului este alcătuită din 2 encoder și un giroscop. Defectele care pot apărea în cazul propus se pot clasifica în defecte hard (consecințe sesizabile asupra operării robotului) și defecte soft (care determină o alterare nu foarte pronunțată a comportamentului robotului). În acest caz în prima categorie intră defectele care se manifestă prin faptul că valoarea oferită de senzor rămâne constantă (înghetață pe o anumită valoare) și măsurările disponibile de la acesta sunt ignorate. A doua categorie de defecte se referă la situația în care performanțele senzorului se degradează, însă acesta este încă utilizabil, dând informații în medie valide dar având o pondere mai mică în generarea informației de stare a robotului. Concret o ipoteză de operare cu defect a robotului se manifestă prin degradarea giroscopului care măsoară unghiul de orientare al robotului în sensul modificării profilului de zgomot. O altă ipoteză se referă la ieșirea din uz a giroscopului și astfel poziționarea va rămâne accesibilă doar de la sistemul odometric. O ultimă ipoteză se referă la pierderea completă a informației de la un encoder, care putea fi determinată fie de căderea tensiunii de alimentare pe encoder, fie de un defect în circuitul optic de recepție al impulsurilor. În cazul defectelor în actuatori, de exemplu motoarele de curent continuu, defectele de tip hard determină încheierea operării robotului, deși după detectia unui astfel de defect se impun măsuri de siguranță la oprirea de urgență. Defectele în motoarele de curent continuu, care nu determină ieșirea din uz, pot fi determinate de trei contexte de apariție a defectelor. Astfel prima anomalie se referă la reducerea suprafeței de

contact a perilor colectoare (care are loc datorită depunerilor de cărbune), care se manifestă prin variația parametrilor R_a până la +10% și Ψ până la -2%, deviațiile scăzând pentru R_a și crescând pentru Ψ datorită efectului de șlefuire. Al doilea tip de anomalie este determinată de insuficiența aerului de răcire, astfel o reducere a curentului de aer, prin încapsularea motorului, determină creșterea temperaturii în motor care determină o creștere a parametrului R_a până la 6% și o scădere a fluxului Ψ , până la -1.5%. O ultimă anomalie se referă la pornirea motorului, care până la intrarea în regim nominal de funcționare determină o variație a parametrilor de până la -10% pentru R_a , +5% pentru Ψ și +45% pentru coeficientul de frecare, F_a . Aceste tipuri de anomalii, caracterizate prin variațiile parametrilor actuatorilor se reflectă în timpul de răspuns și în ieșirea sistemului. În continuare este tratată problematica defectelor care se manifestă prin variația parametrilor sistemului, care determină modificări în dinamica acestuia și descrierea abordării bazate pe filtrul Kalman extins pentru diagnoza defectelor de acest tip pentru robotul mobil.

4.2.2 Implementarea sistemului de diagnoză și toleranță a defectelor bazat pe filtrul Kalman extins

În cazul implementării curente am utilizat o metodă bazată pe model pentru detecția și identificarea defectelor robotului mobil care are la bază utilizarea filtrului Kalman, pentru estimarea stării sistemului.[77] Aparatul matematic necesar implementării unui filtru Kalman utilizează elemente de teoria probabilităților, variabile aleatoare, medii și varianțe, distribuții normale și Gauss, elemente de estimare stochastică și problema proiectării observerelor.[78] De interes sunt aspectele legate de problema proiectării unui observer. Astfel proiectarea unui observer se bazează pe determinarea (estimarea) stărilor unui sistem având acces doar la ieșirile sistemului. Au fost dezvoltate mai multe abordări în această direcție care se bazează pe reprezentarea de stare a sistemului, care redă transformările dinamicei sistemului. Generic putem reprezenta modelul sistemului ca o ecuație stochastică cu diferențe,

$$x_k = Ax_{k-1} - Bu_k + w_{k-1}, \quad (84).$$

În plus se poate formula și un model al măsurătorilor care pune de fapt în evidență relația între stările sistemului și măsurători,

$$z_k = Hx_k + v_k, \quad (85).$$

Termenii w_k și v_k sunt variabile aleatoare reprezentând zgomotul de proces, respectiv zgomotul de măsură. Vectorul z_k pune în evidență faptul că măsurătorile nu trebuie să fie neapărat stări, ci pot fi și combinații liniare de stări. În ceea ce privește zgomotul de măsură există mai multe surse posibile, de exemplu fiecare tip de senzor are limitări fundamentale legate de mediul fizic de operare și în momentul în care aceste limitări sunt depășite măsurătorile sunt degradate. O parte de zgomot este adăugată semnalului util prin circuitorul de conectare a senzorului.[81] În continuare se descrie formal filtrul Kalman motivând alegerea utilizării acestuia pentru problema de față.

4.2.2.1 Descrierea formală a filtrului Kalman extins și motivarea utilizării în contextul problemei abordate

Filtrul Kalman este în esență un set de ecuații ce implementează un estimator de tipul predictor-corector optimal, în sensul că minimizează covarianța erorii de estimare când anumite condiții sunt satisfăcute. Filtrul este utilizat atât pentru estimarea stării cât și pentru estimare parametrică, utilizând o procedură recursivă de estimare bazată pe seturi secvențiale de date de măsură. Se impune cunoașterea apriorică a stării (exprimată prin matricea de covarianță) care este modificată la fiecare pas de eșantionare, utilizând informația anterioră și informația nouă pentru estimările ulterioare.[79] Încă din momentul introducerii, filtrul Kalman, a dat dovada unor avantaje notabile în implementarea digitală care i-au extins gama de aplicabilitate, fiind în același timp și simplu de implementat și robust.

În mod general filtrul Kalman propune rezolvarea problemei estimării vectorului de stare $x \in \Re^n$ a unui sistem în timp discret guvernă de o ecuație cu diferențe liniară stochastică, redat în ecuația 84. Ca răspuns pentru situațiile în care procesul care trebuie estimat și / sau relația măsurători-proces este neliniară a fost dezvoltat filtrul Kalman extins care realizează o liniarizare în jurul mediei și covarianței curente.[80] Presupunem că sistemul este descris de ecuația stochastică neliniară cu diferențe,

$$\begin{aligned}x_k &= f(x_{k-1}, u_k, w_{k-1}) \\z_k &= h(x_k, v_k)\end{aligned}, \quad (86)$$

În cazul de față funcția neliniară f leagă valoarea stării la pasul anterior k-1 cu valoarea stării la momentul curent k.

Funcția neliniară h leagă vectorul de stare x_k de vectorul măsurătorilor z_k . În practică nu se pot cunoaște valorile individuale ale zgomotului de proces și de măsură, w_k și v_k și se pot realiza astfel niște estimări ale vectorului de stare și de măsură, neglijând aceste valori,

$$\begin{aligned}\tilde{x}_k &= \hat{f}(x_{k-1}, u_k, 0) \\ \tilde{z}_k &= h(\tilde{x}_k, 0)\end{aligned}, \quad (87)$$

unde \hat{x}_k este o estimare a posteriori a stării (de la un moment de timp k anterior). Un aspect important, care constituie și un dezavantaj al EKF, constă în faptul că distribuțiile variabilelor aleatoare nu mai sunt normale, după aplicarea transformărilor neliniare. EKF este astfel un estimator de stare care aproximează optimalitatea legii lui Bayes prin liniarizare. Pentru a putea estima modelul unui sistem cu o reprezentare neliniară se pornește de la ecuațiile care realizează o liniarizare în jurul unei estimări,

$$\begin{aligned}\tilde{x}_k &\approx \hat{x}_k + \Phi(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \\ \tilde{z}_k &\approx z_k + H(x_k - \tilde{x}_k) + Vv_k\end{aligned}, \quad (88)$$

unde,

- x_k și z_k sunt valorile curente ale vectorului de stare și vectorului măsurătorilor,
- \tilde{x}_k, \tilde{z}_k sunt aproximările vectorului de stare și vectorului măsurătorilor,
- \hat{x}_k este o estimare a posteriori la pasul k,
- w_k și v_k sunt variabile aleatoare reprezentând zgomotul de proces, respectiv zgomotul de măsură,
- Φ este matricea Jacobian a derivatelor lui f în raport cu x ,

$$\Phi_{(i,j)}[k] = \frac{\partial f_i}{\partial x_j}(\hat{x}_{k-1}, u_k, 0), \quad (89),$$

- W este matricea Jacobian a derivateelor lui f în raport cu w ,

$$W_{(i,j)}[k] = \frac{\partial f_i}{\partial w_j}(\hat{x}_{k-1}, u_k, 0), \quad (90),$$

- H este matricea Jacobian a derivateelor lui h în raport cu x ,

$$H_{(i,j)}[k] = \frac{\partial h_i}{\partial x_j}(\tilde{x}_k, 0), \quad (91),$$

- V este matricea Jacobian a derivateelor lui h în raport cu v ,

$$V_{(i,j)}[k] = \frac{\partial h_i}{\partial v_j}(\tilde{x}_k, 0), \quad (92).$$

Se definește în continuare eroarea de predicție ca fiind,

$$\tilde{e}_{x_k} = \tilde{x}_k - \hat{x}_k, \quad (93),$$

și reziduul de măsură

$$\tilde{e}_{z_k} = z_k - \hat{z}_k, \quad (94).$$

În realitate nu putem avea acces la variabila x_k din ecuația 93, fiind cantitatea care se dorește a fi estimată. În schimb mărimea z_k este accesibilă în ecuația 94 fiind măsurătoarea care este utilizată la estimarea lui x_k . Utilizând ecuațiile anterioare se poate scrie modelul unui proces eroare, sub forma,

$$\begin{aligned} \tilde{e}_{x_k} &\approx \Phi(\hat{x}_{k-1} - \hat{x}_{k-1}) + \varepsilon_k \\ \tilde{e}_{z_k} &\approx H \tilde{e}_{x_k} + \eta_k \end{aligned}, \quad (95),$$

unde ε_k și η_k sunt variabile independente aleatoare având media zero și matricile de covarianță WQW^T și VRV^T în care Q este matricea de covarianță a zgromotului de proces și R este matricea de covarianță a zgromotului de măsură și în plus $p(w) \sim N(0, Q)$, $p(v) \sim N(0, R)$.

Prima ecuație 95 este o eroarea de predicție care poate fi notată \hat{e}_k și poate fi utilizată împreună cu ecuația 93 pentru a obține o estimare a posteriori pentru sistem,

$$\hat{x}_k = \hat{x}_k + \hat{e}_k, \quad (96).$$

Se fac aproximările următoare,

$$\begin{aligned} p(\tilde{e}_k) &\sim N(0, E[\tilde{e}_{x_k}, \tilde{e}_{x_k}^T]) \\ p(\varepsilon_k) &\sim N(0, WQW^T) \\ p(\eta_k) &\sim N(0, VRV^T) \end{aligned}, \quad (97).$$

Pornind de la aproximările făcute și considerând $\hat{e}_k = 0$, ecuația filtrului Kalman pentru estimarea lui \hat{e}_k este,

$$\hat{e}_k = K_k \tilde{e}_{z_k}, \quad (98),$$

efectuând apoi manipularile necesare putem obține ecuația care se poate utiliza pentru update-ul măsurătorilor în EKF,

$$\hat{x}_k = \hat{x}_k + \tilde{K}_k (\tilde{z}_k - \tilde{z}_k), \quad (99).$$

În continuare dacă se notează $\hat{x}_k = \tilde{x}_k$ ecuațiile care descriu EKF pentru cele două etape de predicție (update-ul timpului) și corecție (update-ul măsurătorilor) sunt ilustrate în figura următoare care descrie și operarea EKF,

