Adaptive Sliding Mode Controller Design for Mobile Robot Fault Tolerant Control. Introducing ARTEMIC.

Cristian Axenie
Automation and Industrial Informatics Department
Computer Science Faculty from "Dunarea de Jos" University
Stiintei Street, Galati, Romania
Email : cristian.axenie@gmail.com
Web : http://robotics.viviti.com

Daniela Cernega, Assistant Professor, PhD
Control Systems and Applied Informatics
Computer Science Faculty from "Dunarea de Jos" University
Stiintei Street, Galati, Romania
Email : Daniela.Cernega@ugal.ro

Abstract – **Current real-time applications should timely deliver synchronized data-sets, minimize latency in their response and meet their performance specifications in the presence of disturbances and faults. The adaptive features of the designed controller are present at the lower control level using specific artificial intelligence techniques. Fuzzy inference system design is the fundamental element to generate an adaptive nonlinear controller for the robot operation in the presence of disturbances and modeling inaccuracies. This paper introduces an adaptive real-time distributed control application with fault tolerance capabilities for differential wheeled mobile robots, named ARTEMIC. Specific design, development and implementation details will be provided in this paper.**

Index Terms—Mobile Robotics, Fault Tolerant Control, Sliding mode, EKF, Real-Time Linux, Distributed Control

I. INTRODUCTION

By accepting the challenge to ensure fault tolerant real-time control of a self designed and built wheeled differential mobile robot the author has 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 detection and identification for the minimal robot structure.[4] The main application is designed on a distributed, client-server pattern, based on a TCP/IP wireless communication. The applications base level is built over the interface with the actuators (H-bridge MOSFET driver and 2 DC motors) and sensors (encoders and bumpers) found in the robots 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 to ensure the robot's autonomy and performance in trajectory tracking operation. The fault tolerant module is based on an set of Extended Kalman Filters used to estimate the current robot position and is using a residual computation and a statistical analysis to determine if a fault occurred in the system and to discriminate between the faults. The third level is a responsible with the communication task, to the extent that it implements a data server and a control server to ensure reliable data and command flows over a wireless network to the other nodes in the distributed system. 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 data. Due to the need of interactivity when injecting the faults a software fault injection framework was developed at this level. Its purpose is to inject a certain fault at a certain moment in time chosen by the client user. ARTEMIC or Autonomous Real Time Embedded Mobile robot Intelligent Controller was developed as the diploma thesis project by the author.

II. SERVER-SIDE APPLICATION DESCRIPTION

Next an extended description of the server-side application running on the robot embedded computer is given. All architectural and functional aspects will be presented, with focus on both software development elements and implemented control engineering concepts. The server-side application is a real-time Linux C/C++ application. The real-time capabilities were obtained at the OS level by modifying the standard Linux kernel with the RTAI (Real Time Application Interface) patch based on ADEOS (Adaptive Domain Environment for Operating Systems)to the extent that the enhanced micro- kernel can ensure a proper preemption level and I/O latency minimization. The hardware interface with the sensors and actuators is ensured by a data acquisition card (NI-PCI6024E) managed by a powerful kernel-space real-time driver and a rich API in user-space, both part of the COMEDI (Linux COntrol and MEasurement Device Interface). To properly manage the execution of the two main tasks, namely the control task and the monitoring task the application is organized on an abstract Grafcet (Graphe Fonctionnel de Controle des Etapes et Transitions) which supports the higher level parallelism in the execution of the two tasks. The control task is comprised of two inner speed control PID loops (running at 50ms) for the 2DC motors and a main robot position control loop based on a sliding mode controller (running at 200ms). Concurrently, the monitoring and diagnosis task receives sensor data which is then processed by a set of 5 Extended

978-1-4244-6886-7/10/$26.00 ©2010 IEEE

Kalman Filters every 50 ms. By examining the real and predicted data the application can decide if a fault occurred and what kind of fault based only on sensor information using a simple thresholding mechanism and a statistical test on the residuals. Next an architectural overview of the whole application is depicted in figure 1.