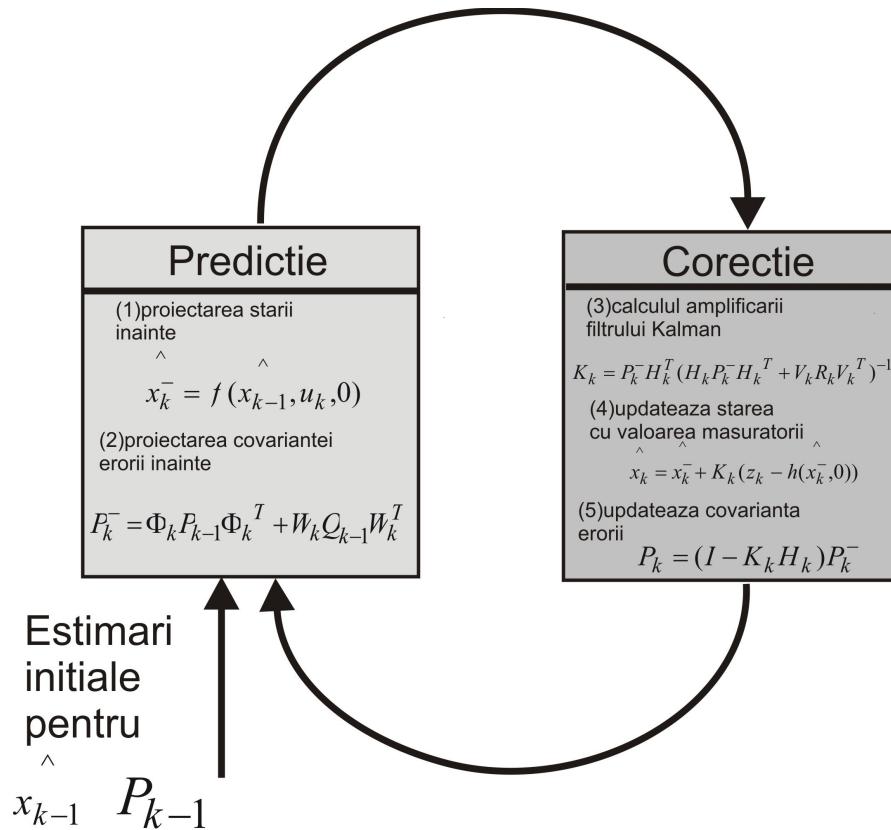


Figura 53 Modul de operare al EKF

Pornind de la ecuațiile generice ale EKF s-au considerat anumite simplificări care se regăsesc în ecuațiile de mai jos, care sunt cele implementate actual în aplicație.

$$\begin{aligned} \hat{x}_k^- &= f(\hat{x}_{k-1}^-, u_k, 0) \\ P_k^- &= \Phi_k P_{k-1} \Phi_k^T + Q_{k-1} \\ K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \\ P_k &= (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \quad (\text{forma Joseph}) \end{aligned}, \quad (100).$$

4.2.2.2 Detalii de implementare ale metodei diagnozei multi-model bazată pe utilizarea unui banc de filtre Kalman extinse.

Definirea tipurilor de defecte ce pot fi discriminate cu metoda utilizată

Metoda implementată pentru detecția și identificarea defectelor robotului se bazează pe utilizarea unui banc de filtre Kalman extinse. Metoda a fost dezvoltată prima dată de Magill, care a utilizat o structură paralelă de estimatoare pentru a estima modelul unui proces stochastic. Apoi Athans a utilizat un banc de filtre Kalman care oferea estimări ale stării către un număr egal de compensatoare LQG pentru a controla diferite regimuri de operare a unei aeronave. Fiecare estimator se baza pe un set de ecuații liniarizate ale sistemului în jurul unor puncte statice de funcționare. Mai târziu, Maybeck [83] a utilizat aceeași tehnică împreună cu o strategie de control adaptiv pentru controlul unei aeronave F-15. Metoda de estimare adaptivă multi-model a fost utilizată cu succes oferind rezultate foarte bune la detecția și identificarea defectelor în senzori și actuatori pentru avioane. În ultimii ani localizarea bazată pe filtre Kalman a fost foarte des utilizată și studiată în domeniul roboticii.[79]

Pentru aplicația implementată s-a utilizat un banc de filtre EKF cu 5 filtre. Benchmarkul de defecte considerat conține numai defecte la nivelul parametrilor sistemului, defecte mecanice, care determină modificarea comportamentului robotului în mod vizibil. Astfel am definit 2 clase de defecte, fiecare clasă conținând două tipuri de defecte. Astfel, prima clasă de defecte, introduce defectul de tip pană a roții, pentru fiecare roată a robotului. Prima clasă conține astfel 2 defecte, pană roată stânga și pană roată dreapta. La nivel de implementare defectul de tip pană se manifestă de fapt prin micșorarea razei roții cu o anumită valoare. Datorită micșorării razei unei roți are loc o modificare a întregului model cinematic al robotului și astfel se va putea observa în comportamentul său o deviație de la operarea nominală. A doua clasă de defecte introdusă are la baza defectul de tip denivelare periodică a roții, asemănătoare contextului în care un corp de masă neglijabilă dar de lungime dată devine solidar cu roata determinând astfel la momente definite în timp o modificare a razei roții cu valoarea lungimii corpului respectiv. Variația razei va fi astfel periodică și se va manifesta pentru o perioadă scurtă de timp. A doua clasă de defecte cuprinde astfel defect denivelare roată stânga și denivelare roată dreapta. Datorită faptului că defectele nu pot fi injectate fizic în timpul operării robotului din motive constructive, am ales injectarea lor printr-un framework software de injecție interactivă a defectelor. Utilizând aplicația client distribuită Java putem controla pe lângă startul/stopul operării robotului și momentele de injecție ale defectelor.

Efectiv funcțiile de injecție a defectelor acționează prin modificarea constantelor fizice ale robotului în calculele specifice sistemului odometric. Astfel în cazul în care nu este injectat nici un defect robotul va opera în regim nominal, sistemul odometric realizându-și obiectivele corect. În continuare este ilustrată structura modului de detecție, identificare a defectelor și reconfigurarea controlului bazat pe un banc de 5 filtre EKF.

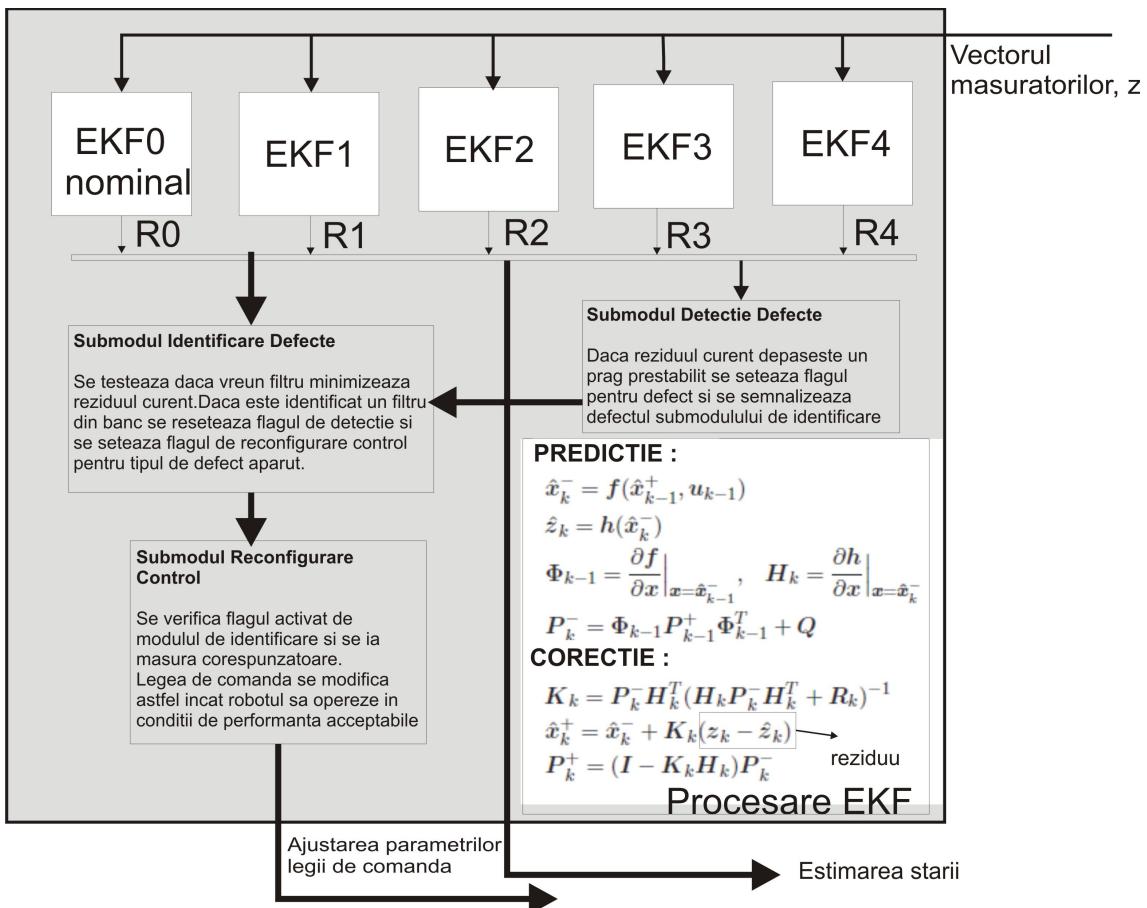


Figura 54 Modulul de detecție, identificare și reconfigurare

Fiecare EKF din banc înglobează în structura sa un model cinematic al robotului, dar cu parametrii diferiți. Ideea de bază este că pentru același vector de intrare z , afectat de zgomot, fiecare filtru realizează o predicție a stării robotului. Fiecare filtru este asociat unui anumit tip de defect. Filtrul EKF0 este filtrul nominal și corespunde situației nominale de operare fără defecte. Filtrul EKF1 înglobează modelul cinematic al robotului, dar cu parametrii modificați pentru a pune în evidență defectul, pană roată dreaptă, adică raza roții dreapta este micșorată și astfel predicția filtrului va fi de fapt valoarea vectorului de stare în prezența defectului respectiv. Filtrul EKF2 înglobează de asemenea modelul cinematic al robotului alterat

pentru defectul de tip, până roată stânga, sinonim cu faptul că valoarea razei pentru roata stânga este modificată și astfel filtrul va da o estimare a stării în prezența defectului respectiv. În mod analog celele două filtre EKF3 și EKF4 dau o predicție a vectorului de stare în prezența defectelor din a doua clasă de defecte. Pe lângă estimarea valorii vectorului de stare EKF generează și o estimare a vectorului de măsură în etapa de predicție (care e de fapt utilizată la estimarea valorii viitoare a stării, în etapa de corecție). Astfel, pe lângă predicția stării fiecare din cele 5 filtre EKF generează și un reziduu, care este definit ca diferența între valoarea reală a vectorului de măsură și valoarea predictată de filtru. După cum am menționat în începutul prezentării am ales o implementare a redundanței analitice care are la bază calculul reziduurilor. Calculul reziduurilor are o dublă importanță, în primul rând influențează predicția stării, dar este și un bun indicator al apariției defectelor.[82] Astfel metoda implementată se bazează pe analiza reziduurilor pentru detecția și identificarea defectelor.

Metoda de detecție are la bază calculul unui reziduu nominal care dacă depășește un anumit prag prestabilit (thresholding) semnalizează apariția unui defect, apoi ținând cont de acest semnal modulul de identificare verifică care este reziduul minim.[84] Această abordare se traduce prin faptul că la apariția unui defect reziduul nominal este afectat, datorită vectorului măsurătorilor (de la sistemul odometric) care este afectat de efectele defectului apărut, astfel reziduul va ieși dintr-un interval prestabilit al operării nominale. În partea de identificare ideea de bază este de a determina reziduul minim, în sensul că la apariția unui anumit defect vectorul de măsură e modificat și se apropie ca valoare de vectorul de măsură predictat de filtrul care încorporează modelul cu defectul respectiv.

4.2.2.3 Detecția defectelor și sinteza modulului de detecție a defectelor.

Identificarea defectelor și sinteza modulului de identificare a defectelor.

Reconfigurarea controlului pe baza tipului de defect diagnosticat

În continuare este prezentată sinteza efectivă a modulului de detecție și se face o analiză pe baza unor loturi de date preluate de la robot pentru a evidenția eficiența metodei. Pentru început se descrie modul de proiectare ale EKF pentru benchmarkul cu cele 4 defecte. Prima etapă a fost discretizarea modelului cinematic al robotului mobil pentru a-l îngloba în structura EKF. Astfel, robotul este descris de următorul sistem de ecuații :

$$\begin{cases} x(k+1) = x(k) + [r_r \omega_r(k) (\frac{\cos(\theta(k))}{2} - \frac{D \sin(\theta(k))}{L}) + r_l \omega_l(k) (\frac{\cos(\theta(k))}{2} + \frac{D \sin(\theta(k))}{L})] T_S, \\ y(k+1) = y(k) + [r_r \omega_r(k) (\frac{\sin(\theta(k))}{2} + \frac{D \cos(\theta(k))}{L}) + r_l \omega_l(k) (\frac{\sin(\theta(k))}{2} - \frac{D \cos(\theta(k))}{L})] T_S, \\ \theta(k+1) = \theta(k) + (\frac{r_r \omega_r(k) - r_l \omega_l(k)}{L}) T_S \end{cases}, \quad (101)$$

Utilizând formalismul descris anterior pentru proiectarea unui filtru EKF, se definește vectorul de stare al robotului ca fiind $x_k = [x \ y \ \theta]^T$, vectorul de intrare $u_k = [\omega_L \ \omega_R]^T$.

Sistemul este descris de ecuația 86,

$$\begin{aligned} x_k &= f(x_{k-1}, u_k, w_{k-1}), z_k = h(x_k, v_k) \\ f &= [f_x \ f_y \ f_\theta]^T, \\ f_x(k) &= x(k+1) \\ f_y(k) &= y(k+1) \\ f_\theta(k) &= \theta(k+1) \end{aligned}, \quad (102).$$

În implementarea practică au fost impuse anumite ipoteze simplificatoare. Astfel, considerăm matricile de covarianță a zgromotului de proces și de măsură, Q, R ca fiind invariante în timp și deviațiile standard ale zgromotului de proces pentru cele 3 coordonate $\sigma_x = \sigma_y = \sigma_\theta = 0.1$, iar deviația standard a zgromotului de măsură este $\sigma_{\text{mas}} = 0.01$. Matricile Q și R au următoarele

expresii, $Q = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_x^2 \end{bmatrix}$, $R = \begin{bmatrix} \sigma_{\text{mas}}^2 & 0 & 0 \\ 0 & \sigma_{\text{mas}}^2 & 0 \\ 0 & 0 & \sigma_{\text{mas}}^2 \end{bmatrix}$. Pentru partea de inițializări am considerat

$P_0^- = \begin{bmatrix} 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \end{bmatrix}$ și am considerat matricea Jacobian a derivatelor lui h în raport cu

vectorul de stare x_k , $H = I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Umătorul pas este calculul matriiciei Φ_k . Pornind de la

expresia de definiție 89 și calculând Jacobianul se obține următoarea expresie analitică,

$$\Phi_k = \begin{bmatrix} 1 & 0 & Ts(\frac{-r_R \omega_R(k) + r_L \omega_L(k)}{2} \sin(\theta(k)) - D \frac{r_R \omega_R(k) - r_L \omega_L(k)}{L} \cos(\theta(k))) \\ 0 & 1 & Ts(\frac{r_R \omega_R(k) + r_L \omega_L(k)}{2} \cos(\theta(k)) - D \frac{r_R \omega_R(k) - r_L \omega_L(k)}{L} \sin(\theta(k))) \\ 0 & 0 & 1 \end{bmatrix}, \quad (103)$$

Problema determinării și testării depășirii pragului impus de către reziduul nominal în contextul apariției unui defect presupune un calcul mai complex, plecând de la conceptul introdus anterior, pentru a asigura eficiența detecției. Astfel pentru a distinge funcționarea normală de funcționarea cu defecte se realizează un test statistic asupra reziduului nominal. Parametrii statistici ai reziduului (media și varianța) vor fi comparați on-line la fiecare perioadă de eșantionare cu valorile obținute pentru acești parametri în condițiile de operare nominală (fără defecte).[85] În condițiile nominale, fără defecte reziduul este o secvență de

zgomot alb de medie 0 și varianță $\eta_k = (H_k(x_k^-)P_k^-H_k^T(x_k^-) + R)$. Se emit astfel două ipoteze, ipoteza H_0 , corespunzând parametrilor statistici în operarea fără defecte și ipoteza H_1 , definită de un comportament anormal al robotului. Pragul de decizie statistic permite delimitarea a două zone distincte de funcționare a robotului. Prima regiune este caracterizată de valori ale reziduului care sunt considerate acceptabile, în timp ce a doua regiune marchează zona în care valorile reziduului nu sunt acceptabile (regiunea defectului) având valori peste pragul statistic. Astfel dacă considerăm regiunea defectelor notată cu π în spațiul discret R^n , o regiune de acceptare C_π este definită ca complementul lui π în spațiul R^n . Dacă considerăm un punct M reprezentând un eșantion (o reprezentare reală a procesului) atunci regula de decizie poate fi formulată astfel, ipoteza H_0 este adevarată dacă $M \in \pi$ și ipoteza H_1 este adevarată dacă $M \notin \pi$. Se utilizează o secvență standard a reziduului pentru ipotezele standard ale testelor

statistice dată de $\eta_s = (H_k(x_k^-)P_k^-H_k^T(x_k^-) + R)^{-1/2}(y_k - h_k(x_k^-)) = \eta_k^{-1/2} r_k$ cu medie 0 și varianță unitară. Orice abatere de la comportamentul normal va determina o modificare substanțială a acestor proprietăți statistice. Astfel ne propunem să determinăm estimarea

valorii reale a mediei unui eșantion dată de, $\hat{\eta}_s = \frac{1}{N} \sum_{j=1}^N \eta_{sj}$. În ipoteza H_0 adevarată $\hat{\eta}_s$ are o

distribuție gaussiană de medie 0 și covarianță I/N . Peste un anumit nivel de acceptare (ipoteza H_1) pe care-l putem nota ρ (prag impus) ipoteza H_0 este rejectată și deci $|\hat{\eta}_s| > \rho$.

Înainte de a trece la analiza modulului de detecție a defectelor se face o analiză a modului de operare a filtrelor EKF utilizate pentru implementarea diagnozei. Se vor testa individual cele 5 filtre și se vor analiza valorile reziduurilor generate și estimările stării. Astfel primul filtru analizat este cel nominal, care are încorporat un model cinematic al robotului fără defecte. Se simulează răspunsul EKF nominal pentru traectoria demonstrativă aleasă (circuit simplu).

În figura următoare este redată traectoria reală a robotului (obținută din loturi de date de la encoderele robotului), măsurările (efectuate de zgomot) și estimarea realizată cu filtrul Kalman extins,

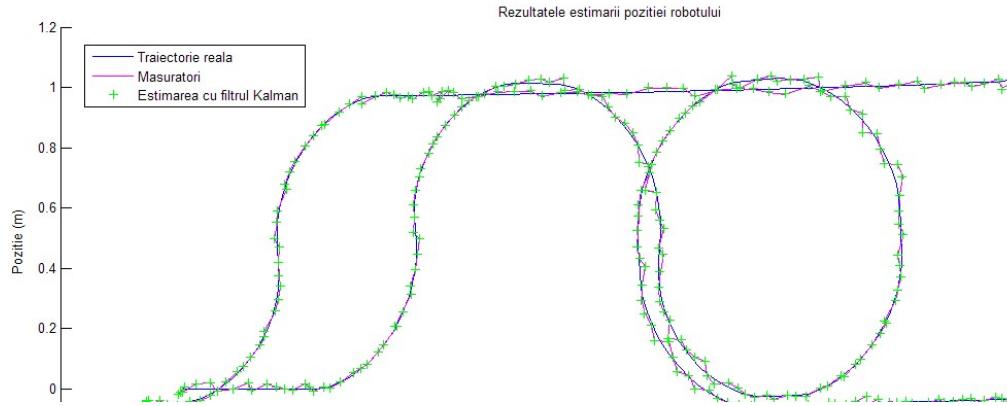
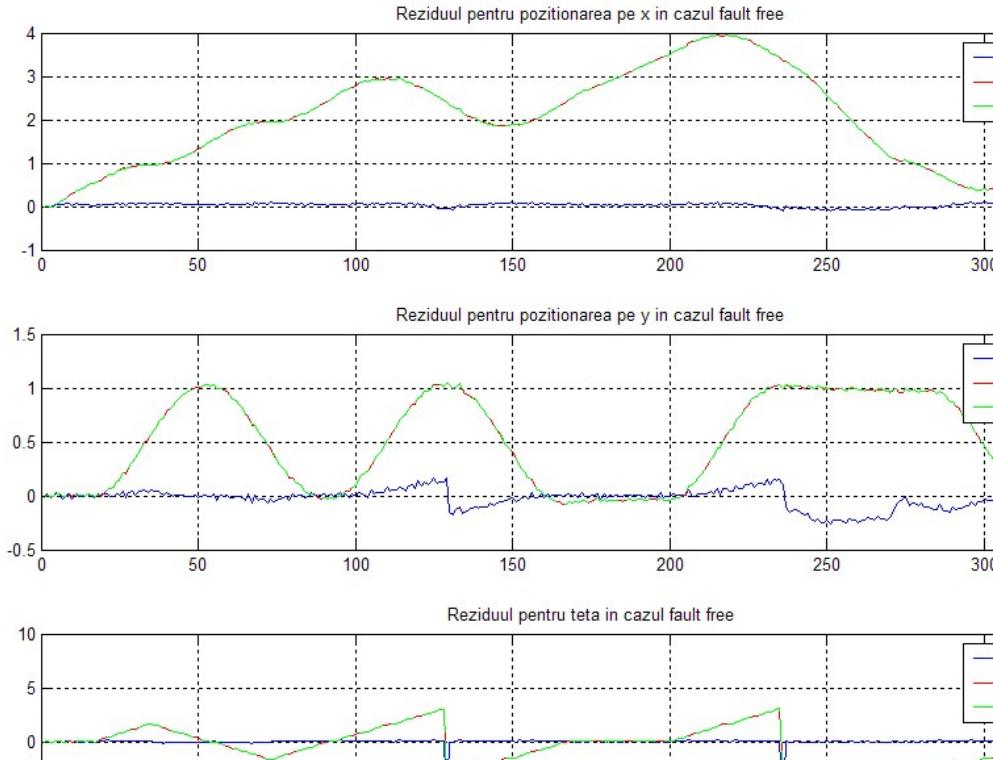


Figura 55 Analiza operării EKF nominal

Se poate observa că filtrul realizează o predicție bună a stării robotului în condițiile unei operări fără defecte. Reziduul nominal, definit ca diferența între măsurătoare și estimare măsurătorii, descompus pe componente este redat în continuare în figura 56,



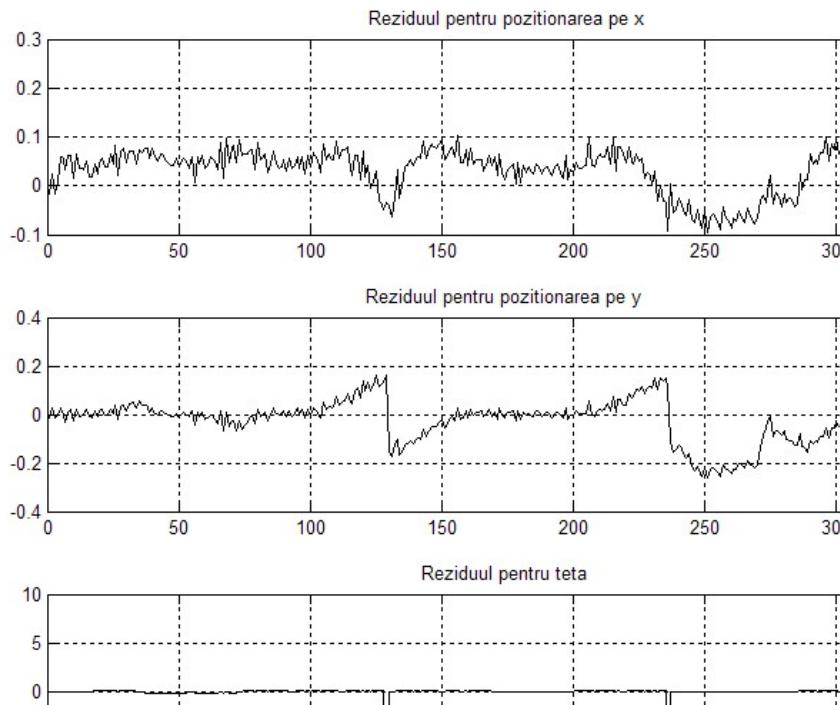


Figura 56 Analiza componentelor reziduului nominal

În cazul filtrului EKF1, care încorporează defectul pană roată dreapta, am ales introducerea defectului la eșantionul 100 în simulare pentru a studia și comportamentul reactiv, al filtrului. În figura următoare este ilustrată variația reziduului pe componente înainte și după injectarea defectului,