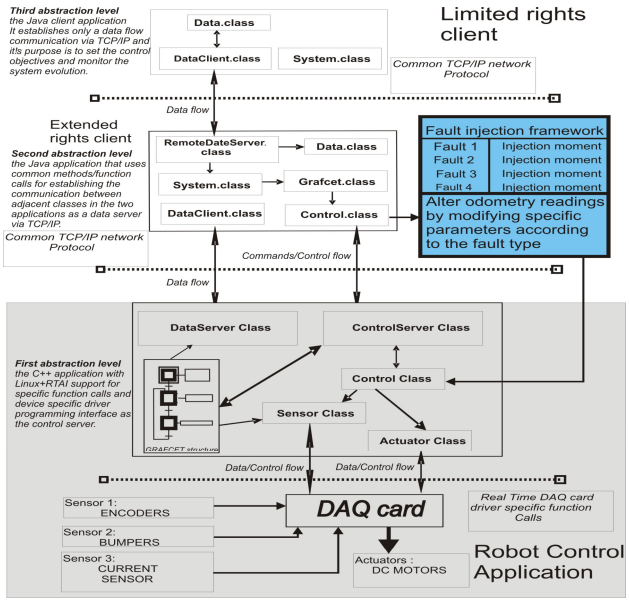

Fig. 1. Main application architecture.

A. Base level description

The base level is the main support for the control and monitoring tasks to the extent that the sensors and actuators are interfacing with the software by the real-time data I/O features given by the Linux-RTAI and COMEDI tools. Next the presentation shall focus on the base level of the server- side application running on the embedded robot computer. The specific I/O functions are presented in the application context. The server-side application is described in figure2.

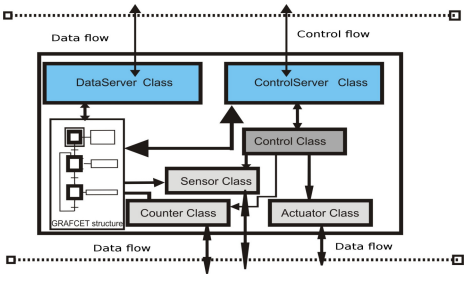

Fig. 2. Server-side application architecture

To properly understand how the robot control application works, a typical operation context is described, emphasizing class communication during execution. Each sampling period the application reads the data through the DAQ card from the encoders and computes, according to the read values, a control signal for the 2 DC motors. The specific functions to obtain a value from a sensor or to set a value to an actuator are comprised in the Counter, Sensor and Actuator classes. The outer sliding mode loop computes the control signal for the robot to achieve the next point in the trajectory in a timely manner and then the computed control signal is transformed in speed set-points for the inner PID loops. The monitoring and diagnosis task that runs concurrently is fed with sensor data, in fact the odometry computations and each sampling period the 5 EKF are testing if a fault occurred and if occurred, what kind of fault is it. The fault tolerant control module is then issuing specific warning messages to the client application regarding the current status of the robot operation. Next an in depth description of the two tasks running on the embedded robot controller is given.

B. Control algorithm level description

As mentioned earlier the robot is operating in trajectory tracking mode so its main objective is to follow a certain trajectory under time constraints. The control algorithm goal is to minimize the position errors, namely the longitudinal error, the lateral error and the heading error. To ensure proper error convergence in the presence of disturbances and modeling uncertainties a sliding mode controller was chosen. A sliding mode controller based on the kinematic model of the robot was used. By using a sliding mode controller a trade-off between tracking performance and parametric uncertainty was made, to the extent that it provides a good solution in maintaining the stability and consistent performance in the face of modeling imprecision and disturbances.[2] Next the complete robot control system is presented in figure3.

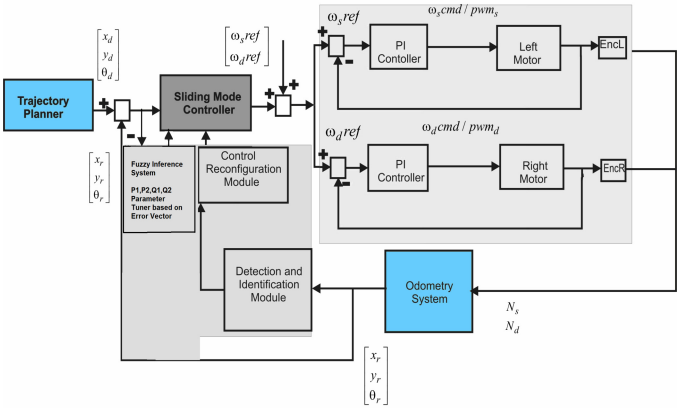

Fig. 3. Control system architecture.