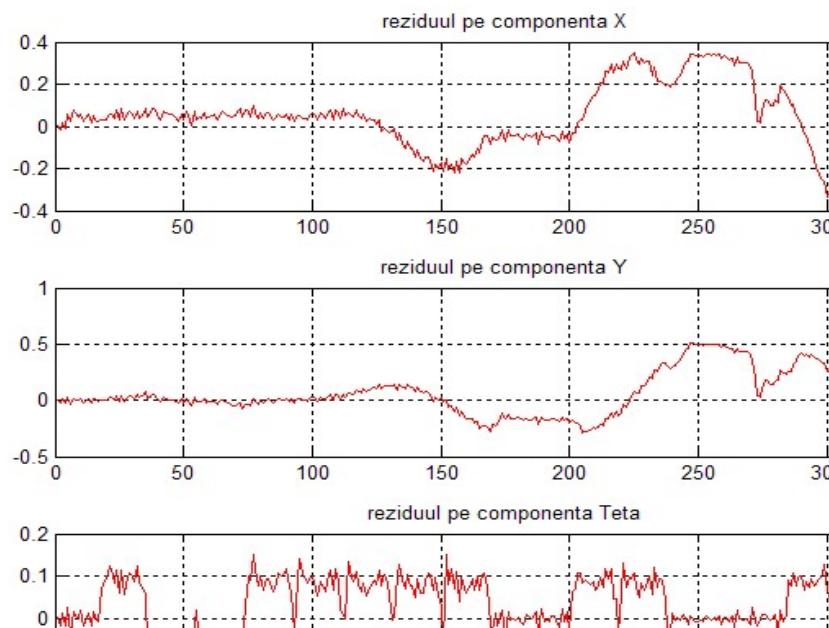


Figura 57 Analiza reziduului generat de EKF1

Se poate observa o variație semnificativă a componentelor reziduului începând de la eșantionul 100, cele mai vizibile fiind pe componente X și Y. De menționat că reziduurile figurate sunt cele generate de filtru. Pentru detecție, filtrul nominal în cazul defectului pană roată dreapta, va depăși limitele impuse, setând astfel semnalul pentru detecție, iar la etapa de identificare, EKF1 generând predicția pentru acest defect continuu va minimiza diferența între valoarea măsurată pentru o anumită componentă și predicția pentru măsurătoarea acelei componente (va minimiza reziduul) setând flagul tipului de defect corespunzător. În mod similar se poate face o analiză a reziduurilor pentru al doilea tip de defect din prima clasă și anume pană roată stânga. Astfel în figura 58 avem ilustrată variația reziduurilor la apariția unui defect la eșantionul 100,

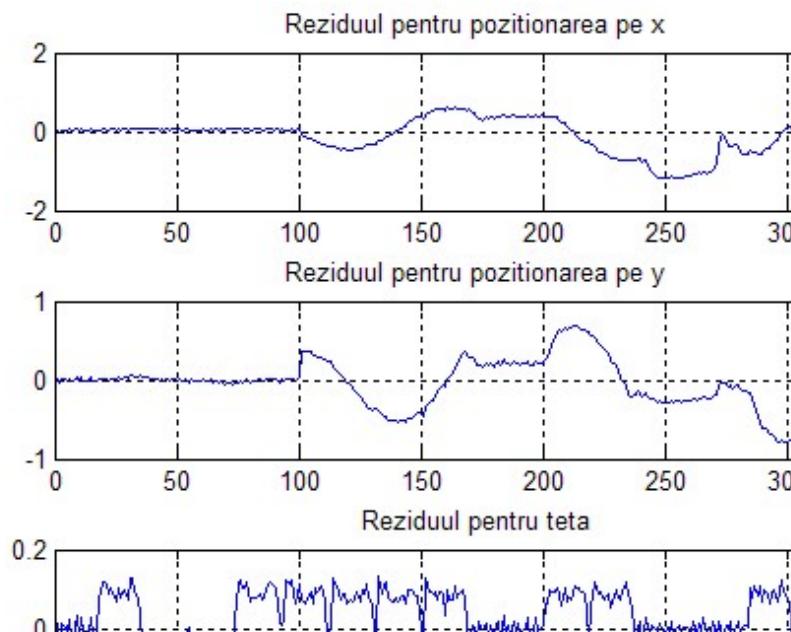


Figura 58 Analiza reziduului generat de EKF2

Alura reziduurilor este influențată de valorile măsurate și cele predictate de filtrul EKF. În cazul defectelor din a doua clasă reziduurile redau foarte bine comportamentul periodic al defectului. Astfel în figurile 59 și 60 avem figurate reziduurile generate de EKF3 și EKF4, pentru defectul denivelare roată dreaptă și stânga care apare la eșantionul 100,

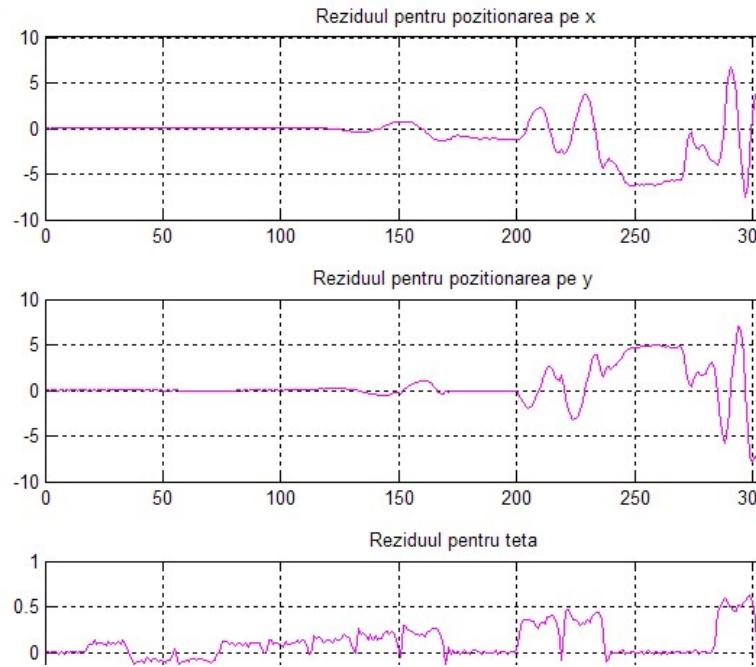


Figura 59 Analiza reziduului generat de EKF3

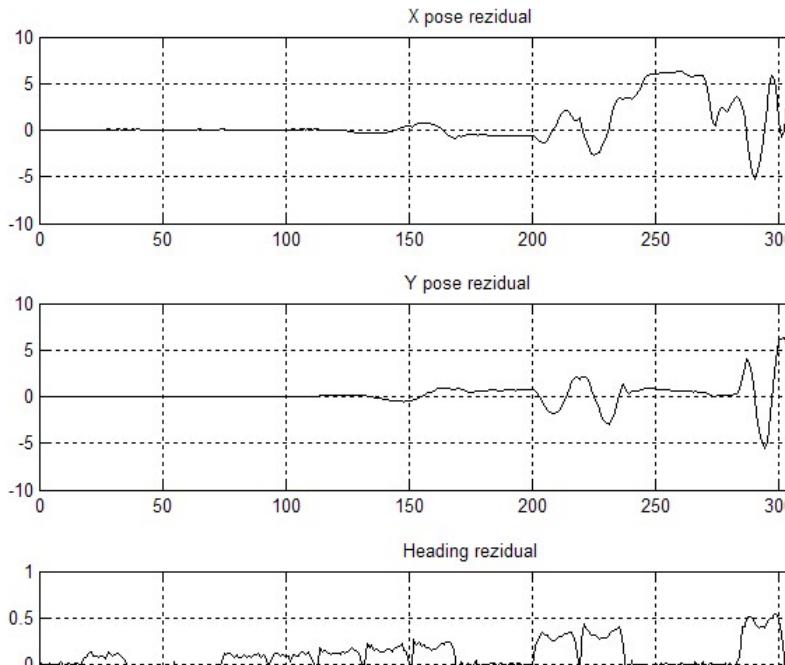
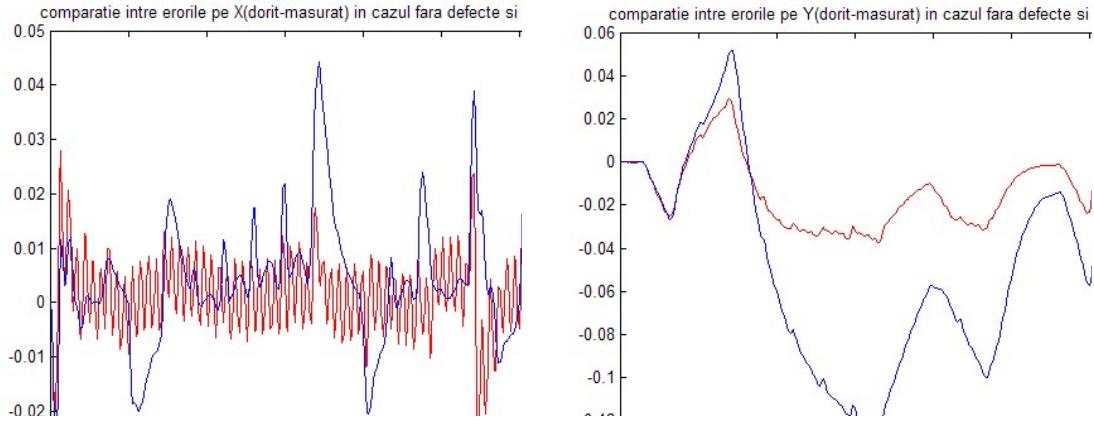


Figura 60 Analiza reziduului generat de EKF4

Se poate observa din analiza separată că fiecare filtru este sensibil la un anumit tip de defect și astfel utilizând aceste proprietăți utile am putut sintetiza modulele de detecție și identificare a defectelor. În figurile anterioare este redată funcționarea individuală a filtrelor, în continuare

se face o analiză a situației reale de operare. Astfel presupunând ca am injectat defectul de tip 1, pană roată dreapta, la momentul de timp 10s, putem observa manifestarea defectului în analiza erorii de poziționare, între cazul fără defect și cel cu defect în figurile 61,



Cu roșu este figurat cazul fără defect și cu albastru este figurat cazul în care defectul pană roată dreapta (raza inițială este micșorată cu 3 cm) este injectat.

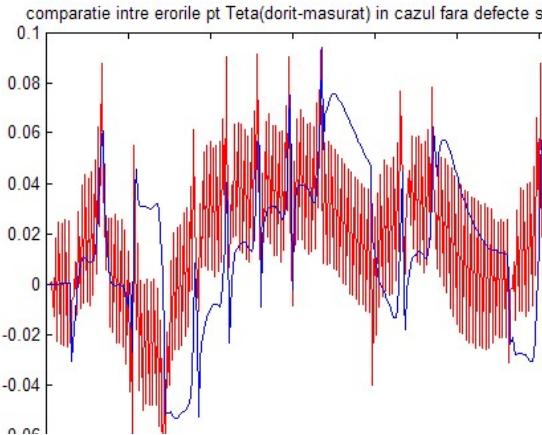


Figura 61 Marcarea apariției defectului 1, comportamentul în prezența defectului

În cazul injectării celui de al doilea defect pană roată stângă (micșorare a razei roții stânga cu 4cm) la momentul de timp 20s se poate observa cum comportamentul robotului nu mai este identic cu cel anterior, însă defectul este vizibil în analiza erorii și mai ales este detectat de modulul de detecție și identificat ca fiind din prima clasă de defecte. În figura 62 sunt analizate rezultatele.

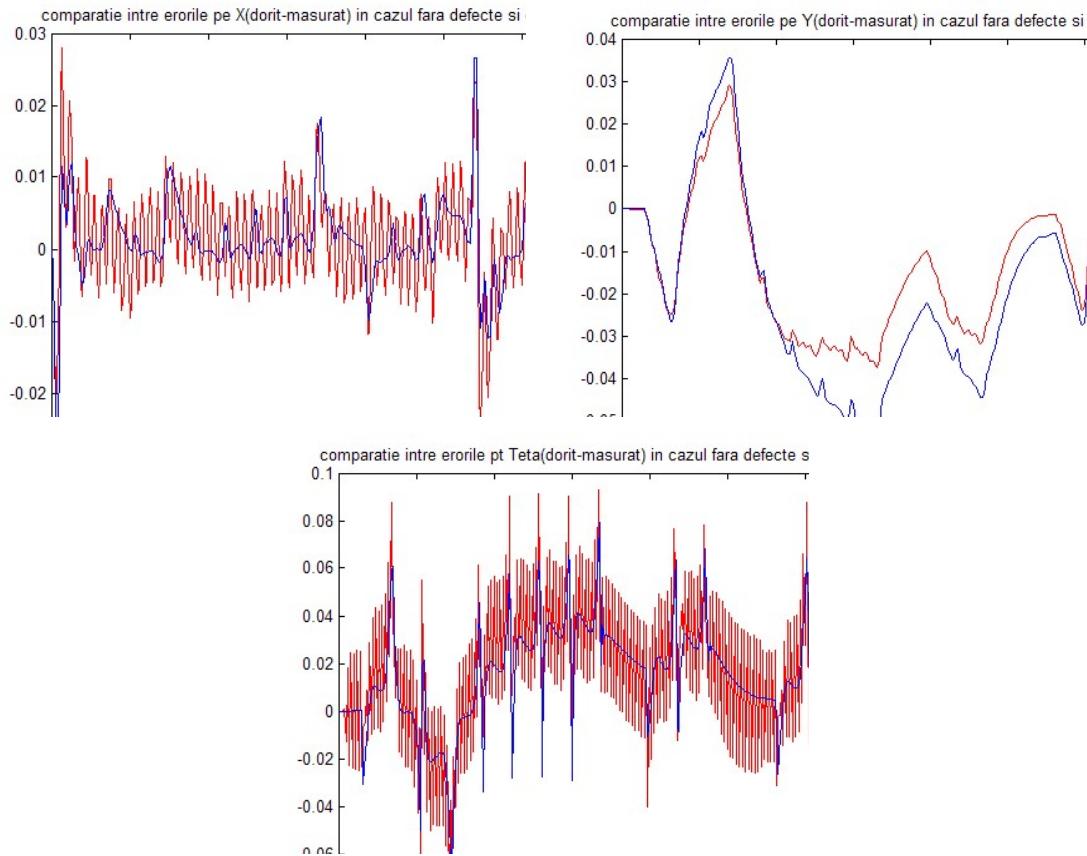
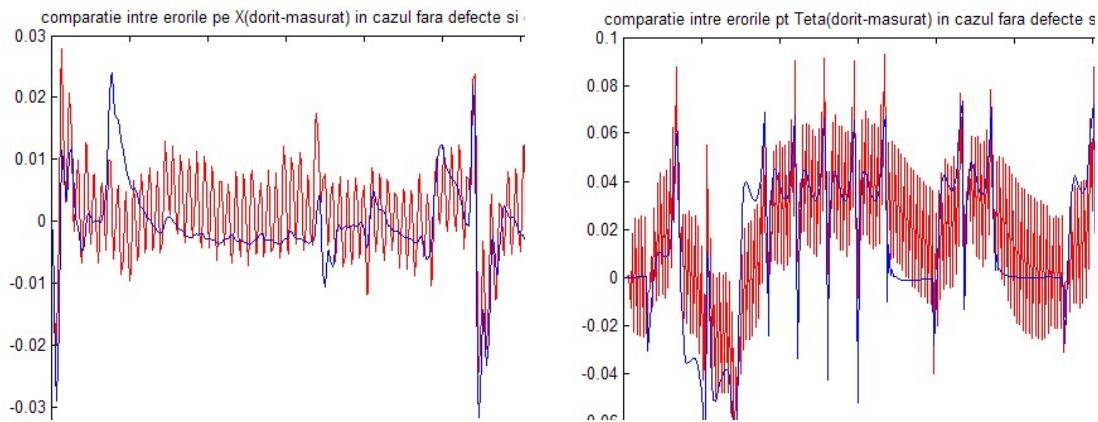


Figura 62 Marcarea apariției defectului 2, comportamentul în prezența defectului

Următorul defect injectat este defectul denivelare periodică roată dreapta (denivelare 3cm) aplicat la momentul de timp 7s. În figura 63 sunt analizate rezultatele obținute.



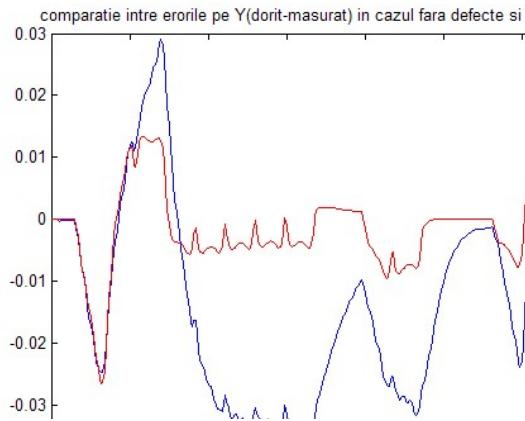


Figura 63 Marcarea apariției defectului 3, comportamentul în prezența defectului

În final ultimul defect injectat este defectul de tip 4 , denivelare roata stânga (denivelare 2cm) aplicat la momentul de timp 16s. În figura 64 sunt ilustrate rezultatele.

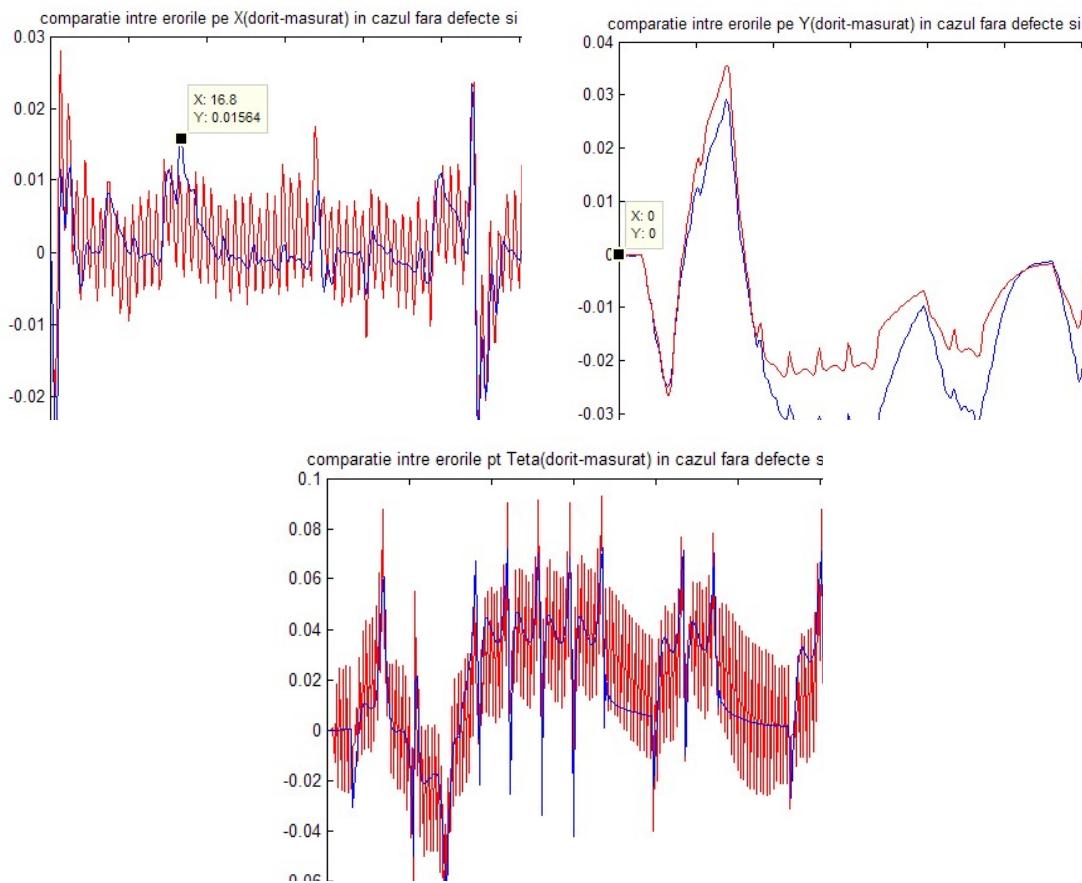
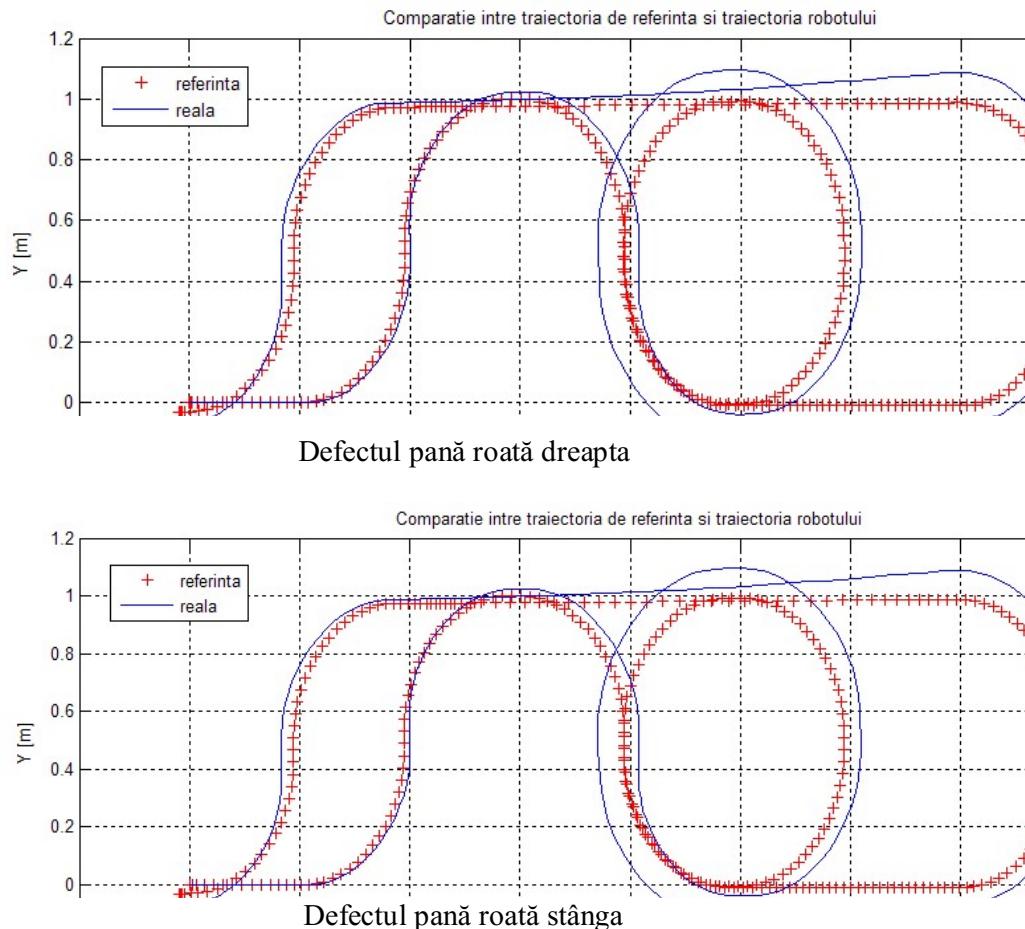


Figura 64 Marcarea apariției defectului 4, comportamentul în prezența defectului

Un aspect important de menționat în analiza tuturor cazurilor în care apar defecte se referă la intervalul de timp de la injectare și până când defectul este detectat și se manifestă efectiv în comportamentul sistemului. Astfel, în cazul unor defecte manifestarea este mai rapidă pe o componentă și mai lentă pe celelalte componente sau poate chiar să nu fie vizibilă pe celelalte componente. Această problemă a determinat restrângerea doomeniului de discriminare a defectelor, în implementarea curentă, doar la nivelul clasei de defecte. Astfel sistemul dezvoltat detectază toate cele 4 tipuri de defecte însă identifică doar clasa din care face parte defectul, funcție de specificul lui de manifestare în comportamentul robotului. Înainte de a trece la descrierea modulului de identificare se prezintă câteva rezultate privind modul în care robotul operează după injectarea defectului, fără nici o măsură de reconfigurare a controlului. Această analiză e utilă pentru dezvoltarea ulterioară a unei metode de reconfigurare a controlului specifică sistemului curent. Astfel se prezintă în figura 65 traectoriile robotului după injectarea defectelor din cele două clase,