The robot should track the reference parametric curve with time constraints and real-time control signal computation is critical. The idea behind the trajectory tracking operation mode considers a virtual robot that tracks the reference trajectory with no error and "pulls" the real robot after it to the extent that the controller reacts to minimize the errors. Let the robot be defined by the next position vector (state vector) $[x_r, y_r, \theta_r]^T$ and we define the error vector as $[x_e, y_e, \theta_e]^T = [x_d, y_d, \theta_d]^T - [x_r, y_r, \theta_r]^T$,(1) where $[x_d, y_d, \theta_d]^T$ is the virtual robot state vector, the set-point vector to track. Next the robot's kinematic model and tracking error computation is considered.

$$\begin{cases} \dot{x}_r = v_r \cos\theta \\ \dot{y}_r = v_r \sin\theta, \\ \dot{\theta}_r = \omega_r \end{cases} \begin{cases} \dot{x}_d = v_d \cos\theta_d \\ \dot{y}_d = v_d \sin\theta_d, \\ \dot{\theta}_d = \omega_d \end{cases} \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_d \\ y_r - y_d \\ \theta_r - \theta_d \end{bmatrix} \quad ,(2)$$

The main objective was to design a stable and robust controller for robot position control that should output a control signal vector $[v_c, \omega_c]$ (linear and angular speeds) in the presence of disturbances. The error derivative is given by

$$\begin{aligned} \dot{x}_e &= -v_d + v_r \cos(\theta_e) + y_e \omega_d \\ \dot{y}_e &= v_r \sin(\theta_e) - x_e \omega_d \\ \dot{\theta}_e &= \omega_r - \omega_d \end{aligned} \quad ,(3)$$

where the set-point values for the speed control loops are v_d and ω_d. [8] The set-point for the main robot position control loop is generated by a trajectory planner based on 5th degree equations for smoothing the robot's operation and control effort. The trajectory planner outputs linear and angular speed and acceleration set-points at each sampling period. These values are transformed using inverse kinematics equations in vector form $[x_d, y_d, \theta_d]^T$. The next step was to design the switching surfaces for the sliding mode controller such that it can ensure proper error convergence. The first switching surface was chosen to ensure lateral error convergence and is given by the next equation $s_1 = \dot{x}_e + k_1 x_e$,(4) To ensure proper longitudinal and heading error convergence the second switching surface combines the two objectives and is given by $s_2 = \dot{y}_e + k_2 y_e + k_0 \text{sgn}(y_e)\theta_e$, (5) , where the k_1, k_2, k_3 parameters are specific weights associated to each error component. Practically the control signal could be computed using $\dot{s} = -Qs - P\,\text{sgn}(s)$, (6), where Q,P are positive constant values that are determining the error convergence speed.[5] By deriving the surfaces equations and replacing the error components one can easily obtain the control signal vector under the form of

$$\dot{v}_c = \frac{-Q_1 s_1 - P_1 \text{sgn}(s_1) - k_1(\dot{x}_e) - \dot{\omega}_d y_e - \omega_d \dot{y}_e + \dot{v}_r \theta_e \sin(\theta_e) + \dot{v}_d}{\cos(\theta_e)} \quad ,(7)$$

$$\dot{\omega}_c = \frac{-Q_2 s_2 - P_2 \text{sgn}(s_2) - k_2(\dot{y}_e) + \dot{\omega}_d x_e + \omega_d \dot{x}_e - \dot{v}_r \sin(\theta_e)}{v_r \cos(\theta_e) + k_0 \text{sgn}(y_e)} + \dot{\omega}_d$$

To ensure a good convergence speed of the robot state vector to the switching surface (which in fact gives the desired system dynamics) and also to eliminate chattering (caused by fast control signal switches) the P and Q terms should be cautiously chosen. To ensure the stability of the system we defined a Lyapunov candidate function given by $V = 0.5 s^T s$ and with $\dot{V} = s_1 \dot{s}_2 + s_2 \dot{s}_1 = -s^T Qs - P_1 |s_1| - P_2 |s_2|$, (8), the main condition is that $V\dot{V} < 0$ achieved if $Q_i >= 0$ and $P_i >= 0$. [2] As mentioned earlier the sliding mode controller outputs angular speed set-points for the inner PID loops. The PID controllers for the two DC motors were implemented using first-order-hold method.[10] The PID loops are running at 20Hz and the specific parameters were tuned to meet a fast response time (0.5s) and a small overshoot (10%). Next some results from the closed loop fault-free operation of the robot are given to analyze the robot behavior on a specific trajectory.