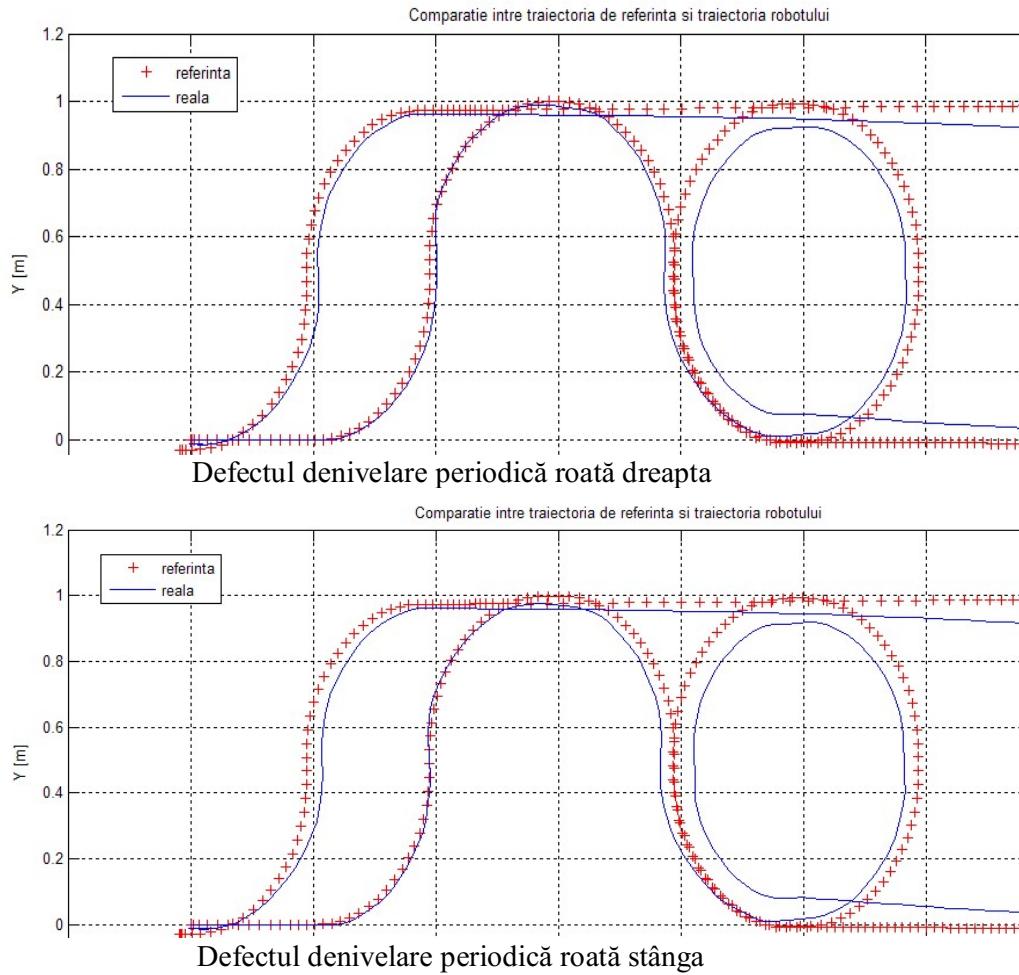


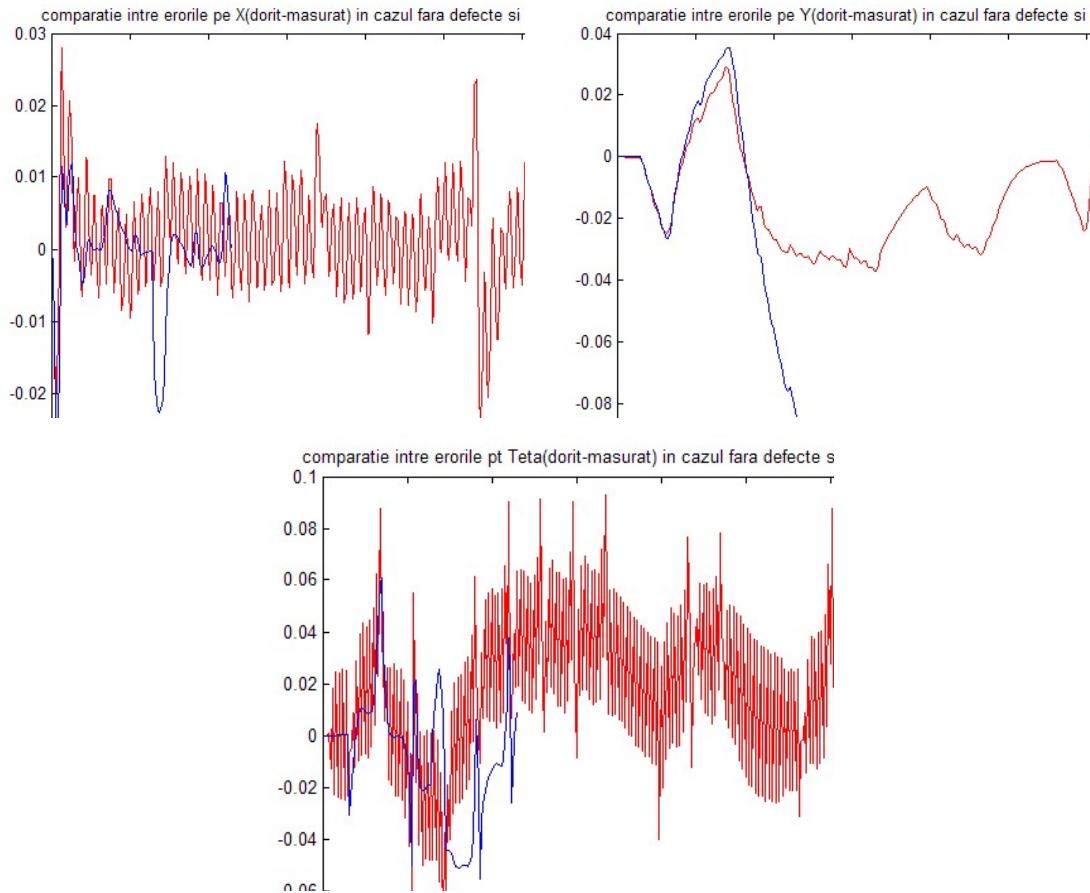
Figura 65 Traiectoriile robotului după injectarea defectelor

Se observă o alterare a traiectoriei robotului în mod specific fiecărui defect injectat. Amplitudinea efectului manifestat în comportamentul robotului este direct proporțională cu valorile impuse pentru variațiile parametrilor robotului.

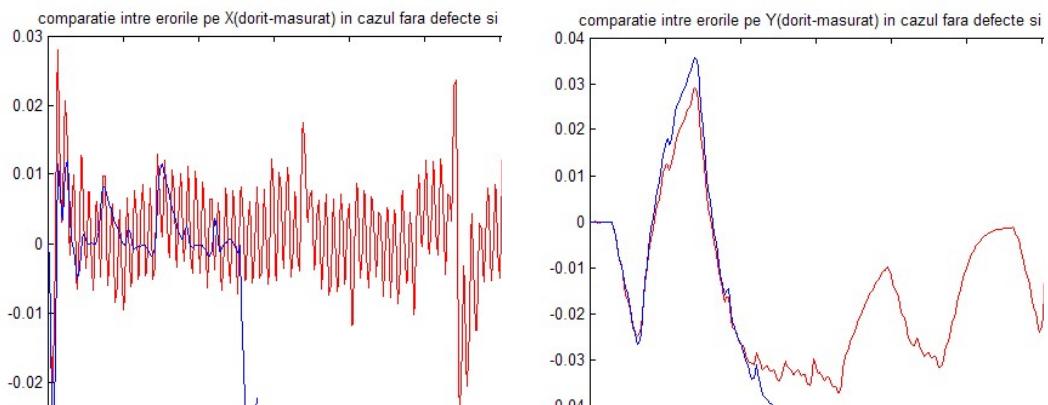
În cele ce urmează se analizează profilul erorii la momentul detecției defectului pentru fiecare clasă de defecte și latența între momentul apariției defectului și detecția sa - identificarea sa.

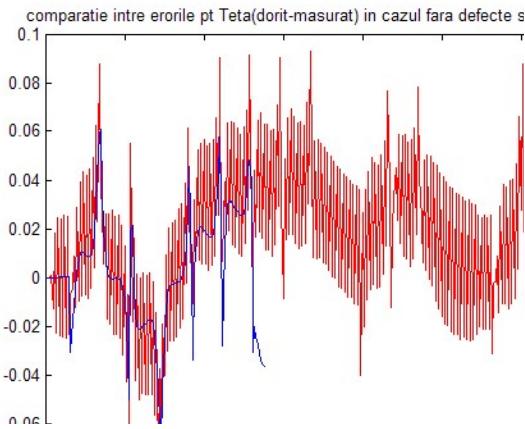
Latența între momentul apariției / injectării defectului și momentul detecției acestuia este stâns legată de tipul defectului, în sensul că puteam observa o latență relativ scăzută pentru prima clasă de defecte (care se manifestă mai intens în timp în comportamentul robotului) și o latență ridicată în ceea ce privește a doua clasă de defecte (care determină o alterare vizibilă a comportamentului robotului doar pentru valori mari ale denivelării). Figura 66 redă o ilustrare a acestor aspecte.

Defectul de tip pană roată dreapta e injectat la 14s și detectat la 23s deci latența este de 9s.

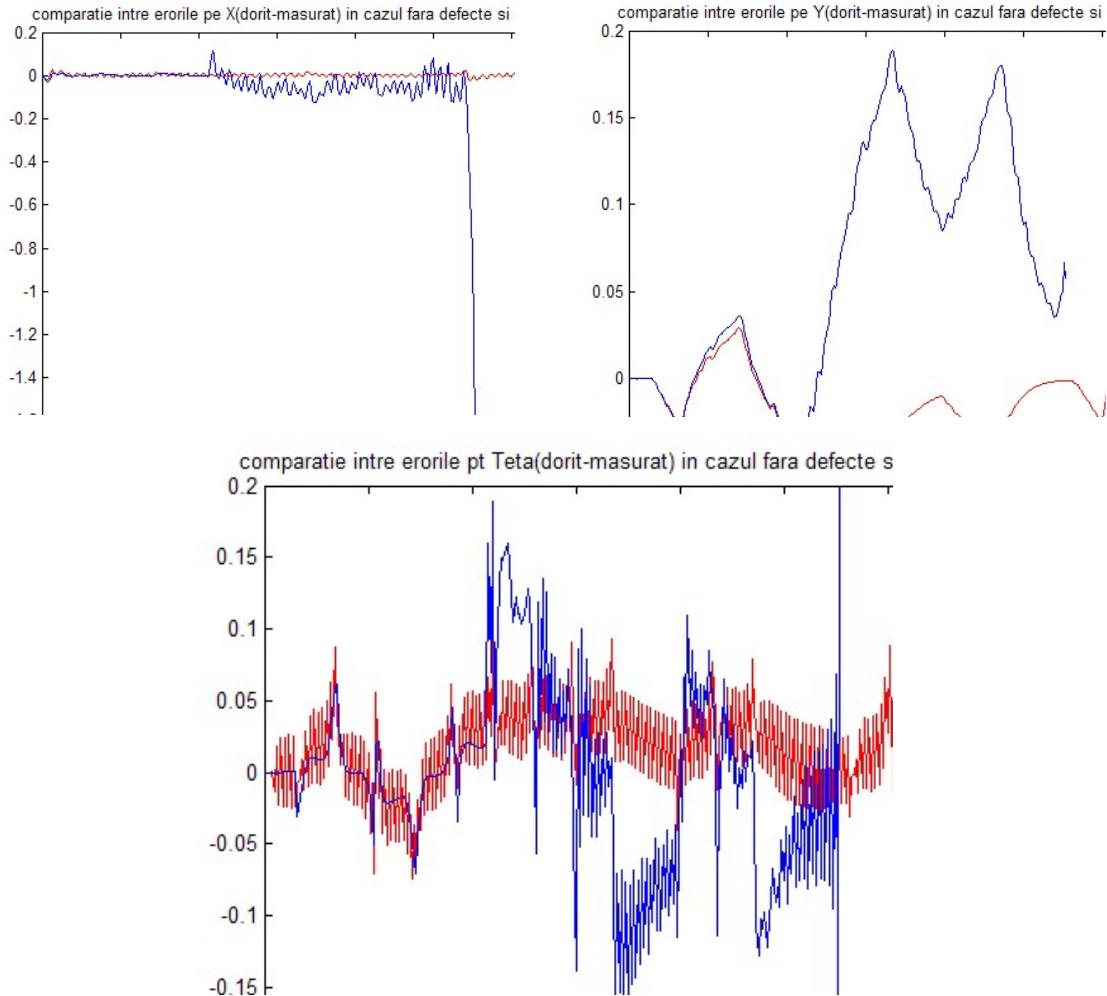


Defectul tip pană roată stânga a fost injectat la 26s și detectat foarte repede la 28s, latență specifică este minimă de 2 s.





Defectul 3 a fost injectat la 22s și detectat la 55s , latența este de 33s, defectul se manifestă greu în comportamentul robotului, pentru valori mici ale amplitudinii denivelării (2cm).



Defectul 4 a fost injectat la momentul $t=10s$ și detectat la $t=32s$, latența fiind de 22s

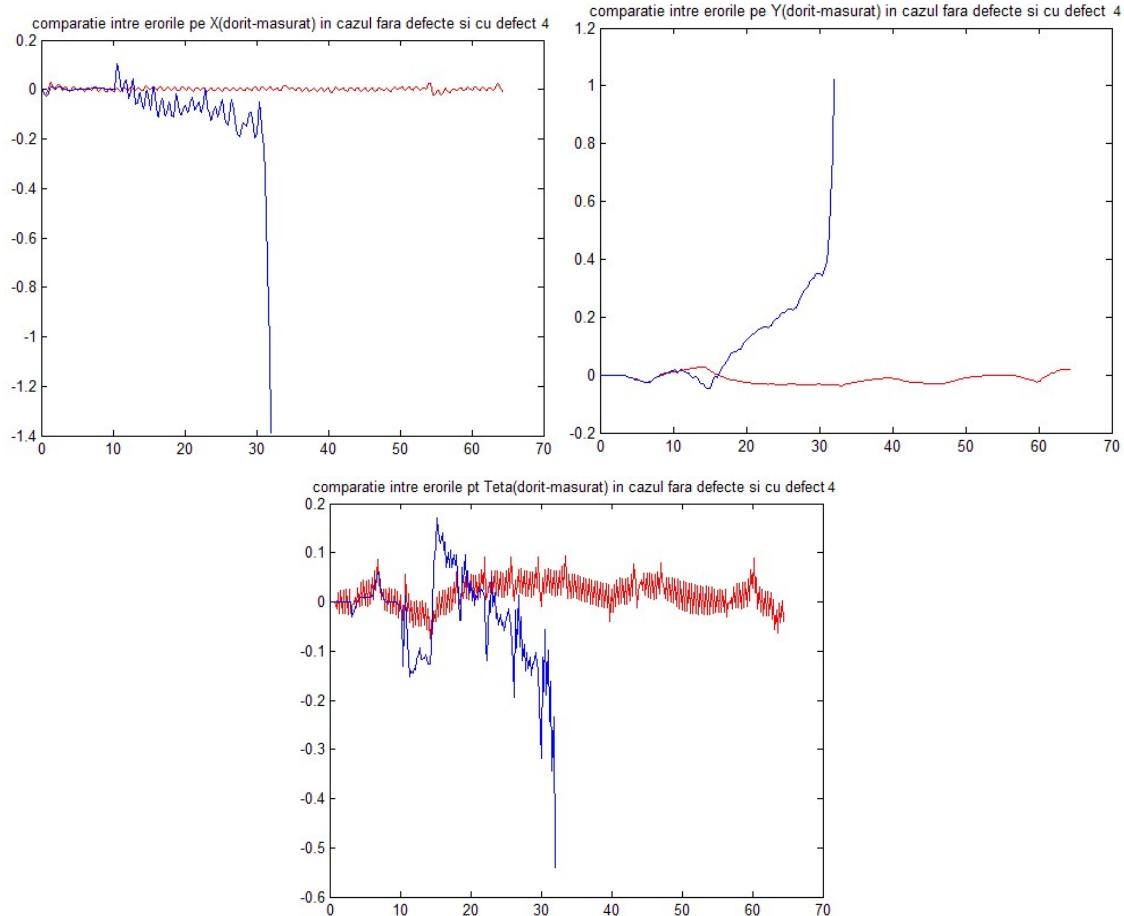


Figura 66 Analiza contextului de detecție al defectelor

Un alt aspect deocamdată ignorat și de mare importanță și interes pentru îmbunătățirile viitoare se referă la posibilitatea dezvoltării unui algoritm care să poată discrimina între profilurile de timp ale apariției defectelor multiple, care se manifestă în reziduu.

Identificarea defectelor se realizează testând care dintre cele 4 reziduuri generate de filtrele sensibile la defecte este minim. Astfel practic se creează un vector cu 4 componente reprezentând valorile reziduurilor generate de cele 4 EKF. Apoi, se testează dacă flagul de detecție defect a fost setat, în caz afirmativ se sortează elementele vectorului de reziduuri și se verifică componentă cu componentă dacă profilul reziduului minim (obținut la sortare) este același cu profilul unuia din cele 4 reziduuri. Dacă minim două componente ale unui reziduu generat aparțin reziduului minim, atunci defectul apărut este defectul la care este sensibil filtrul care a generat reziduul respectiv. Astfel determinând indexul acelui filtru putem discrimina care defect a apărut. Un alt aspect practic se referă la apariția mai multor defecte

simultan și apariția-dispariția manifestării unui defect. În cazul aplicației dezvoltate un defect(și doar unul), care apare este detectat și identificat, aplicația neavând implementată facilitatea utilizării unei cozi cu priorități pentru tratarea defectelor apărute în paralel (deschidere pentru viitoarele direcții de studiu). Pentru a două problemă a manifestării sporadice a unui defect aplicația semnalează în fluxul de interacțiune între modulele de diagoză apariția defectului și flagul corespunzător va rămâne setat până la finalizarea misiunii robotului sau până la resetarea aplicației.

Modul în care robotul se comportă după apariția / injectarea defectului definește etapa de fault accomodation (acomodarea defectului). În aplicația curentă utilizarea unui controller sliding mode a influențat în mod pronunțat caracterul tolerant la defecte. Fiind un controller robust acesta a reușit în cele mai multe situații să asigure o comandă potrivită pentru a asigura operarea robotului și după apariția / injectarea defectului, altfel spus prin utilizarea controllerului sliding mode am asigurat o toleranță la defecte intrinsecă. Toleranța la defecte intrisecă este susținută și de faptul că actuatorii nu sunt perfect identici (actuator mismatch) fapt care determină tempi de răspuns diferenți pentru aceștia, problemă care a fost compensată cu succes de către controllerul sliding mode. S-ar putea spune că nivelul de toleranță la defecte este solidar cu nivelul de control în sensul unei extensii a modului de reconfigurare a controlului la nivelul însuși a controllerului sliding mode. Efectiv metoda de reconfigurare a controlului nu este una bine definită întrucât necesită un studiu de analiză structurală și o analiză a modalităților posibile de ajustare on-line a parametrilor controllerului sliding mode. Astfel raportându-ne la aplicația dezvoltată, în momentul în care defectul a fost detectat și identificat ca aparținând unei anumite clase se setează un flag care marchează posibilitatea activării unei metode sau unei măsuri specifice de reconfigurare a controlului pentru defectul apărut.

În continuare sunt prezentate concluziile generale ale dezvoltării aplicației, urmărind aspectele importante privind implementarea și posibilele îmbunătățiri viitoare care se pot aduce la nivelul structurii hardware, aplicației de conducere în timp real, la nivelul algoritmului de control și la nivelul sistemului de monitorizare și diagoză.

Capitolul 5

| Concluzii și direcții viitoare de studiu

Concluzii și direcții viitoare de studiu

Pe parcursul lucrării de față s-a realizat o prezentare amplă și o analiză amănunțită a sistemului mechatronic dezvoltat și aplicației software pentru controlul tolerant la defecte. Realizările obținute sunt analizate în continuare la fiecare nivel al implementării, urmărind apoi și eventuale propuneri de îmbunătățire și direcțiile viitoare de studiu. Astfel primul nivel se referă la structura hardware a robotului. În acest context pentru tema de lucru propusă s-a pornit de la premisa dezvoltării unei structuri minimale care poate implementa concepțele teoretice dezvoltate în prealabil. S-a dezvoltat astfel o structură mobilă cu locomoție diferențială și o rețea senzorială formată din numărul minim de senzori (2 senzori) pentru realizarea obiectivelor impuse pentru operarea în regim de trajectory tracking. Ca propuneri de îmbunătățire la acest nivel se poate aminti utilizarea unui nou set de actuatori, extinderea rețelei senzoriale cu senzori utili pentru validarea odometriei (redundanță fizică, accelerometru, giroscop) și care să permită și operarea în regim adaptiv de ocolire obstacole în mediu static și dinamic (sonare, laser, senzori proximitate infraroșu). Direcțiile viitoare de studiu pentru acest nivel se referă la posibilitatea înlocuirii roții pasive de tip castor cu un element director de tip sferă (care conferă stabilitate suplimentară, mărește gradul de manevrabilitate și elimină unele probleme care apar la operarea robotului în contextul în care roata pasivă este poziționată în sensul contrar mișcării șasiului) și proiectarea unei noi structuri a șasiului pentru utilizarea unui nou sistem de locomoție (bazat pe roți suedeze) care să determine obținerea unui vehicul holonomic. Nivelul aplicației de control s-a suprapus peste aplicația cu facilități real time. În acest context al aplicației real time s-au obținut rezultate bune în ceea ce privește minimizarea latenței, prin mecanisme specifice SO și asigurarea unui suport eficient de implementare a operațiilor de I/O la perioade de eșantionare acceptabile la nivelul aplicației (50-200ms). Ca viitoare direcții de studiu la acest nivel atenția se îndreaptă către utilizarea strictă a unei platforme embedded și avetual asigurarea unei metode simple și eficiente de portare a aplicației pe alte platforme fapt care ar mări gradul de flexibilitate și aplicabilitate al aplicației. La nivelul aplicației, implementarea algoritmului de control a fost asigurată prin crearea unei structuri de clase cu funcționalitate extinsă, care determină utilizarea oricărei tehnici de control. Avantajele utilizării controlului sliding mode au fost clar prezентate anterior. Astfel robustețea în prezența incertitudinilor și perturbațiilor a determinat utilizarea cu succes în aplicația de trajectory tracking. Pentru a

conferi o și mai mare putere tehnicii de control utilizate se poate introduce o metodă de ajustare on-line a parametrilor controllerului (analitică și practică) care ar extinde caracteristicile de robustețe și precizie ale controllerului. Un aspect important de menționat se referă la alegerea perioadei de eșantionare, care în cazul implementării curente a fost limitată doar de timpii de răspuns mari și diferenți ai actuatorilor (problemă actuator mismatch) și de rezoluția scăzută a senzorilor (codare 500PPR). În acest context controllerul sliding mode poate fi încouit cu orice altă metodă de control bazată pe modelul cinematic al robotului, de exemplu utilizarea unei rețele neuronale sau a unui control fuzzy. În final, la nivelul superior al taskului de monitorizare și diagnoză s-au obținut rezultatele propuse prin tema de proiectare și anume detecția defectelor din benchmarkul propus și discriminarea defectelor. Efectiv benchmarkul creat a avut rolul de a demonstra eficiența metodei bazate pe filtrul Kalman extins pentru detecția și identificarea defectelor. Având o implementare relativ facilă, întradevăr după o analiză complexă a elementelor implicate, soluția utilizării unui banc de filtre EKF pentru detecția celor 4 defecte și identificarea claselor de apartenență s-a dovedit a fi una viabilă. Benchmarkul utilizat poate fi extins și prin introducerea unor noi tipuri de defecte ale senzorilor și / sau actuatorilor, fapt care ar crește numărul filtrelor EKF din banc și ar mări efortul computațional care este considerabil și în cazul de față când calculul matriceal (inverse, înmulțiri) este intens. Ideea de bază din spatele acestei metode s-a concentrat pe utilizarea caracteristicilor de estimator de stare ale EKF și posibilitatea generării unei estimări a stării robotului care comparată cu loturile de date de la robot au oferit suficientă informație pentru detecția comportamentului anormal în sistem. Ca viitoare direcții de studiu în acest context intenționez să concentrez pe îmbunătățirea nivelului de identificare utilizând o rețea neuronală și dezvoltarea unor algoritmi pentru reconfigurarea controlului în etapa de fault accommodation (acomodare a defectului) care să ofere posibilitatea creării unui framework generic pentru utilizarea în cazul roboților mobili. Un ultim aspect se referă la implementarea unui sistem SLAM pentru planning peste nivelul actual, evident în cazul unei extensii a structurii hardware existente.