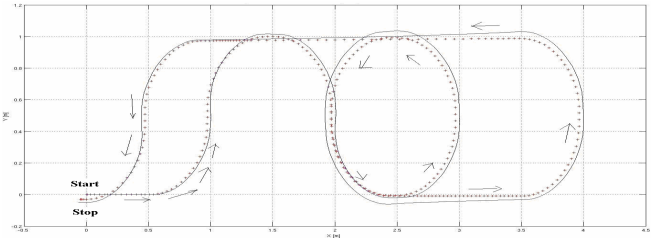

Fig. 4. Robot operation in trajectory tracking with sliding mode controller in fault free operation.

The real robot trajectory is represented by a continuous line and the reference trajectory is given with crosses. The movement direction is indicated by arrows. As one can see the position error is kept in acceptable limits by the robust sliding mode controller and the robot can achieve its objective in time. Once more one can observe that good performance is achieved in using the sliding mode controller. Another important aspect regards the analysis of the error components in the fault free operation, because it can be considered as a proper validation for the controller.[6] Next the three error components are depicted.

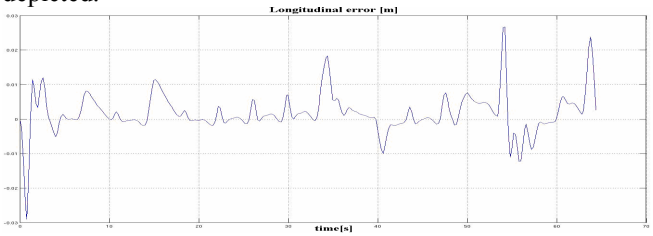

Fig. 5. Robot longitudinal error x_e.

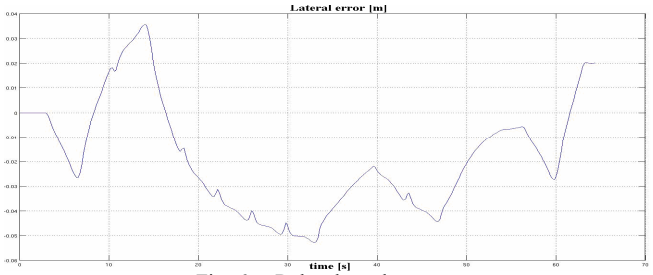

Fig. 6. Robot lateral error y_e.

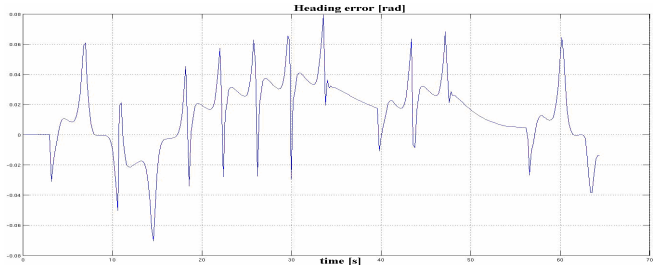

Fig. 7. Robot heading error θ_e.

One can see that the error convergence is properly ensured by the controller. The longitudinal error in the fault free operation is under 3cm, the lateral error is under 6cm and the heading error in under 8rad. Experiments have shown that even for simpler trajectories (simple 2m line, 0.75m radius circle) the controller behaves well and despite the fact that the robot structure had some hardware limitations (encoder resolution, 500PPR) the real-time control goals were met even for slightly high sampling periods ; all the results are validated with the earlier error analysis.

Next the adaptive Sliding mode controller implementation details are given. The main focus in synthesizing an adaptive Sliding mode controller was to gain flexibility and consistency in tuning the controller parameters. So, a fuzzy inference system was developed to properly adjust the P_1, P_2, Q_1, Q_2 parameters that ensure the convergence speed. The parameter tuning is made by taking account of the main control objectives. Chattering reduction and faster system state trajectory convergence towards the reference sliding surface are desired. The implemented fuzzy adaptive module is a Mamdani fuzzy inference system model. It is based on a MIMO pattern using 3 inputs and 4 outputs. The inputs are the 3 components of the position error (longitudinal, lateral and heading error) while the outputs are the 4 parameters of the sliding mode controller. The fuzzy inference system was designed using a MIN implication method, a MAX aggregation method, two defuzzification methods namely centroid and bisector (for comparison) and triangular membership functions. The first test was deployed using the centroid defuzzification method defining 3 membership functions for each input and output defining minimum, medium and maximum deviations/errors. The variation ranges that define the membership functions were chosen relatively with some initial experiments with the robot. A set of 27 rules were defined for the inference module. Following the error analysis is presented for the standard and the adaptive controller.