Prin dezvoltarea sistemului robotic și implementarea aplicației de control tolerant la defecte am încercat testarea unor noi posibilități de abordare în operarea autonomă, individuală și în siguranță a roboților mobili care se poate continua prin extinderea conceptelor și ideilor de lucru într-un mediu robotic colaborativ din industrie sau alte domenii (agrement, turism, asistență persoane cu handicap sau în vîrstă, explorare, militar).

Capitolul 6

| Bibliografie

Bibliografie, referințe

Capitolul 1

1. M Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki - Diagnosis and Fault-Tolerant Control, Springer, 2006.
2. Chen, J., & Patton, R. J. - Robust model-based fault diagnosis for dynamic systems , Massachusetts: Kluwer Academic Publishers, 1999.
3. Chow, E. Y., & Willsky, A. S. - Analytical redundancy and the design of robust failure detection systems. IEEE Transactions on Automatic Control 29, 1984.
4. Frank, P. M. - On-line fault detection in uncertain nonlinear systems using diagnostic observers: a survey. International Journal Systems Science 25, 1994.
5. Frank, P. M., Ding, S. X., & Marcu, T. - Model-based fault diagnosis in technical processes. Transactions of the Institution of Measurement and Control 22, 2000.
6. Frank, P. M., & Wunnenberg, J. - Robust fault diagnosis using unknown input observer schemes. In R. J. Patton, P. M. Frank & R. N. Clark (Eds.), Fault diagnosis in dynamic systems: theory and applications . NY: Prentice Hall, 1989.
7. Gertler, J. - Residual generation in model-based fault diagnosis. Control-theory and advanced technology, 1993
8. Gertler, J. - Fault detection and diagnosis in engineering systems, 1998.
9. Gertler, J., Fang, X., & Luo, Q. - Detection and diagnosis of plant failures: the orthogonal parity equation approach. Control and Dynamic Systems 37, 1990.
10. Gertler, J., & Luo, Q. - Robust isolable models for failure diagnosis. AIChE 31, 1989.
11. Huang, Y., Dash, S., Reklaitis, G.V., & Venkatasubramanian, V. - EKF based estimator for FDI in Model IV FCCU. IFAC Proceedings SAFEPROCESS2000 Budapest, Hungary, 2000.
12. Isermann, R. - Process faults detection based on modelling and estimation methods a survey. Automatica 20, 1984.
13. Isermann, R. - Process fault diagnosis based on dynamic models and parameter estimation methods. In R. J. Patton, P. M. Frank & R. N. Clark (Eds.), Fault diagnosis in dynamic systems: theory and applications . NY: Prentice Hall, 1989.
14. Kramer, M. A., & Mah, R. S. H. - Model-based monitoring., Proceedings of the second international conference on ‘foundations of computer-aided process operations’ (FOCAPO), 1993.
15. Patton, R., Frank, P., & Clark, R. - Fault Diagnosis in Dynamic Systems: Theory and Applications . New York: Prentice Hall, 1989.
16. Yin, K., & Gertler, J.- Fault detection and isolation in the directional residual approach. In Preprint IFAC workshop on online fault detection and supervision in the chemical and process industries (pp. 194_ 199). Newcastle, 1995.
17. Watanabe, K., & Himmelblau, D. M. - Instrument fault detection in systems with uncertainties. International Journal System Science 13, 1982.

18. Chang, C. C., & Yu, C. C. - On-line fault diagnosis using the signed directed graph. Industrial and Engineering Chemistry Research 29, 1990.
19. Davis, R. - Diagnosis reasoning based on structure and behavior, 1984.
20. De Kleer, J., & Brown, S. - A qualitative physics based on confluences, 1984.
21. Fussell, J. B. - Fault tree analysis state of the art, IEEE Transactions on Reliability 23, 1974.
22. Milne, R. - Strategies for diagnosis, IEEE Transactions on Systems, Man and Cybernetics 17, 1987.
23. Garcia, C. E., Ramaker, B. L., & Pollard, J. F. - Total process control beyond the design of model predictive controllers, 1991.
24. Gertler, J., & McAvoy, T.J. - Principal component analysis and parity relations a strong duality. In IFAC SAFEPROCESS'97, 1997.
25. Janusz, M., & Venkatasubramanian, V. - Automatic generation of qualitative description of process trends for fault detection and diagnosis. Engineering Applications of Artificial Intelligence 4, 1991.
26. Henley, E.J. - Application of expert systems to fault diagnosis. In AIChE annual meeting , San Francisco, 1984.
27. Mufeed M. Mahmoud, Jin Jiang, Youmin Zhang - Active Fault Tolerant Control Systems, Stochastic Analysis and Synthesis, Springer, 2003.
28. George Vachtsevanos, Frank Lewis, Michael Roemer, Andrew Hess, Biqing Wu - Intelligent fault diagnosis and prognosis for engineering systems, 2000.

Capitolul 2

29. Bruno Siciliano, Oussama Khatib (Eds.) - Springer Handbook of Robotics, 2008
30. Shuzhi Sam Ge, Frank L. Lewis - Autonomous Mobile Robots, Sensing, Control, Decision Making and Applications, CRC Press, 2008.
31. Paul E. Sandin - Robot Mechanisms and Mechanical Devices Illustrated, 2007.
32. J. Borenstein, H. R. Everett, and L. Feng - Where am I? Sensors and Methods for Mobile Robot Positioning .
33. Owen Bishop - Robot Builder's Cookbook, Elsevier, Newnes, 2007.
34. Mark W. Spong, Seth Hutchinson, M. Vidyasagar - Robot Modeling and Control , JOHN WILEY & SONS, INC., 2007.
35. Krzysztof Kozłowski (Ed.) - Robot Motion and Control , 2006, Springer
36. Wolfram Burgard, Rüdiger Dillmann, Christian Plagemann, Nikolaus Vahrenkamp - Intelligent Autonomous Systems 10, IOS Press, 2006.
37. J.-P. Laumond (Ed.) - Robot Motion Planning and Control , Springer, 2003.
38. Bruno Siciliano , Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo - Robotics, Modelling, Planning and Control , Springer , 2009.
39. Srikanta Patnaik, Lakhmi C. Jain, Spyros G. Tzafestas, Germano Resconi, Amit Konar - Innovations in Robot Mobility and Control , Springer, 2005.

40. Roland Siegwart and Illah R. Nourbakhsh - Introduction to Autonomous Mobile Robots , A Bradford Book, 2006
41. John Holland - Designing Autonomous Mobile Robots, Elsevier, Newnes, 2004 .
42. Wolfram Burgard, Oliver Brock, and Cyrill Stachniss - Robotics, Science and Systems III, The MIT Press, 2007.
43. Thomas Bräunl - Embedded Robotics, Mobile Robot Design and Applications with Embedded System , Springer, 2006.
44. Nehmzow, U - Mobile robotics, a practical introduction, Springer, 2003.
45. Kurfess T.R. - Robotics and automation handbook, CRC Press, 2005.
46. Craig J. J - Introduction to Robotics: Mechanics and Control, Prentice Hall, 2004.
47. W. Leonard - Control of Electric Drives, Verlag, Berlin, 1985.

Capitolul 3

48. Axenie Cristian s.a. - A Client-Server Based Real-Time Control Tool for Complex Distributed Systems, Real-Time Linux Workshop 2007, Linz, Austria.
(<ftp://ftp.realtimelinuxfoundation.org/pub/events/rtlws-2007/Axenie.pdf>)
49. Karim Yaghmour – Adaptive Domain Environment for Operating Systems.
50. Philippe Gerum - Life with Adeos, 2005.
51. RTAI User Manual 3.4 – rev 0.3.
52. Herman Bruyninckx - Real-Time and Embedded Guide, K.U.Leuven, Mechanical Engineering, 2002
53. K Computing - Embedded and real-time Linux development.
54. Charles Curley - Open source software for real-time solutions. Linux Journal.
55. Kevin Dankwardt - What is real time? and benchmarks on real time Linux - Part 1,2,3.
56. Dipartimento di Ingegneria Aerospaziale Politecnico di Milano. About RTAI, 2002.
<http://www.aero.polimi.it/rtai/about/index.htm>
57. Phil Wilshire - Real time Linux: Testing and evaluation. In Real Time Linux Workshop, Orlando, FL, 2000.
58. <http://www.comedi.org/doc/index.html>
59. Doreen L. Galli - Distributed Operating Systems, Springer, 2000.

60. Burns A, Wellings A - Real-Time Systems ans Programming Languages, AddWesley, California 1996.
61. K.E.Arzen - Real-Time Control Systems, curs Departamental Automatic Control de la Institutul de Tehnologie Lund.
62. GRAFCET, <http://www.lurpa.ens-cachan.fr/grafcet.html>.
63. P. Raghavan, Amol Lad, Sriram Neelakandan - Embedded Linux System Design And Development, Auerbach Publications Taylor & Francis Group, 2006.
64. J. Kiszka - The Real-Time Driver Model and First Applications, University of Hannover.
65. Mark Mitchell, Jeffrey Oldham, and Alex Samuel - Advanced Linux Programming, NewRiders, 2001.

Capitolul 4

66. Katsuhiko Ogata - Modern Control Engineering (4th Edition), Prentice Hall,2007.
67. N. S. Nise - Control Systems Engineering 2nd Ed., Addison-Wesley, NY, 1995.
68. H. Khalil - Nonlinear Systems, 2nd Ed., Prentice Hall, New Jersey, 1996.
69. Perruquetti Wilfrid, Jean Pierre Barbot - Sliding mode control in engineering, Marcel Dekker, 2002.
70. D. Young s.a. - A Control Engineer's Guide to Sliding Mode Control, IEEE Trans. On Control Systems Technology v. 7 no. 3, pp. 328-342, 1999.
71. J. Slotine and W. Li - Applied Nonlinear Control, Prentice Hall, New Jersey, 1991.
72. Utkin, V.I. -Sliding Modes in Optimization and Control Problems, Springer Verlag, New York, 1992
73. Utkin, V.I. - Variable Structure System with Sliding Modes. IEEE Transactions on Automatic Control, 1977.
74. Edwards, C. and Spurgeon, S.K. - Sliding Mode Control: Theory and Applications. Taylor & Francis, 1998.
75. Razvan Solea, Urbano Nunes - Trajectory Planning And Sliding Mode Control For Wmr Trajectory-Tracking And Path-Following Respecting Human Comfort Travel, Controlo 2006.
76. Tewari, A. - Modern Control Design With Matlab and Simulink, John Wiley & Sons, Ltd. , 2003.
77. A.Gelb, J.F.Kasper, R.A.Nash, C.F.Price, and A.A.Sutherland - Applied optimal estimation, The MIT Press, Cambridge, Massachusetts, 1974.
78. P.S.Maybeck - Stochastic Models, Estimation, and Control, Mathematics in Science and Engineering: A Series of Monographs and Textbooks, vol. 141, Academic Press, Inc., New York, USA, 1979.

79. P. S. Maybeck - The Kalman Filter: An Introduction to Concepts, in Autonomous Robot Vehicles, I.J. Cox, G. T.Wilfong (eds), Springer-Verlag, 1990.
80. J. Mendel - Lessons in Digital Estimation Theory , Prentice-Hall, 1987.
81. Papoulis J - Probability, Random Variables and Stochastic Processes, McGraw-Hill, 1965.
82. Ali Zolghadri – Algorithm for real time failure detection in kalman filters, IEEE Transactions on Automatic Control, Vol 41, No 10, 1996.
83. Maybeck P.S. și Hanlon R.D. – Performance enhancement of a multiple model adaptive estimator, IEEE Transactions on Aerospace and Electronic Systems, Vol31 , No 4, 1995.
84. Maybeck P.S. și Hanlon R.D – Multiple Model Adaptive Estimation using a residual correlation kalman filter bank, IEEE Transactions on Aerospace and Electronic Systems, Vol36, No2, 2000.
85. Yahya Chetouani - Using the Kalman Filtering for the Fault Detection and Isolation (FDI) in the Nonlinear Dynamic Processes, IJCRE 2008, Vol 6, A43.