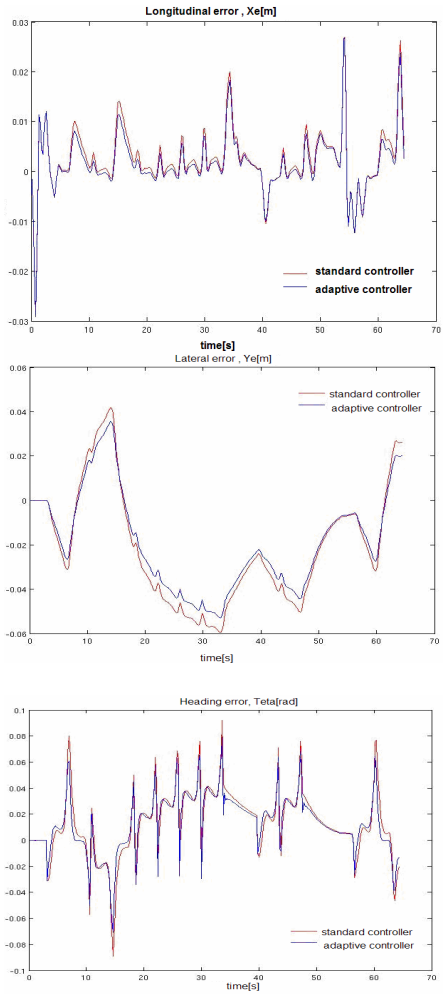

Fig. 8. Error analysis in the standard and adaptive case

One can observe that in the case of the adaptive sliding mode controller the error is slightly reduced, but the variation is maintained due to the fact that the fuzzy inference system cannot determine the optimal value at each sampling period, so an parametric optimization method would be required. The fixed controller parameters (i.e. Q_1= 0.05, Q_2=0.05, P_1=0.75, P_2=1.75) were chosen after a trial-and-error sequence and together with the robustness of the sliding mode controller these proved to be suitable for certain trajectories. Due to the need of good performance in new trajectories, the proposed method is suitable and can enhance the performance by redesigning some components of the fuzzy inference system. Following the initial Mamdani fuzzy inference system is considered with a SUM aggregation method and bisector

defuzzification method. The most significant modification and performance enhancement can be observed in the heading error variation. Next, one can properly observe that the new feature of the fuzzy inference system increases the performance index in heading, but maintains a constant performance in the longitudinal and lateral positioning.

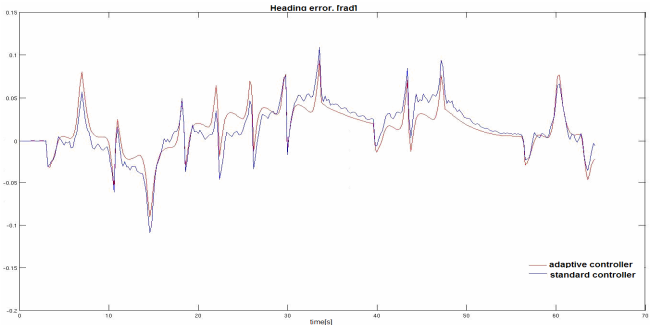

Fig.9. Heading error comparison in bisector defuzzification

The design time can be reduced by using the adaptive controller based on the fuzzy inference system due to the fact that it offers good performance levels and doesn't need time consuming trial-and-error sequences to tune the nonlinear controller parameters. The separate fuzzy module can be merged with the sliding mode controller to obtain a compact adaptive sliding mode controller, that combines robustness, stability, precision and flexibility in the presence of uncertainties (i.e. structured/non-structured). So the presented analysis can validate a suitable method to design a robust controller to suit the needs of a fault tolerant system operating with time restrictions and in the presence of disturbances.
Following the next level of the application is presented.

C. Monitoring and diagnosis level description

Due to hardware limitations in injecting faults the application is also responsible to inject faults by software using a fault injection framework. By benefiting of a certain degree of robustness given by the adaptive sliding mode controller the application resides on a simple mechanism for fault detection, identification and support for control reconfiguration based on the Extended Kalman Filter (EKF). Hence, the application implements a set of 5 EKF that form the fault detection and identification module. The current application considers a fault benchmark comprised of faults manifested by system's parameter variation specifically mechanical faults that alter the robot behavior. Two fault classes were defined each containing two types of faults. The first class of faults introduces the flat tire fault for each of the robot wheels. At the implementation level this fault exhibits by diminishing the wheel radius with a certain value. As a consequence the whole kinematic model of the robot is modified and visible modifications in the robot's behavior will be present. The second class of faults introduces a periodic variation of the wheel radius (a bump, caused hypothetically by an object attached to the wheel). Hence, for a small time period the variation will manifest itself and alter the robot operation. As mentioned earlier a fault injection framework was implemented and its main purpose is to alter the values from the sensors in odometry specific computations. Hence, when no fault is injected the robot will behave normally (as presented when the control level was described) but when a fault was injected the robot position determining system (in fact the control system feedback) will feed altered data to the controller as if a physical fault occurred. Next a depiction of the fault detection and identification module is given.

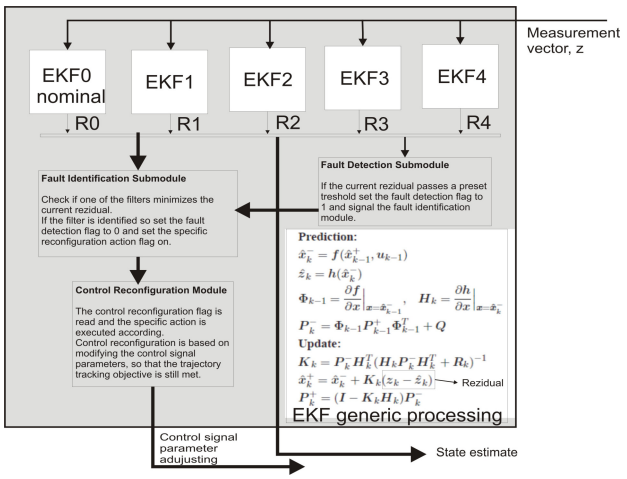

Fig. 10. Fault detection and identification module.

Each of the five EKF encapsules a kinematic model of the robot but with different parameters. The main idea resides on the fact that for the same input vector, z, with noise, each filter generates a prediction of the robot's state vector. Each filter is associated with a certain fault type. EKF0 is the nominal filter corresponding to the fault free robot operation. EKF1 filter contains the same robot kinematic model but with modified parameters to emphasize the right tire flat fault, so the right wheel radius has a smaller value. Hence, the prediction of EKF1 will be the robot state vector if a right tire flat occurred. The EKF2 filter is similar to EKF1 excepting it encapsules the left tire flat fault specific model. In the same way the other two filters associated with the second fault class, EKF3 and EKF4, are predicting the robot's state vector in the presence of a second fault class fault. Besides the state estimate each EKF generates a measurement vector estimate during the prediction stage, which is used in the correction stage of the filter.[3] Basically the detection method is based on a nominal residual computation and a simple thresholding test on the nominal residual which can output that a fault

occurred. It is the only one affected by the fault occurrence so it will step out from some preset value intervals. [9] When identifying the fault the application just finds the smallest residual from the 5 ones, because when a fault occurs the main measurement vector is altered and it nears the predicted measurement vector by the EKF encapsuling that fault behavior.[1] Next, specific implementation details of the EKFs are presented. Starting from the generic structure during the implementation some simplifying hypothesis were issued and we know that Q is the process noise covariance matrix, R is the measurement covariance matrix, $\sigma_x = \sigma_y = \sigma_\theta = 0.1$ are the standard deviations for the process noise, σ_{meas}=0.01 is the standard deviation of the measurement noise. To correctly detect if a fault occurred some statistical tests on the residual are required. The statistical parameters of the residual (mean and variance) will be compared on-line at each sampling period with the values obtained for these parameters in fault-free operation. In fault-free operation the residual is a white noise sequence with 0 mean and with a variance of

$\eta_k = H_k(\hat{x}_k^-)P_k^- H_k^T(\hat{x}_k^-) + R$,(9). Basically if a fault occurs in the system it will determine a modification of the next residual based standard sequence,

$$\eta s_k = (H_k(\hat{x}_k^-)P_k^- H_k^T(\hat{x}_k^-) + R)^{-1/2}(y_k - h_k(\hat{x}_k^-)) = \eta_k^{-1/2} r_k, (10)$$

The goal is to determine the estimate of the real value of a sample given using the next relation where N is the total samples number,

$$\hat{\bar{\eta}}_s = \frac{1}{N}\sum \eta_{sj} \quad , (11)$$

So, in the hypothesis H_0 of no fault $\hat{\eta}_s$ has a gaussian distribution with 0 mean and 1/N covariance. Over a certain acceptance threshold the H_0 hypothesis is no longer true and so a new hypothesis H_1 is now true marking a fault occurrence. Next some experimental results are presented. First it is useful to analyze the residuals generated by the nominal filter when a fault occurs, in fact these residuals are those predicted by the fault embedding filters, from the moment the fault occurs. For example if a right tire flat fault occurs at sample 100 in the execution the residual evolution from 100th sample is the predicted residual by the EKF1.

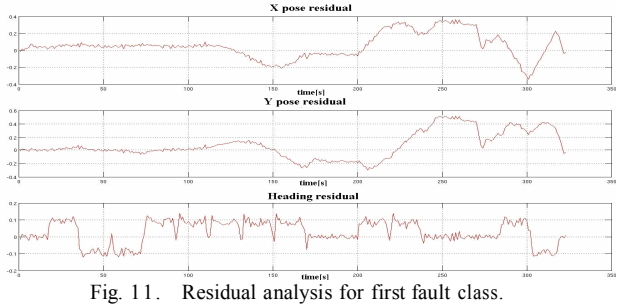

Fig. 11. Residual analysis for first fault class.

From this residual analysis one can see that each EKF is sensitive to a certain fault type and using these properties the detection and identification modules were built. Assuming that a right tire flat fault was injected at moment t=10s and the new radius of the wheel is 3cm smaller than the initial value the behavior of the robot is altered as one can see in the next figure.

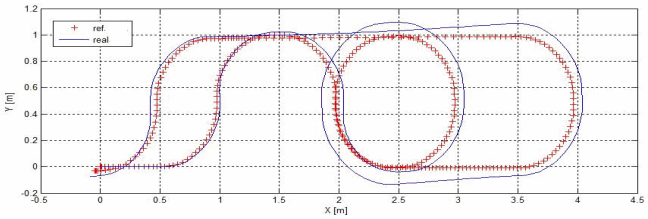

Fig. 12. Robot faulty operation vs. reference trajectory.

Next an extended position error analysis over the fault free and faulty operation is given (red marks fault free operation and blue marks the faulty operation).

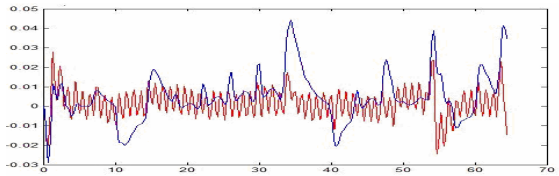

Fig. 13. Longitudinal error comparison fault-free/faulty operation.

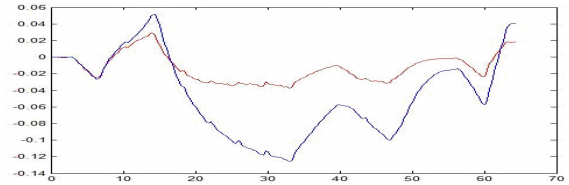

Fig. 14 Lateral error comparison fault-free/faulty operation.

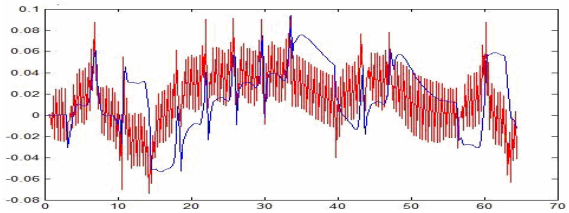

Fig. 16. Heading error comparison fault-free/faulty operation.

As one can see there are major variations from the nominal behavior after a fault occurred in the system. It is important to mention that the thresholding mechanism reacts if at least two components of the residual vector are bigger than the preset values. So, the fault could be visible in all the residual components or not. As a typical operation context when a fault is detected, it is then identified and the client application is informed what kind

of fault occurred. At the moment only one fault can be detected and identified, no fault queue or priority system are implemented. Next the adaptive sliding mode case is described comparatively with the standard sliding mode controller in regarding fault tolerance capabilities and performance. Following a brief description of the client application is given.

III. THE CLIENT-SIDE APPLICATION DESCRIPTION

The two main components of the application are linked through a wireless communication ensuring the client application to be responsible with the external events that determine the robot operation. It implements a similar structure with the embedded application and supports the fault injection framework. Besides the interactive task of injecting faults the client application is responsible to choose the Grafcet structure that encapsules the execution rules for the control algorithm and the specific supported actions (start/pause/stop) for the robot operation, also it offers the possibility to choose between multiple trajectories and to visualize the robot's current position. Using a full load of the embedded system latency measurements were taken. So, by using a 50ms sampling period for the inner control loop and the monitoring and diagnosis tasks and a 200ms sampling period for the robot position controller good average ms level latencies were obtained. The two possible Grafcet execution schemes serial/ parallel assume a serial or parallel execution of user defined tasks for the robot. Currently the two implemented tasks, real-time control and monitoring and diagnosis must run concurrently to ensure consistent and valid results in the fault tolerant control problem.
Following a short summarizing overview over the paper is given emphasizing the main advantages and some future work ideas.

IV. CONCLUSIONS AND FUTURE WORK

ARTEMIC was designed as a flexible application dedicated to mobile robotics fault tolerant control. Its architecture is open and supports extensibility at a "plug-and-play" level because at the lowest level it supports and offers a simple mechanism to add new hardware to the robot and provides a generic and compatible interface with the hardware. As future work a port of the developed application from the current Intel Celeron based embedded computer to a MPC8315ERDB PowerPC platform is started. The new platform uses an enhanced RTOS named Xenomai that is a super-set of RTAI and it provides better performance and it is dedicated to embedded architectures. Another direction in the future work regards the possibility to combine multiple fault detection and identification methods to gain the advantages of each composing method and obtain a hybrid and efficient mechanism. This approach shall be combined with the fuzzy adaptive sliding mode controller to ensure faster error convergence. Next figure introduces the ARTEMIC powered robot.

Fig. 17. The ARTEMIC powered mobile robot.

V. REFERENCES

[1] Patton R., Frank P., Clark R. Fault Diagnosis in Dynamic Systems Theory and Applications Prentice Hall, New York, 1989.
[2] Razvan Solea, Urbano Nunes Trajectory Planning and sliding mode Control for WMR Trajectory Tracking and Path Following Respecting Human Comfort Travel , CONTROLO Conference Portugal, 2006.
[3] Mayback P.S.,Cox I.J.,Wilfong , The Kalman Filter: An Introduction to Concepts in Autonomous Robot Vehicles Springer-Verlag, 1990
[4] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo Robotics, Modelling, Planning and Control Springer, 2009
[5] Gazenfer Rustamov sliding modes in Finite-Time Control Systems with Variable Structure Proceedings of the 9th WSEAS International Conferenceon Automatic Control - Modeling and Simulation, Istanbul, Turkey, May 27-29, 2007
[6] I. Susnea, G. Vasiliu, A. Filipescu Real-time, embedded fuzzy control of the Pioneer3-DX robot for path following 12th WSEAS International Conference on SYSTEMS, Heraklion, Greece, July 22-24, 2008
[7] Burns A., Wellings A. Real-Time Systems and Programming Languages (4th edition) AddWesley, California, 2009.
[8] Siegwart R., Nourkbash. I. An Introduction to Autonomous Mobile Robots, MIT press, 2004
[9]Evgenia Suzdaleva,Utia AV , Initial Conditions for Kalman Filtering: Prior Knowledge Specification, Proceedings of the 7th WSEAS International Conference on Systems Theory and Scientific Computation, Athens, Greece, August 24-26, 2007
[10] Yong Xu, Wansheng Tang et al., Control Design for Uncertain Systems Based on Integral-Type Sliding Surface, Proceedings of the 6th WSEAS International Conference on Robotics, Control and Manufacturing Technology, Hangzhou, China, April 16-18, 2006

Acknowledgment

This work was supported by CNCSIS-UEFISCSU, project number PNII - IDEI 506/2008.

Cristian Axenie Graduated in Automation and Applied Informatics at the Computer Science Faculty from "Dunarea de Jos" University. The main author's main focus is in the fault tolerant control and nonlinear control design for mobile robots and autonomous vehicles and embedded systems.
Daniela Cernega is Assistant Professor, PhD at the Control Systems and Applied Informatics Department from "Dunarea de Jos" University. The main interests comprise Discrete Event Systems, Collaborative Robotics, Software Systems for Flexible Assembly Lines and Factory Automation.